[How to get the specific data through api by using flutter\_bloc](https://stackoverflow.com/questions/59512571/how-to-get-the-specific-data-through-api-by-using-flutter-bloc)

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MultiBlocProvider(

providers: [

BlocProvider<PrayerBloc>(

create: (BuildContext context) => PrayerBloc(repository: PrayerRepositoryImpl()),

),

BlocProvider<MethodBloc>(

create: (BuildContext context) => MethodBloc(methodRepository: MethodRepositoryImpl()),

),

],

child: HomePage(),

);

Home Page

class HomePage extends StatefulWidget {

@override

\_HomePageState createState() => \_HomePageState();

}

class \_HomePageState extends State<HomePage> {

PrayerBloc prayerBloc;

@override

void initState() {

super.initState();

prayerBloc = BlocProvider.of<PrayerBloc>(context);

prayerBloc.add(FetchPrayerEvent());

}

@override

Widget build(BuildContext context) {

return MaterialApp(

home: Builder(

builder: (context) {

return Material(

child: Scaffold(

appBar: AppBar(

title: Text("Prayer API"),

),

body: Container(

child: BlocListener<PrayerBloc, PrayerState>(

listener: (context, state) {

if (state is PrayerErrorState) {

Scaffold.of(context).showSnackBar(

SnackBar(

content: Text(state.message),

),

);

}

},

child: BlocBuilder<PrayerBloc, PrayerState>(

builder: (context, state) {

if (state is InitialPrayerState) {

return buildLoading();

} else if (state is PrayerLoadingState) {

return buildLoading();

} else if (state is PrayerLoadedState) {

return buildArticleList(state.item);

} else if (state is PrayerErrorState) {

return buildErrorUi(state.message);

}

},

),

),

),

),

);

},

),

);

}

Widget buildLoading() {

return Center(

child: CircularProgressIndicator(),

);

}

Widget buildErrorUi(String message) {

return Center(

child: Padding(

padding: const EdgeInsets.all(8.0),

child: Text(

message,

style: TextStyle(color: Colors.red),

),

),

);

}

Widget buildArticleList(List<Item> item) {

return ListView.builder(

itemCount: item == null ? 0 : item.length,

itemBuilder: (BuildContext ctx, int pos) {

return new Container(

child: new Center(

child: new Column(

crossAxisAlignment: CrossAxisAlignment.center,

children: <Widget>[

new Card(

child: Padding(

padding: const EdgeInsets.all(8.0),

child: new Container(

child: Column(

children: <Widget>[

Padding(

padding: EdgeInsets.only(top: 10.0),

),

Row(

children: <Widget>[

Text("Fajr"),

Padding(

padding: EdgeInsets.only(left: 50.0),

),

Text(item[pos].fajr),

],

),

Row(

children: <Widget>[

Text("Dhuhr"),

Padding(

padding: EdgeInsets.only(left: 30.0),

),

Text(item[pos].dhuhr),

],

),

Builder(

builder: (context)=>

RaisedButton(

onPressed: (){

Navigator.push(

context,

MaterialPageRoute(builder: (context) => SettingsPage()),

);

},

),

)

],

),

),

),

)

],

),

),

);

},

);

}

Prayer\_bloc.dart

class PrayerBloc extends Bloc<PrayerEvent, PrayerState> {

PrayerRepository repository;

PrayerBloc({@required this.repository});

@override

PrayerState get initialState => InitialPrayerState();

@override

Stream<PrayerState> mapEventToState(

PrayerEvent event,

) async\* {

if (event is FetchPrayerEvent) {

yield PrayerLoadingState();

try {

List<Item> item = await repository.getItem();

yield PrayerLoadedState(item: item);

} catch (e) {

yield PrayerErrorState(message: e.toString());

}

}

}

}

PlayerEvent.dart

abstract class PrayerEvent extends Equatable {}

class FetchPrayerEvent extends PrayerEvent {

@override

// TODO: implement props

List<Object> get props => null;

}

PlayerState.dart

abstract class PrayerState extends Equatable {

const PrayerState();

}

class InitialPrayerState extends PrayerState {

@override

List<Object> get props => [];

}

class PrayerLoadingState extends PrayerState {

@override

List<Object> get props => [];

}

class PrayerLoadedState extends PrayerState {

List<Item> item;

PrayerLoadedState({@required this.item});

@override

List<Object> get props => null;

}

class PrayerErrorState extends PrayerState {

String message;

PrayerErrorState({@required this.message});

@override

List<Object> get props => [message];

}

PrayerRepository.dart

abstract class PrayerRepository {

Future<List<Item>> getItem();

}

class PrayerRepositoryImpl implements PrayerRepository {

@override

Future<List<Item>> getItem() async {

var response = await http.get("https://muslimsalat.com/riyadh.json?key=");

if (response.statusCode == 200) {

var data = json.decode(response.body);

List<Item> item = Welcome.fromJson(data).items;

return item;

} else {

throw Exception();

}

}

}

SettingsPage

class SettingsPage extends StatefulWidget {

@override

\_SettingsPageState createState() => \_SettingsPageState();

}

class \_SettingsPageState extends State<SettingsPage> {

int selectedRadio;

@override

void initState() {

super.initState();

selectedRadio=0;

}

setSelectedRadio(int val){

setState(() {

selectedRadio=val;

});

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(),

body: Container(

child: ListView(

children: <Widget>[

BlocBuilder<MethodBloc,MethodState>(

builder: (context,state){

return Column(

mainAxisAlignment: MainAxisAlignment.start,

children: <Widget>[

Container(

padding: EdgeInsets.all(15.0),

child: Text(

"Prayer Methods",

style: TextStyle(fontSize: 20, fontWeight: FontWeight.bold),

),

),

Column(

children: <Widget>[

RadioListTile(

value: 1,

groupValue: selectedRadio,

activeColor: Colors.black,

title: Text(

"Egyptian General Authority of Survey",

),

onChanged: (val) {

print(val);

setSelectedRadio(val);

}),

RadioListTile(

value: 2,

groupValue: selectedRadio,

activeColor: Colors.black,

title: Text(

"University Of Islamic Sciences, Karachi (Shafi)",

),

onChanged: (val) {

print(val);

setSelectedRadio(val);

}),

FloatingActionButton(

onPressed: (){

Navigator.pop(context);

BlocProvider.of<MethodBloc>(context).add(MethodChangedEvent(method: selectedRadio)); //I have try this code in onpressed but unfortunately not succeed

print(selectedRadio);

},

child: Text('Save')

)

],

),

],

);

},

)

],

),

),

);

}

}

MethodBloc.dart

class MethodBloc extends Bloc<MethodEvent, MethodState> {

MethodRepository methodRepository;

MethodBloc({@required this.methodRepository});

@override

MethodState get initialState => InitialMethodState();

@override

Stream<MethodState> mapEventToState(

MethodEvent event,

) async\* {

if(event is MethodChangedEvent){

yield MethodLoadingState();

try {

List<Item> item = await methodRepository.getMethod(event.method);

yield MethodLoadedState(item: item);

} catch (e) {

yield MethodErrorState(message: e.toString());

}

}

}

}

MethodEvent.dart

abstract class MethodEvent extends Equatable {

const MethodEvent();

}

class MethodChangedEvent extends MethodEvent {

final int method;

MethodChangedEvent({this.method}) : assert(method != null);

@override

List<Object> get props => null;

}

MethodState.dart

abstract class MethodState extends Equatable {

const MethodState();

}

class InitialMethodState extends MethodState {

@override

List<Object> get props => [];

}

class MethodLoadingState extends MethodState {

@override

List<Object> get props => [];

}

class MethodLoadedState extends MethodState {

List<Item> item;

MethodLoadedState({@required this.item});

@override

List<Object> get props => null;

}

class MethodErrorState extends MethodState {

String message;

MethodErrorState({@required this.message});

@override

List<Object> get props => [message];

}

MethodRepository.dart

abstract class MethodRepository{

Future<List<Item>> getMethod(int method);

}

class MethodRepositoryImpl implements MethodRepository {

@override

Future<List<Item>> getMethod(int method) async {

var response = await http.get("https://muslimsalat.com/riyadh/$method.json?key=");

if (response.statusCode == 200) {

var data = json.decode(response.body);

List<Item> item = Welcome.fromJson(data).items;

return item;

} else {

throw Exception();

}

}

}

class Welcome {

String title;

String query;

String welcomeFor;

int method;

String prayerMethodName;

String daylight;

String timezone;

String mapImage;

String sealevel;

TodayWeather todayWeather;

String link;

String qiblaDirection;

String latitude;

String longitude;

String address;

String city;

String state;

String postalCode;

String country;

String countryCode;

List<Item> items;

int statusValid;

int statusCode;

String statusDescription;

Welcome({

this.title,

this.query,

this.welcomeFor,

this.method,

this.prayerMethodName,

this.daylight,

this.timezone,

this.mapImage,

this.sealevel,

this.todayWeather,

this.link,

this.qiblaDirection,

this.latitude,

this.longitude,

this.address,

this.city,

this.state,

this.postalCode,

this.country,

this.countryCode,

this.items,

this.statusValid,

this.statusCode,

this.statusDescription,

});

factory Welcome.fromJson(Map<String, dynamic> json) => Welcome(

title: json["title"],

query: json["query"],

welcomeFor: json["for"],

method: json["method"],

prayerMethodName: json["prayer\_method\_name"],

daylight: json["daylight"],

timezone: json["timezone"],

mapImage: json["map\_image"],

sealevel: json["sealevel"],

todayWeather: TodayWeather.fromJson(json["today\_weather"]),

link: json["link"],

qiblaDirection: json["qibla\_direction"],

latitude: json["latitude"],

longitude: json["longitude"],

address: json["address"],

city: json["city"],

state: json["state"],

postalCode: json["postal\_code"],

country: json["country"],

countryCode: json["country\_code"],

items: List<Item>.from(json["items"].map((x) => Item.fromJson(x))),

statusValid: json["status\_valid"],

statusCode: json["status\_code"],

statusDescription: json["status\_description"],

);

Map<String, dynamic> toJson() => {

"title": title,

"query": query,

"for": welcomeFor,

"method": method,

"prayer\_method\_name": prayerMethodName,

"daylight": daylight,

"timezone": timezone,

"map\_image": mapImage,

"sealevel": sealevel,

"today\_weather": todayWeather.toJson(),

"link": link,

"qibla\_direction": qiblaDirection,

"latitude": latitude,

"longitude": longitude,

"address": address,

"city": city,

"state": state,

"postal\_code": postalCode,

"country": country,

"country\_code": countryCode,

"items": List<dynamic>.from(items.map((x) => x.toJson())),

"status\_valid": statusValid,

"status\_code": statusCode,

"status\_description": statusDescription,

};

}

class Item {

String dateFor;

String fajr;

String shurooq;

String dhuhr;

String asr;

String maghrib;

String isha;

Item({

this.dateFor,

this.fajr,

this.shurooq,

this.dhuhr,

this.asr,

this.maghrib,

this.isha,

});

factory Item.fromJson(Map<String, dynamic> json) => Item(

dateFor: json["date\_for"],

fajr: json["fajr"],

shurooq: json["shurooq"],

dhuhr: json["dhuhr"],

asr: json["asr"],

maghrib: json["maghrib"],

isha: json["isha"],

);

Map<String, dynamic> toJson() => {

"date\_for": dateFor,

"fajr": fajr,

"shurooq": shurooq,

"dhuhr": dhuhr,

"asr": asr,

"maghrib": maghrib,

"isha": isha,

};

}

class TodayWeather {

int pressure;

String temperature;

TodayWeather({

this.pressure,

this.temperature,

});

factory TodayWeather.fromJson(Map<String, dynamic> json) => TodayWeather(

pressure: json["pressure"],

temperature: json["temperature"],

);

Map<String, dynamic> toJson() => {

"pressure": pressure,

"temperature": temperature,

};

}

Using Bloc and Provider in same App

Future main() async{

  WidgetsFlutterBinding.ensureInitialized();

  await SystemChrome.setPreferredOrientations([DeviceOrientation.portraitUp]);

  runApp(MyApp());

}

class MyApp extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return MultiProvider(

      providers: [

        ChangeNotifierProvider<ButtonState>(create: (\_)=>ButtonState(),),

        ChangeNotifierProvider<TestService>(create: (\_)=>TestService(),),

        Provider<LoginServices>(create: (\_)=>LoginServices(),),

        // Provider<DraftOffers>.value(value: draftOffers),

        ChangeNotifierProvider<OfferServices>(create: (\_)=> OfferServices(),),

        ChangeNotifierProvider<ProductServices>(create: (\_)=> ProductServices(),),

        ChangeNotifierProvider<CreateServices>(create: (\_)=> CreateServices(),),

        ChangeNotifierProvider<PickImage>.value(value:   PickImage(),),

        ChangeNotifierProvider<CreateProductServices>(create: (\_)=> CreateProductServices(),),

      ],

      child: MultiBlocProvider(

              providers: [

    BlocProvider(create:(\_)=>LoginBloc(signInRepository: FetchApi()))

              ],

              child: MaterialApp(

            debugShowCheckedModeBanner: false,

            title: 'Floradlite',

            // theme: ThemeData(

            //   primarySwatch: Colors.blue,

            //   visualDensity: VisualDensity.adaptivePlatformDensity,

            // ),

             initialRoute: '/',

            onGenerateRoute: RouteGenerator.generateRoute,

          ),

      ),

    );

  }

}

Fetch Varients in FloraDlite

Bloc.dart

class LoginBloc extends Bloc<VarientsEvent,LoginState> {

  final SignInRepository signInRepository;

 ProductsList productsList;

  LoginBloc({this.signInRepository}) : super(LoginInitState());

  @override

  Stream<LoginState> mapEventToState(VarientsEvent event) async\* {

   if (event is FetchButtunPress) {

    yield LoginLoadingState();

    try {

     productsList=await signInRepository.fetchVariants(event.productId);

     yield LoginLoaded(productsList: productsList);

    } on SocketException {

     yield LoginErrorState(error: NoInterNetExeption(messege:"No internet"));

         }

         catch (e) {

          yield LoginErrorState(error: UnkwonExeption(messege: "Uknownexeption"));

         }

        }

  }

}

 class NoInterNetExeption {

 var messege;

 NoInterNetExeption({this.messege});

 }

 class UnkwonExeption{

 var messege;

 UnkwonExeption({this.messege});

 }

 class HttpExeption{

  var messege;

  HttpExeption({this.messege});

}

Events.dart

import 'package:equatable/equatable.dart';

abstract class VarientsEvent extends Equatable{

  @override

  List<Object> get progs => [];

}

class FetchButtunPress extends VarientsEvent{

  final String productId;

  FetchButtunPress({this.productId});

  @override

  List<Object> get props => [productId];

}

State.dart

import 'package:equatable/equatable.dart';

import 'package:floradlite/models/products/ProductsList.dart';

abstract class LoginState extends Equatable{

  List<Object> get progs=>[];

}

class LoginInitState extends LoginState{

  @override

  List<Object> get props => [];

}

class LoginLoadingState extends LoginState{

  @override

  List<Object> get props => [];

}

class LoginLoaded extends LoginState{

 final ProductsList productsList;

 final bool val;

  LoginLoaded({this.productsList,this.val});

  @override

  List<Object> get props =>[];

}

class LoginErrorState extends LoginState{

  final error;

  LoginErrorState({this.error});

  @override

  List<Object> get props =>[];

}

Repository.dart

import 'dart:convert';

import 'package:floradlite/models/products/ProductsList.dart';

import 'package:floradlite/utils/my\_exception.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'package:floradlite/common/common\_constants.dart' as constants;

import 'package:http/http.dart' as http;

abstract class SignInRepository{

  Future<ProductsList>fetchVariants(String productId);

}

class FetchApi implements SignInRepository{

  ProductsList fp=new ProductsList();

  @override

  Future<ProductsList> fetchVariants(String productId) async {

try {

      SharedPreferences prefs = await SharedPreferences.getInstance();

      var headers = {

        "Accept": "application/json",

        "Authorization": "key=${prefs.getString(constants.App\_Accesskey)}"

      };

      String url = constants.Products\_Url +

          "?country=ind&storeid=" +

          prefs.getString(constants.Storeid) +

          "&productid=" +

          productId;

      print("variants url : " +

          constants.Products\_Url +

          "?country=ind&storeid=" +

          prefs.getString(constants.Storeid) +

          "&productid=" +

          productId);

      final drafts = await http.get(url, headers: headers);

      print("variants: " + drafts.body + url + headers.toString());

      if (drafts.statusCode == 200) {

        fp = ProductsList.fromJson(json.decode(drafts.body));

      }

    } on CustomException catch (e) {

      print(e);

    }

return fp;

  }

}

ProductListItem

BlocConsumer<LoginBloc,LoginState>(builder: (context,state){

                return  GestureDetector(

                    onTap: (){

           BlocProvider.of<LoginBloc>(context).add(

       FetchButtunPress(productId: productid

           ));

Navigator.push(context, new MaterialPageRoute(builder: (context)=>TestVarient()));

                    },

                    child: Container(

                      alignment: Alignment.topLeft,

                      padding: EdgeInsets.only(left: 10, right: 10, top: 8),

                      child: subHeadingText(name),));

                   },listener: (context,state){

                   }, ),

Display Fetched Data in Testvariant

import 'package:floradlite/bloc/bloc.dart';

import 'package:floradlite/bloc/events.dart';

import 'package:floradlite/bloc/state.dart';

import 'package:floradlite/models/products/ProductsList.dart';

import 'package:floradlite/services/product\_services.dart';

import 'package:floradlite/services/testService.dart';

import 'package:floradlite/widgets/custom\_popup\_menu.dart';

import 'package:floradlite/widgets/helper.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

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

import 'package:provider/provider.dart';

class TestVarient extends StatefulWidget{

  @override

  State<StatefulWidget> createState() {

    return TestVarientWidget();

      }

    }

    class TestVarientWidget extends State<TestVarient>{

      @override

      void initState(){

        super.initState();

     //   \_loadOrders();

      }

      // \_loadOrders(){

      //   context.read<LoginBloc>().add(GetVarients());

      // }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      body:BlocListener<LoginBloc,LoginState>(listener: (context,state){

if(state is LoginErrorState){

  return         Scaffold.of(context).showSnackBar(

                      SnackBar(

                          content: Text(state.error),

                        ));

}

},child:  BlocBuilder<LoginBloc,LoginState>(builder: (context,state){

if( state is LoginErrorState){

          return Container(height: 100,

            margin: EdgeInsets.only(top:100),

            child: Text('error',style: TextStyle(fontSize: 100),

            ),

          );

        }

   if(state is LoginLoaded){

    ProductsList productsList=state.productsList;

    return Container(height: 500,

    margin: EdgeInsets.only(top:200),

    child: list(productsList)

    );

}

return CircularProgressIndicator();

},

   ),

  ) );} }

        Widget list(ProductsList fp){

          return GridView.count(

              crossAxisCount: 2,

              mainAxisSpacing: 5,

              crossAxisSpacing: 5,

              padding: EdgeInsets.all(1.0),

              childAspectRatio: 8.0 / 12.0,

              children: List<Widget>.generate(

                  fp.success[0].variants.length, (index) {

                return Padding(

                  padding: const EdgeInsets.all(10.0),

                  child: GridTile(

                      child: GridTilesProducts(

                    name: 'rakesh',

                    imageUrl: fp.success[0].variants[index].images[index],

                    description:fp.success[0].variants[index].description,

                    // color: model.fp.success[index].variants[index].varianttype,

                    mrp: '10',

                    sellingPrice:'120',

                    // psize: model.fp.success[index].variants[index].availability,

                  )),

                );

              }),

            );

        }

        class GridTilesProducts extends StatelessWidget {

  String name;

  String imageUrl;

  String description;

  String color;

  String mrp;

  String sellingPrice;

  String psize;

  bool fromSubProducts = false;

  GridTilesProducts(

      {Key key,

      @required this.name,

      @required this.imageUrl,

      @required this.description,

      // @required this.color,

      @required this.mrp,

      @required this.sellingPrice,

      // @required this.psize,

      this.fromSubProducts})

      : super(key: key);

  @override

  Widget build(BuildContext context) {

    Size size = MediaQuery.of(context).size;

    return Container(

      child: Card(

          color: Colors.grey[200],

          shape: RoundedRectangleBorder(

            borderRadius: const BorderRadius.all(

              Radius.circular(10.0),

            ),

          ),

          elevation: 0,

          child: Center(

            child: Column(

              children: <Widget>[

                Stack(

                  children: [

                    Container(

                      width: double.infinity,

                      height: size.height \* 0.18,

                      decoration: BoxDecoration(),

                      child: ClipRRect(

                        borderRadius: BorderRadius.circular(8.0),

                        child: Image.network(

                          imageUrl,

                          fit: BoxFit.fill,

                        ),

                      ),

                    ),

                    Positioned(

                      right: 8.0,

                      top: 8.0,

                      child: new GestureDetector(

                        child: PopupMenuButton<CustomPopupMenu>(

                          elevation: 2.2,

                          child: popUpMenuIcon(),

                          initialValue: choices[0],

                          onCanceled: () {

                            print('You have not choosed anything');

                          },

                          tooltip: 'This is tooltip',

                          onSelected: (choice) {

                            if (choice.title == 'Edit') {

                              print("Clicked on edit item");

                            } else if (choice.title == 'Share') {

                            } else if (choice.title == 'Publish') {

                            } else if (choice.title == 'Delete') {

                            } else if (choice.title == 'Copy') {

                            } else if (choice.title == 'AddVarients') {

                              // Navigator.pushNamed(context, '/addproductvarient');

                            }

                          },

                          itemBuilder: (BuildContext context) {

                            return choices.map((CustomPopupMenu choice) {

                              return PopupMenuItem<CustomPopupMenu>(

                                value: choice,

                                child: new Row(children: <Widget>[

                                  new Icon(choice.icon),

                                  Padding(

                                    padding: const EdgeInsets.all(8.0),

                                    child: Text(choice.title),

                                  )

                                ]),

                              );

                            }).toList();

                          },

                        ),

                      ),

                    ),

                  ],

                ),

                Container(

                  alignment: Alignment.topLeft,

                  padding: EdgeInsets.only(left: 4, right: 4, top: 4),

                  child: headingText(name),

                ),

                Container(

                  alignment: Alignment.topLeft,

                  padding: EdgeInsets.only(left: 4, right: 4, top: 4),

                  child: subHeadingText(description),

                ),

                Row(

                  mainAxisAlignment: MainAxisAlignment.spaceBetween,

                  children: [

                    Container(

                      alignment: Alignment.topLeft,

                      padding: EdgeInsets.only(top: 6, left: 4.0),

                      child: multiColorText('Color: ', color),

                    ),

                    Container(

                      alignment: Alignment.topLeft,

                      padding: EdgeInsets.only(top: 4, right: 4.0),

                      child: lineThroughText(mrp),

                    ),

                  ],

                ),

                Row(

                  mainAxisAlignment: MainAxisAlignment.spaceBetween,

                  children: [

                    Container(

                      alignment: Alignment.topLeft,

                      padding: EdgeInsets.only(top: 4, left: 4.0),

                      child: multiColorText('Size: ', psize),

                    ),

                    Container(

                      alignment: Alignment.topLeft,

                      padding: EdgeInsets.only(top: 4, right: 4.0),

                      child: headingText(sellingPrice),

                    ),

                  ],

                ),

              ],

            ),

          )),

    );

  }

}

Creating Login Using Provider

Modelclass

import 'dart:convert';

LoginPost loginPostFromJson(String str) => LoginPost.fromJson(json.decode(str));

String loginPostToJson(LoginPost data) => json.encode(data.toJson());

class LoginPost {

    LoginPost({

        this.code,

        this.message,

        this.status,

        this.storeid,

        this.success,

    });

    int code;

    String message;

    String status;

    String storeid;

    Success success;

    factory LoginPost.fromJson(Map<String, dynamic> json) => LoginPost(

        code: json["code"],

        message: json["message"],

        status: json["status"],

        storeid: json["storeid"],

        success: Success.fromJson(json["success"]),

    );

    Map<String, dynamic> toJson() => {

        "code": code,

        "message": message,

        "status": status,

        "storeid": storeid,

        "success": success.toJson(),

    };

}

class Success {

    Success({

        this.accesskey,

        this.created,

        this.dateofbirth,

        this.employeeid,

        this.firstname,

        this.gender,

        this.lastmodified,

        this.lastname,

        this.ownerid,

        this.password,

        this.phonenumber,

        this.profilepic,

        this.profilepictype,

        this.requestfrom,

        this.role,

        this.status,

        this.storeid,

        this.useremail,

        this.verified,

    });

    String accesskey;

    String created;

    String dateofbirth;

    String employeeid;

    String firstname;

    String gender;

    String lastmodified;

    String lastname;

    String ownerid;

    String password;

    String phonenumber;

    String profilepic;

    String profilepictype;

    String requestfrom;

    String role;

    String status;

    String storeid;

    String useremail;

    String verified;

    factory Success.fromJson(Map<String, dynamic> json) => Success(

        accesskey: json["accesskey"],

        created: json["created"],

        dateofbirth: json["dateofbirth"],

        employeeid: json["employeeid"],

        firstname: json["firstname"],

        gender: json["gender"],

        lastmodified: json["lastmodified"],

        lastname: json["lastname"],

        ownerid: json["ownerid"],

        password: json["password"],

        phonenumber: json["phonenumber"],

        profilepic: json["profilepic"],

        profilepictype: json["profilepictype"],

        requestfrom: json["requestfrom"],

        role: json["role"],

        status: json["status"],

        storeid: json["storeid"],

        useremail: json["useremail"],

        verified: json["verified"],

    );

    Map<String, dynamic> toJson() => {

        "accesskey": accesskey,

        "created": created,

        "dateofbirth": dateofbirth,

        "employeeid": employeeid,

        "firstname": firstname,

        "gender": gender,

        "lastmodified": lastmodified,

        "lastname": lastname,

        "ownerid": ownerid,

        "password": password,

        "phonenumber": phonenumber,

        "profilepic": profilepic,

        "profilepictype": profilepictype,

        "requestfrom": requestfrom,

        "role": role,

        "status": status,

        "storeid": storeid,

        "useremail": useremail,

        "verified": verified,

    };

}

InsideLoginpage Ui

import 'dart:convert';

import 'package:floradlite/screens/auth/forgotPassword.dart';

import 'package:floradlite/services/login\_services.dart';

import 'package:floradlite/utils/button\_state.dart';

import 'package:flutter/cupertino.dart';

import 'package:flutter/material.dart';

import 'package:material\_design\_icons\_flutter/material\_design\_icons\_flutter.dart';

import 'package:provider/provider.dart';

import 'package:floradlite/common/common\_constants.dart' as constants;

import 'package:shared\_preferences/shared\_preferences.dart';

class LoginPage extends StatelessWidget {

  @override

  Widget build(BuildContext context) {

    return LoginWidget();

  }

}

class LoginWidget extends StatefulWidget {

  @override

  State<StatefulWidget> createState() {

    return LoginItem();

  }

}

class LoginItem extends State<LoginWidget> {

  bool passwordvisible = true;

  TextEditingController nameController = TextEditingController();

  TextEditingController pwdController = TextEditingController();

  Future userSignin(BuildContext context, String name, String pwd) async {

    SharedPreferences prefs = await SharedPreferences.getInstance();

    try {

      var body = jsonEncode({

        "requestfrom": constants.RequestFrom,

        "useremail": name,

        "password": pwd

      });

      final auth = Provider.of<LoginServices>(context, listen: false);

      final model = Provider.of<ButtonState>(context, listen: false);

      final response = await auth.signIn(body);

      print(response.status);

      prefs.setBool(constants.Login\_Status, true);

      prefs.setString(constants.Storeid, response.success.storeid);

      prefs.setString(constants.App\_Accesskey, response.success.accesskey);

      prefs.setString(constants.App\_Userid, response.success.employeeid);

      model.changeState();

      await Future.delayed(Duration(milliseconds: 200), () {

        Navigator.pushNamed(context, '/bottomNav');

        model.changeState();

      });

    } catch (e) {

      print(e);

    }

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      resizeToAvoidBottomInset: false,

      body: Column(

        crossAxisAlignment: CrossAxisAlignment.start,

        children: [

          SizedBox(height: MediaQuery.of(context).size.height \* 0.2),

          Container(

              margin: EdgeInsets.only(left: 40),

              child: Text(

                'Lets Sign you In ',

                style: TextStyle(fontSize: 24, fontWeight: FontWeight.w400),

              )),

          Container(

              margin: EdgeInsets.only(left: 40, top: 7),

              child: Text(

                'Welcome back, you\'ve been missed! ',

                style: TextStyle(

                  fontSize: 14,

                ),

              )),

          Padding(

            padding: const EdgeInsets.all(28.0),

            child: TextFormField(

              controller: nameController,

              decoration: InputDecoration(

                  prefixIcon: Icon(Icons.person\_outline\_outlined),

                  labelText: 'Email'),

            ),

          ),

          Padding(

            padding: const EdgeInsets.only(left: 28, right: 28, top: 0),

            child: TextFormField(

              controller: pwdController,

              obscureText: passwordvisible,

              decoration: InputDecoration(

                  prefixIcon: Icon(Icons.lock\_outlined),

                  labelText: 'Password',

                  suffixIcon: GestureDetector(

                    child: Icon(passwordvisible

                        ? Icons.remove\_red\_eye\_outlined

                        : MdiIcons.eyeOffOutline),

                    onTap: () {

                      setState(() {

                        passwordvisible = !passwordvisible;

                      });

                    },

                  )),

            ),

          ),

          Container(

              margin: EdgeInsets.only(top: 19, right: 22),

              alignment: Alignment.bottomRight,

              child: GestureDetector(

                onTap: () {

                  Navigator.push(

                      context,

                      new MaterialPageRoute(

                          builder: (context) => ForgotpassWord()));

                },

                child: Text(

                  'Forgot Password',

                  textAlign: TextAlign.center,

                  style: TextStyle(

                    fontSize: 15,

                    letterSpacing: 0.5,

                  ),

                ),

              )),

          SizedBox(

            height: MediaQuery.of(context).size.height \* 0.10,

          ),

          Consumer<ButtonState>(

            builder: (\_, model, child) => Container(

                alignment: Alignment.center,

                child: FlatButton(

                  height: MediaQuery.of(context).size.height \* 0.078,

                  minWidth: MediaQuery.of(context).size.width \* 0.8,

                  color: Colors.black,

                  onPressed: () async {

                    await userSignin(

                        context, nameController.text, pwdController.text);

                    // model.changeState();

                    // await Future.delayed(Duration(milliseconds: 200), () {

                    //   Navigator.pushNamed(context, '/bottomNav');

                    //   model.changeState();

                    // });

                  },

                  child: Text(

                    model.status ? 'Signing in...' : 'Sign In',

                    style: TextStyle(

                      color: Colors.white,

                    ),

                  ),

                  shape: RoundedRectangleBorder(

                    borderRadius: BorderRadius.circular(10),

                  ),

                )),

          ),

          SizedBox(

            height: MediaQuery.of(context).size.height \* 0.05,

          ),

          GestureDetector(

            child: Row(

              mainAxisAlignment: MainAxisAlignment.center,

              children: [

                Text('Don\'t have an Account?'),

                Text(

                  'Sign Up ',

                  style: TextStyle(fontWeight: FontWeight.w700),

                ),

              ],

            ),

            onTap: () {},

          )

        ],

      ),

    );

  }

}

Api call

import 'dart:convert';

import 'package:floradlite/models/login/LoginPost.dart';

import 'package:floradlite/utils/my\_exception.dart';

import 'package:http/http.dart' as http;

import 'package:floradlite/common/common\_constants.dart' as constants;

// class UserLogin{

//   final String userId,password,requestfrom;

//   const UserLogin({@required this.userId,@required this.password, this.requestfrom});

// }

class LoginServices{

  LoginPost post;

  Future<LoginPost> signIn(body) async {

    try{

    final authResult = await http.post(constants.Login\_Url,body: body);

    print("authResult: "+ authResult.body+","+constants.Login\_Url);

    return LoginPost.fromJson(json.decode(authResult.body));

    }on CustomException catch (e, st) {

      print(st);

    }

  }

}

Bloc listenwhen option

|  |
| --- |
| return BlocConsumer<RegionalReportBloc, RegionalReportState>( |
|  |
|  | listenWhen: (previous, current) => current is RegionalReportLoadingError, |
|  | listener: (context, state) { |
|  | if (state is RegionalReportLoadingError) { |
|  | Scaffold.of(context).showSnackBar( |
|  | SnackBar( |
|  | content: Text( |
|  | tr('error', namedArgs: {'reason': state.reason}), |
|  | ), |
|  | ), |
|  | ); |

Counter app using bloc and Cubit

  flutter\_bloc: ^8.0.1

  equatable: ^2.0.3

  bloc: ^8.0.2

Main page and Ui :

import 'package:flutter/material.dart';

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

import 'package:rakesh\_test/counter\_bloc/counter\_state.dart';

import 'package:rakesh\_test/counter\_bloc/counterbloc.dart';

void main() {

  runApp(const MyApp());

}

final \_messangerKey = GlobalKey<ScaffoldMessengerState>();

class MyApp extends StatelessWidget {

  const MyApp({Key? key}) : super(key: key);

  @override

  Widget build(BuildContext context) {

    return BlocProvider<CounterCubit>(

      create: (context) => CounterCubit(),

      child: MaterialApp(

        scaffoldMessengerKey:\_messangerKey ,

        title: 'Flutter Demo',

        theme: ThemeData(

          primarySwatch: Colors.blue,

        ),

        home: const MyHomePage(title: 'Flutter Demo Home Page'),

      ),

    );

  }

}

class MyHomePage extends StatefulWidget {

  const MyHomePage({Key? key, required this.title}) : super(key: key);

  final String title;

  @override

  State<MyHomePage> createState() => \_MyHomePageState();

}

class \_MyHomePageState extends State<MyHomePage> {

  int \_counter = 2;

  void \_incrementCounter() {

    setState(() {

      \_counter++;

    });

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(

        title: Text(widget.title),

      ),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: <Widget>[

            const Text(

              'You have pushed the button this many times:',

            ),

            BlocListener<CounterCubit, CounterState>(

              listener: (context, state) {

                if(state.wasIncremented == true)

                {

                 \_messangerKey.currentState?.showSnackBar(

                    const SnackBar(content:Text('Incremented'),

                    duration: Duration(milliseconds: 300),

                    ),

                  );

                }

                else if (state.wasIncremented == false)

                {

                 \_messangerKey.currentState?.showSnackBar(

                  const  SnackBar(

                    content: Text('Decremented'),

                    duration: Duration(milliseconds: 300),

                  ),

                  );

                }

              },

              child: BlocBuilder<CounterCubit, CounterState>(

                builder: (context, state) {

                  return Text(

                    state.counterValue.toString(),

                    style: Theme.of(context).textTheme.headline4,

                  );

                },

              ),

            ),

            Row(

              mainAxisAlignment: MainAxisAlignment.spaceEvenly,

              children: [

                FloatingActionButton(

                  onPressed: () {

                    //access the counter cubit

                    BlocProvider.of<CounterCubit>(context).decrement();

                    //context.bloc<CounterCubit>().decrement();

                  },

                  tooltip: 'Decrement',

                  child:const Icon(Icons.remove),

                ),

                FloatingActionButton(

                  onPressed: () {

                    BlocProvider.of<CounterCubit>(context).increment(\_counter);

                   // context.bloc<CounterCubit>().increment();

                  },

                  tooltip: 'Increment',

                  child:const Icon(Icons.add),

                ),

              ],

            )

          ],

        ),

      ),

    );

  }

}

Bloc class:

import 'counter\_state.dart';

import 'package:bloc/bloc.dart';

class CounterCubit extends Cubit<CounterState> {

  CounterCubit() : super(CounterState(counterValue: 0, wasIncremented: false));

  void increment(int counter) {

    emit(CounterState(counterValue: state.counterValue + counter, wasIncremented:true));

  }

  void decrement() => emit(CounterState(counterValue: state.counterValue - 1, wasIncremented: false));

}

Counter State:

class CounterState {

  int counterValue;

  bool wasIncremented;

  CounterState({

    required this.counterValue,

    required this.wasIncremented,

  });

}

Flutter\_bloc with emitter

Event.dart:

import 'package:equatable/equatable.dart';

abstract class CampainEvent extends Equatable{

}

class Fetchcampains extends CampainEvent{

  @override

  List<Object?> get props => [];

}

s

State.dart:

import 'package:dynatrade\_enquiry\_app/models/campaign\_model.dart';

import 'package:equatable/equatable.dart';

abstract class CampainState extends Equatable{

}

class CampainInitState extends CampainState{

  @override

  List<Object?> get props =>[];

}

class CampainLoadingState extends CampainState{

  @override

  List<Object?> get props => [];

}

class CampainLodedState extends CampainState{

 final List<OfferDetail>campinList;

 CampainLodedState({required this.campinList});

  @override

  List<Object?> get props => [];

}

class CampainErrorList extends CampainState{

  final errormessege;

  CampainErrorList({this.errormessege});

  @override

  List<Object?> get props => [];

}

Bloc.dart:

import 'dart:async';

import 'dart:io';

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

import 'package:dynatrade\_enquiry\_app/bloc/campaignbloc/camp\_event.dart';

import 'package:dynatrade\_enquiry\_app/bloc/campaignbloc/camp\_state.dart';

import 'package:dynatrade\_enquiry\_app/bloc/campaignbloc/campain\_repository.dart';

import 'package:dynatrade\_enquiry\_app/models/campaign\_model.dart';

class Campainbloc extends Bloc<CampainEvent, CampainState> {

  List<OfferDetail> campainList = [];

  final CampainRepository repository;

  Campainbloc({required this.repository}) : super(CampainInitState()) {

    on<CampainEvent>(\_onCampainEvent);

  }

// @override

// Stream<CampainState>mapEventToState(CampainEvent campainEvent)async\* {

// if(campainEvent is Fetchcampains){

//   yield CampainLoadingState();

//   try{

//     campainList=await repository.fetchcampaings();

//   yield CampainLodedState(campinList: campainList);

//   } on SocketException{

//   yield CampainErrorList(errormessege: "No Internet exeption");

//   }

//   catch(e){

//     yield CampainErrorList(errormessege: "Unknown Exeption");

//   }

// }

// }

  FutureOr<void> \_onCampainEvent(

      CampainEvent event, Emitter<CampainState> emit) async {

    emit(CampainLoadingState());

    try {

      campainList = await repository.fetchcampaings();

      emit(CampainLodedState(campinList: campainList));

    } catch (e) {

      emit(CampainErrorList(

          errormessege: UnkwonExeption(messege: "Unknown exception")));

    }

  }

}

class UnkwonExeption {

  var messege;

  UnkwonExeption({this.messege});

}

Repository.dart:

import 'package:dynatrade\_enquiry\_app/models/campaign\_model.dart';

abstract class CampainRepository{

  Future<List<OfferDetail>>fetchcampaings();

}

Apicall.dart:

import 'package:dynatrade\_enquiry\_app/bloc/campaignbloc/campain\_repository.dart';

import 'package:dynatrade\_enquiry\_app/models/campaign\_model.dart'as camp;

import 'dart:convert';

import 'dart:io';

import 'package:dynatrade\_enquiry\_app/common/common\_constants.dart'

    as constants;

import 'package:fluttertoast/fluttertoast.dart';

import 'package:http/http.dart' as http;

import 'package:shared\_preferences/shared\_preferences.dart';

class FetchCampinApi implements CampainRepository{

  List<camp.OfferDetail> campaignList = [];

  @override

  Future<List<camp.OfferDetail>> fetchcampaings() async{

 try{

   print('enter dragon');

     String campainUrl = constants.Campain\_Url;

      SharedPreferences pref = await SharedPreferences.getInstance();

      var key = pref.getString(constants.AcceesKey);

      var headers = {

        'content-Type': 'application/json',

        'Authorization': 'Bearer $key',

      };

      final campaindetails = await http.post(

        Uri.parse(campainUrl),

        headers: headers,

      );

      if (campaindetails.statusCode == 200) {

        print('status code ok in camp');

        var rest = jsonDecode(campaindetails.body);

        var data = rest["offer\_details"];

        var code = rest["response\_code"];

        if (code.toString() == "404") {

          Fluttertoast.showToast(

              msg: 'Campaign/Promotion details not found',

              toastLength: Toast.LENGTH\_LONG);

          return campaignList;

        }

        campaignList = data

            .map<camp.OfferDetail>((json) => camp.OfferDetail.fromJson(json))

            .toList();

      } else if(campaindetails.statusCode==401){

        Fluttertoast.showToast(msg: "Session Expired Relogin Please");

      }else if(campaindetails.statusCode==400){

      }else if(campaindetails.statusCode==500){

        Fluttertoast.showToast(msg: "Internal Server Error");

      }

    } on SocketException{

      Fluttertoast.showToast(msg: "No Internet Connection");

    }

    catch (e) {

      print(e.toString());

    }

    return campaignList;

  }

}

Main.dart:

child: MultiBlocProvider(providers: [

            BlocProvider<TabCube>(

      create: (context) => TabCube(),

            ),

            BlocProvider<Campainbloc>(create: (\_)=>Campainbloc(repository: FetchCampinApi(),))],

            child: MyApp())))

Ui.dart

import 'package:dynatrade\_enquiry\_app/bloc/campaignbloc/camp\_bloc.dart';

import 'package:dynatrade\_enquiry\_app/bloc/campaignbloc/camp\_event.dart';

import 'package:dynatrade\_enquiry\_app/bloc/campaignbloc/camp\_state.dart';

import 'package:dynatrade\_enquiry\_app/landing\_page/promotions/pdf\_veiwer.dart';

import 'package:dynatrade\_enquiry\_app/models/Campaign\_model.dart';

import 'package:dynatrade\_enquiry\_app/models/campaign\_model.dart' as camp;

import 'package:dynatrade\_enquiry\_app/models/partsearchmodel.dart';

import 'package:dynatrade\_enquiry\_app/reusable\_widgets/side\_navigation.dart';

import 'package:dynatrade\_enquiry\_app/services/campaignservice.dart';

import 'package:flutter/material.dart';

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

import 'package:google\_fonts/google\_fonts.dart';

import 'package:multi\_select\_flutter/chip\_field/multi\_select\_chip\_field.dart';

import 'package:multi\_select\_flutter/util/multi\_select\_item.dart';

import 'package:provider/provider.dart';

import 'package:smooth\_page\_indicator/smooth\_page\_indicator.dart';

class Capmains extends StatefulWidget {

  @override

  State<StatefulWidget> createState() {

    return CampainsWidget();

  }

}

class CampainsWidget extends State<Capmains> {

  static List<Partsearch> brandnames = [

    Partsearch(

      brandName: 'ALL',

    ),

    Partsearch(

      brandName: 'UPCOMING',

    ),

    Partsearch(

      brandName: 'ONGOING',

    ),

  ];

  bool itemok = false;

  List<camp.OfferDetail> filterdlist = [];

  final pagecontroller = PageController(viewportFraction: 1, keepPage: true);

  final branditems = brandnames

      .map((animal) => MultiSelectItem<Partsearch>(animal, animal.brandName))

      .toList();

  @override

  void initState() {

    super.initState();

        \_loadOrders();

  //  Provider.of<CampaignService>(context, listen: false).getcampain(context);

    // BlocProvider.of<Campainbloc>(context).add(Fetchcampains());

  }

    Future \_loadOrders() async{

    context.read<Campainbloc>().add(Fetchcampains());

  }

  @override

  Widget build(BuildContext context) {

    Size size = MediaQuery.of(context).size;

    return Scaffold(

      drawer: MyDrawer(),

      appBar: AppBar(

          backgroundColor: Color(0xff014EA2),

          title: Text('Promotions & Campaigns',

              style: GoogleFonts.montserrat(

                color: Colors.white,

                fontSize: size.height \* 0.02,

              ))),

      body: Container(

        child:

            BlocListener<Campainbloc, CampainState>(listener: (context, state) {

          if (state is CampainErrorList) {

            Scaffold.of(context).showSnackBar(SnackBar(

              content: Text(state.errormessege),

            ));

          }

        },child:BlocBuilder<Campainbloc, CampainState>(builder: (context,state){

          if(state is CampainInitState){

            return CircularProgressIndicator();

          }else if(state is CampainLoadingState){

            return CircularProgressIndicator();

          }

          else if(state is CampainLodedState){

            return builscampainsUi( context, state.campinList);

          }

              return buildErrorUi();

        },

        ) ,

        ),

      ),

    );

  }

    Widget buildErrorUi() {

    return Center(

      child: Padding(

        padding: const EdgeInsets.all(8.0),

        child: Text(

          '',

          style: TextStyle(color: Colors.red),

        ),

      ),

    );

  }

  Widget builscampainsUi(BuildContext context, List<camp.OfferDetail>campainList){

    Size size=MediaQuery.of(context).size;

        return Column(children: [

            Wrap(children: [

              MultiSelectChipField<Partsearch?>(

                initialValue: [brandnames[0]],

                  items: brandnames

                      .map((animal) => MultiSelectItem<Partsearch>(

                          animal, animal.brandName))

                      .toList(),

                  showHeader: false,

                  title: Text('Promotions & Campaigns'),

                  headerColor: Color(0xffF0F8FF),

                  textStyle: TextStyle(

                      color: Colors.black, fontSize: size.height \* 0.018),

                  decoration: BoxDecoration(color: Color(0xffF0F8FF)),

                  selectedChipColor: Color(0xff014EA2),

                  selectedTextStyle: TextStyle(color: Colors.white,),

                  onTap: (values) {

                    if(values.length>1)

                    if(values[1]!.brandName.contains("ALL")){

                      setState(() {

                        itemok=false;

                        values.clear();

                        values.add(Partsearch(brandName: "ALL"));

                      });

                    }

                     if(values.length>2)

                    if(values[2]!.brandName.contains("ALL")){

                      setState(() {

                      values.clear();

                      itemok=false;

                        values.add(Partsearch(brandName: "ALL"));

                      });

                       if(values.length==0)

                    if(values[0]!.brandName.contains("ALL")){

                      setState(() {

                        itemok=false;

                      values.clear();

                        values.add(Partsearch(brandName: "ALL"));

                      });

                    }}

                    if(values.length>0)

                    if(values[0]!.brandName.contains("UPCOMING")||values[0]!.brandName.contains("ONGOING")){

                      setState(() {

                        itemok=true;

                 values.removeWhere((element) => element!.brandName=="ALL");

                      });

                    }

                    if(values.length>1)

                    if(values[1]!.brandName.contains("UPCOMING")||values[1]!.brandName.contains("ONGOING")){

                      setState(() {

                        itemok=true;

                      values.removeWhere((element) => element!.brandName=="ALL");

                      });

                    }

                    if(values.length>2)

                    if(values[2]!.brandName.contains("UPCOMING")||values[2]!.brandName.contains("ONGOING")){

                      setState(() {

                        itemok=true;

                         values.removeWhere((element) => element!.brandName=="ALL");

                      });

                    }

                    filterdlist.clear();

                    for (var i = 0; i < values.length; i++) {

                      campainList.forEach((element) {

                        setState(() {

                          if (element.offerStatus

                              .toUpperCase()

                              .contains(values[i]!.brandName))

                            filterdlist.add(element);

                        });

                        print("filter lenght is"+filterdlist.length.toString());

                      });

                    }

                  }),

            ]),

            SizedBox(

              height: size.height \* 0.25,

              child: PageView(

                controller: pagecontroller,

                // itemCount: pages.length,

                children: [

                  Column(

                    children: [

                      Container(

                          width: size.width,

                          height: size.height \* 0.23,

                          child: Image.asset('assets/lights.png'))

                    ],

                  ),

                  //Second Page

                  Row(

                    children: [

                      Container(

                        width: size.width \* 0.55,

                        height: size.height \* 0.32,

                        decoration: BoxDecoration(),

                        child: Image.asset('assets/screwpartfinal.png'),

                      ),

                      Container(

                        margin: EdgeInsets.only(

                            top: size.height \* 0.02, left: size.width \* 0.06),

                        width: size.width \* 0.3,

                        height: size.height \* 0.3,

                        decoration: BoxDecoration(),

                        child: Image.asset('assets/partthree2.png'),

                      ),

                    ],

                  ),

                ],

              ),

            ),

            Container(

              child: SmoothPageIndicator(

                controller: pagecontroller,

                count: 2,

                effect: ExpandingDotsEffect(

                    dotHeight: size.height \* 0.015,

                    dotWidth: size.height \* 0.017,

                    activeDotColor: Colors.black,

                    dotColor: Color(0xffECEBED)),

              ),

            ),

            Expanded(

                child:filterdlist.length<=0&&!itemok?

                 ListView.builder(

                    itemCount: campainList.length,

                    itemBuilder: (context, index) {

                      return Card(

                        elevation: 02,

                        child: Column(

                          crossAxisAlignment: CrossAxisAlignment.start,

                          children: [

                            Container(

                              margin: EdgeInsets.only(

                                  left: size.width \* 0.05,

                                  top: size.height \* 0.01),

                              child: Text(

                                  '${campainList[index].offerName}',

                                  style: GoogleFonts.dmSans(

                                      fontSize: size.height \* 0.016,

                                      color: Color(0xff42526E))),

                            ),

                            SizedBox(

                              height: size.height \* 0.04,

                            ),

                            Row(

                              mainAxisAlignment:

                                  MainAxisAlignment.spaceEvenly,

                              children: [

                                Container(

                                  child: Text(

                                      "Start :" +

                                          "${campainList[index].startDate}",

                                      style: GoogleFonts.dmSans(

                                          fontSize: size.height \* 0.015,

                                          color: Color(0xff42526E))),

                                ),

                                Container(

                                  child: Text(

                                      "Expires :" +

                                          "${campainList[index].endDate}",

                                      style: GoogleFonts.dmSans(

                                          fontSize: size.height \* 0.015,

                                          color: Color(0xff42526E))),

                                ),

                                GestureDetector(

                                  onTap: () {

                                    Navigator.push(

                                        context,

                                        MaterialPageRoute<dynamic>(

                                          builder: (\_) => PDFViewerFromUrl(

                                            url: campainList[index]

                                                    .filePath +

                                               campainList[index]

                                                    .fileName,

                                          ),

                                        ));

                                  },

                                  child: Container(

                                    child: Text("View Details",

                                        style: GoogleFonts.dmSans(

                                            fontSize: size.height \* 0.017,

                                            color: Color(0xff42526E),

                                            fontWeight: FontWeight.bold)),

                                  ),

                                )

                              ],

                            ),

                            SizedBox(

                              height: size.height \* 0.01,

                            )

                          ],

                        ),

                      );

                    }):

                    //FilteredList

                    ListView.builder(

                    itemCount: filterdlist.length,

                    itemBuilder: (context, index) {

                      return Card(

                        elevation: 02,

                        child: Column(

                          crossAxisAlignment: CrossAxisAlignment.start,

                          children: [

                            Container(

                              margin: EdgeInsets.only(

                                  left: size.width \* 0.05,

                                  top: size.height \* 0.01),

                              child: Text(

                                  '${filterdlist[index].offerName}',

                                  style: GoogleFonts.dmSans(

                                      fontSize: size.height \* 0.016,

                                      color: Color(0xff42526E))),

                            ),

                            SizedBox(

                              height: size.height \* 0.04,

                            ),

                            Row(

                              mainAxisAlignment:

                                  MainAxisAlignment.spaceEvenly,

                              children: [

                                Container(

                                  child: Text(

                                      "Start :" +

                                          "${filterdlist[index].startDate}",

                                      style: GoogleFonts.dmSans(

                                          fontSize: size.height \* 0.015,

                                          color: Color(0xff42526E))),

                                ),

                                Container(

                                  child: Text(

                                      "Expires :" +

                                          "${filterdlist[index].endDate}",

                                      style: GoogleFonts.dmSans(

                                          fontSize: size.height \* 0.015,

                                          color: Color(0xff42526E))),

                                ),

                                GestureDetector(

                                  onTap: () {

                                    Navigator.push(

                                        context,

                                        MaterialPageRoute<dynamic>(

                                          builder: (\_) => PDFViewerFromUrl(

                                            url: filterdlist[index]

                                                    .filePath +

                                                filterdlist[index]

                                                    .fileName,

                                          ),

                                        ));

                                  },

                                  child: Container(

                                    child: Text("View Details",

                                        style: GoogleFonts.dmSans(

                                            fontSize: size.height \* 0.017,

                                            color: Color(0xff42526E),

                                            fontWeight: FontWeight.bold)),

                                  ),

                                )

                              ],

                            ),

                            SizedBox(

                              height: size.height \* 0.01,

                            )

                          ],

                        ));

        }))]);

  }

}

Bloc with Extra Parameter

event

import 'package:equatable/equatable.dart';

import '../models/drafts/OfferDetails.dart';

abstract class FetchOfferEvent extends Equatable{

  @override

  List<Object> get progs => [];

}

class InitFetch extends FetchOfferEvent{

  final OfferDetails ? offerDetails;

InitFetch({this.offerDetails});

  @override

  List<Object?> get props => [offerDetails];

}

State

import 'package:equatable/equatable.dart';

import '../models/drafts/OfferDetails.dart';

abstract class EditOfferState extends Equatable{

  @override

  List<Object?> get props => [];

}

class EditInitState extends EditOfferState{

  @override

  List<Object> get props => [];

}

class EditOfferLoadingState extends EditOfferState{

    @override

  List<Object> get props => [];

}

class EditOfferLoadedState extends EditOfferState{

final OfferDetails ? offerDetails;

final String? id;

EditOfferLoadedState({this.offerDetails,this.id});

 @override

  List<Object> get props => [];

}

class EditOfferErrorState extends EditOfferState{

    final error;

  EditOfferErrorState({this.error});

  @override

  List<Object> get props =>[];

}

Repository

import 'dart:convert';

import 'package:shared\_preferences/shared\_preferences.dart';

import '../models/drafts/OfferDetails.dart';

import 'package:offerji/common/common\_constants.dart' as constants;

import 'package:http/http.dart' as http;

abstract class SignInRepository {

  Future<OfferDetails?> fetchVariants();

  String ?id;

}

class FetchApi implements SignInRepository {

  OfferDetails? fp;

  String? id;

  @override

  Future<OfferDetails?> fetchVariants() async {

    try {

      SharedPreferences prefs = await SharedPreferences.getInstance();

      var headers = {

        "Accept": "application/json",

        "Authorization": "key=${prefs.getString(constants.App\_Accesskey)}"

      };

      String url = constants.FetchOfferDetails\_Url +

          "?requestfrom=" +

          constants.RequestFrom +

          "&accessuserid=" +

          prefs.getString(constants.App\_Userid).toString() +

          "&accesskey=" +

          prefs.getString(constants.App\_Accesskey).toString() +

          "&storeid=" +

          prefs.getString(constants.Storeid).toString() +

          "&offerid=" +

          constants.currOfferId;

      final drafts = await http.get(Uri.parse(url), headers: headers);

      print("variants: " + drafts.body + url + headers.toString());

      if (drafts.statusCode == 200) {

        fp = OfferDetails.fromJson(json.decode(drafts.body));

        id="123";

      }

    } on Exception catch (e) {

      print(e);

    }

    return fp;

  }

}

Bloc

import 'dart:async';

import 'dart:io';

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

import 'package:offerji/bloc/event.dart';

import 'package:offerji/bloc/repository.dart';

import 'package:offerji/bloc/state.dart';

import 'package:offerji/models/drafts/OfferDetails.dart';

class FetchBloc extends Bloc<FetchOfferEvent,EditOfferState> {

    OfferDetails? offerDetails;

    String? id;

  final SignInRepository repository;

  FetchBloc({required this.repository}) : super(EditInitState()) {

    on<FetchOfferEvent>(\_onCampainEvent);

  }

  // @override

  // Stream<EditOfferState> mapEventToState(FetchOfferEvent event) async\* {

  //  if (event is InitFetch) {

  //   yield EditOfferLoadingState();

  //   try {

  //    offerlist=await signInRepository.fetchVariants();

  //    yield EditOfferLoadedState(offerDetails: offerlist,);

  //   } on SocketException {

  //    yield EditOfferErrorState(error: NoInterNetExeption(messege:"No internet"));

  //        }

  //        catch (e) {

  //         yield EditOfferErrorState(error: UnkwonExeption(messege: "Uknownexeption"));

  //        }

  //   }

    FutureOr<void> \_onCampainEvent(

      FetchOfferEvent event, Emitter<EditOfferState> emit) async {

    emit(EditOfferLoadingState());

    try {

      offerDetails = await repository.fetchVariants();

      id=await repository.id;

      emit(EditOfferLoadedState(offerDetails: offerDetails,id: id));

    } catch (e) {

      emit(EditOfferErrorState(

          error: UnkwonExeption(messege: "Unknown exception")));

    }

  }

}

 class NoInterNetExeption {

 var messege;

 NoInterNetExeption({this.messege});

 }

 class UnkwonExeption{

 var messege;

 UnkwonExeption({this.messege});

 }

 class HttpExeption{

  var messege;

  HttpExeption({this.messege});

}

Ui

 @override

  void initState() {

    super.initState();

    setState(() {});

    context.read<FetchBloc>().add(InitFetch());

  }

Container(

        color: Colors.white,

        child: SingleChildScrollView(

            child: BlocListener<FetchBloc, EditOfferState>(

          listener: (context, state) {

            if (state is EditOfferErrorState) {

              Fluttertoast.showToast(msg: "Error Occured is" + state.error);

            }

            else if(state is EditOfferLoadedState){

              print("id value is"+state.id.toString());

              etofferNameController.text=state.offerDetails!.success![0].offertitle.toString();

              etofferTypeController.text=state.offerDetails!.success![0].offertype.toString();

              etofferDescController.text=state.offerDetails!.success![0].offerdescription.toString();

              etstartDateController.text=state.offerDetails!.success![0].offerstartdate.toString().substring(0,10);

              etendDateController.text=state.offerDetails!.success![0].offerenddate.toString().substring(0,10);

            }

          },

          child: BlocBuilder<FetchBloc, EditOfferState>(

              builder: ((context, state) {

            if (state is EditOfferErrorState) {

            return Center(child: Container(child: Text(state.error)),);

            }

            else if(state is EditOfferLoadingState){

           return Center(

                child: ColorLoader(

              dotIcon: Icon(Icons.adjust),

            ));

            }else if (state is EditOfferLoadedState){

              offerDetailsUi=state.offerDetails;

              return buildWidget(offerDetailsUi);

            }

          else{

            return Container(child: Text("none of above"),);

          }

          })),

        )));

  }

Main.dart

 runApp(EasyLocalization(

    supportedLocales: [

      Locale('en'),

      Locale('ml'),

      Locale('te'),

      Locale('ta'),

      Locale('kn'),

//      Locale('ar', 'DZ'),

    ],

    path: 'assets/translations',

    useOnlyLangCode: true,

    fallbackLocale:  Locale('en'),

    child: ProviderScope(child:

     MultiBlocProvider(

       providers: [

       BlocProvider(create:(\_)=>FetchBloc(repository: FetchApi()))

       ],

       child: MyApp())),

//    child: DevicePreview(

//      builder: (context) => MyApp(),

//    )

  ));

}

Bloc Testing with basic Counter Without State Class

Step1:

Add this pluggin: bloc\_test: ^9.0.3

Step2:

Create cubit class

import 'package:bloc\_tes/counter\_state.dart';

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

class CounterCubit extends Cubit<int>{

  CounterCubit() : super(0);

  void increment()=>emit(state+1);

  void decrement()=>emit(state-1);

}

Step3:

Create test cubit in test folder

import 'package:bloc\_tes/counter\_cubit.dart';

import 'package:bloc\_tes/counter\_state.dart';

import 'package:bloc\_test/bloc\_test.dart';

import 'package:flutter/material.dart';

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

import 'package:bloc\_tes/main.dart';

void main() {

  mainCubit();

}

class MockCounterCubit extends MockCubit<int> implements CounterCubit {}

void mainCubit() {

  group('whenListen', () {

    test("Let's mock the CounterCubit's stream!", () {

      // Create Mock CounterCubit Instance

      final bloc = MockCounterCubit();

      whenListen(bloc, Stream.fromIterable([0, 1, 2, 3]));

      expectLater(bloc.stream, emitsInOrder(<int>[0, 1, 2, 3]));

      // Stub the listen with a fake Stream

    });

  });

  group('CounterCubit', () {

    blocTest<CounterCubit, int>(

      'emits [] when nothing is called',

      build: () => CounterCubit(),

      expect: () => const <int>[],

    );

    blocTest<CounterCubit, int>(

      'emits [1] when increment is called',

      build: () => CounterCubit(),

      act: (cubit) => cubit.increment(),

      expect: () => const <int>[1],

    );

    blocTest<CounterCubit, int>(

      'emits [-1] when increment is called',

      build: () => CounterCubit(),

      act: (cubit) => cubit.decrement(),

      expect: () {

        return const <int>[-1];

      },

    );

  });

}

Bloc Testing with basic Counter With State Class

Step1:Add cubit class

import 'package:bloc\_tes/counter\_state.dart';

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

class CounterCubit extends Cubit<CounterState>{

  CounterCubit() : super(CounterState(counterValue: 0));

  void increment()=>emit(CounterState(counterValue: state.counterValue!+1));

  void decrement()=>emit(CounterState(counterValue: state.counterValue!-1));

}

Step2:

Add state class

import 'package:equatable/equatable.dart';

class CounterState extends Equatable{

  int? counterValue;

  CounterState({this.counterValue});

  @override

  List<Object?> get props => [counterValue];

}

Step3:Add test cubit

// This is a basic Flutter widget test.

//

// To perform an interaction with a widget in your test, use the WidgetTester

// utility that Flutter provides. For example, you can send tap and scroll

// gestures. You can also use WidgetTester to find child widgets in the widget

// tree, read text, and verify that the values of widget properties are correct.

import 'package:bloc\_tes/counter\_cubit.dart';

import 'package:bloc\_tes/counter\_state.dart';

import 'package:bloc\_test/bloc\_test.dart';

import 'package:flutter/material.dart';

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

import 'package:bloc\_tes/main.dart';

void main() {

  mainCubit();

}

class MockCounterCubit extends MockCubit<CounterState> implements CounterCubit {

}

void mainCubit() {

  group('whenListen', () {

    test("Let's mock the CounterCubit's stream!", () {

      // Create Mock CounterCubit Instance

      final bloc = MockCounterCubit();

      whenListen(

          bloc,

          Stream.fromIterable([

            CounterState(counterValue: 0),

            CounterState(counterValue: 1),

            CounterState(counterValue: 2)

          ]),

          initialState: CounterState(counterValue: 0));

      expect(bloc.state, equals(CounterState(counterValue: 0)));

      expectLater(

          bloc.stream,

          emitsInOrder(<CounterState>[

            CounterState(counterValue: 0),

            CounterState(counterValue: 1),

            CounterState(counterValue: 2)

          ]));

      //Stub the listen with a fake Stream

    });

  });

  group('CounterCubit', () {

    blocTest<CounterCubit, CounterState>(

      'emits [] when nothing is called',

      build: () {

        return CounterCubit();

      },

      act: (cubit) => cubit.decrement(),

      expect: () => [CounterState(counterValue: -1)],

    );

    blocTest<CounterCubit, CounterState>(

      'emits [1] when increment is called',

      build: () => CounterCubit(),

      act: (cubit) => cubit.increment(),

      expect: () => [CounterState(counterValue: 1)],

    );

  });

}

Drop down with bloc(different blocs with same widget)

 BlocListener<FetchCurrencyBloc, FetchCurrencyState>(

                      listener: (context, state) {},

                      child: BlocListener<FetchStoreBloc, FetchStoreState>(

                          listener: (context, state) {

                            print(state);

                            if (state is FetchStoreLoadedState) {

                              currencyDetection =

                                  state.userStore!.success[0].storecurrency;

                            }

                          },

                          child: BlocListener<SavedCubit, SavedCurrencyState>(

                            listener: (context, state) {

                              currencyDetection = state.storedCurrency;

                            },

                            child: BlocBuilder<FetchCurrencyBloc,

                                FetchCurrencyState>(builder: (context, state) {

                              if (state is FetchCurrencyLoadedState) {

                                return Padding(

                                  padding:

                                      EdgeInsets.fromLTRB(5.0, 0.0, 5.0, 0.0),

                                  child: Container(

                                    padding:

                                        EdgeInsets.symmetric(horizontal: 10.0),

                                    decoration: BoxDecoration(

                                      borderRadius:

                                          BorderRadius.all(Radius.circular(4)),

                                      border: Border.all(

                                          color: Theme.Colors.logoDarkBlue,

                                          style: BorderStyle.solid,

                                          width: 0.80),

                                    ),

                                    child: Padding(

                                      padding: EdgeInsets.all(5.0),

                                      child: DropdownButton(

                                          isExpanded: true,

                                          items:

                                              state.offerDetails!.map((item) {

                                            return new DropdownMenuItem(

                                              child: new Text(item),

                                              value: item.toString(),

                                            );

                                          }).toList(),

                                          onChanged: (newVal) {

                                            BlocProvider.of<SavedCubit>(

                                              context,

                                            ).add(UpdateStoreValueEvent(

                                                inputValue: newVal.toString()));

                                          },

                                          hint: Text('Select currency'),

                                          value: BlocProvider.of<SavedCubit>(

                                                          context,

                                                          listen: true)

                                                      .state

                                                      .storedCurrency !=

                                                  ""

                                              ? BlocProvider.of<SavedCubit>(

                                                      context,

                                                      listen: true)

                                                  .state

                                                  .storedCurrency

                                              : currencyDetection),

                                    ),

                                  ),

                                );

                              } else {

                                return CircularProgressIndicator();

                              }

                            }),

                          ))),

Bloc part

import 'dart:async';

import 'package:flutter/animation.dart';

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

import 'package:offerji/bloc/store\_cubit/saved\_currency\_state.dart';

import 'package:offerji/bloc/store\_cubit/update\_store\_event.dart';

class SavedCubit extends Bloc<UpdateStoreValueEvent, SavedCurrencyState>{

  String? output;

  SavedCubit() : super(SavedCurrencyState(storedCurrency: "")) {

    on<UpdateStoreValueEvent>(onCampaignEvent);

  }

   FutureOr<void> onCampaignEvent(

      UpdateStoreValueEvent event, Emitter<SavedCurrencyState> emit) async {

        output=event.inputValue;

    try {

      emit(SavedCurrencyState(storedCurrency:output));

    } catch (e) {

    }

  }

}

State.dart

import 'package:equatable/equatable.dart';

class SavedCurrencyState extends Equatable{

  String? storedCurrency;

  SavedCurrencyState({this.storedCurrency});

  @override

  List<Object?> get props => [storedCurrency];

}

Event.dart

import 'package:equatable/equatable.dart';

class UpdateStoreValueEvent extends Equatable{

  String? inputValue;

  UpdateStoreValueEvent({this.inputValue});

  @override

  List<Object?> get props =>[];

}

Fetch Stores Api

Event.dart

import 'package:equatable/equatable.dart';

 class FetchStoreEvent extends Equatable{

   @override

  List<Object?> get props => [];

}

State.dart

import 'package:equatable/equatable.dart';

import 'package:offerji/storepage/dataclass/UserStore.dart';

abstract class FetchStoreState extends Equatable{

  @override

  List<Object?> get props => [];

}

class FetchStoreInitState extends FetchStoreState{

  @override

  List<Object> get props => [];

}

class FetchStoreLoadingState extends FetchStoreState{

    @override

  List<Object> get props => [];

}

class FetchStoreLoadedState extends FetchStoreState{

final UserStore ? userStore;

String? storeCurrency;

FetchStoreLoadedState({this.userStore,this.storeCurrency});

 @override

  List<Object> get props => [];

}

class FetchStoreErrorState extends FetchStoreState{

    final error;

  FetchStoreErrorState({this.error});

  @override

  List<Object> get props =>[];

}

Bloc.dart

import 'dart:async';

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

import 'package:offerji/bloc/fetch\_store/fetc\_store\_event.dart';

import 'package:offerji/bloc/fetch\_store/fetch\_repository.dart';

import 'package:offerji/bloc/fetch\_store/fetch\_store\_state.dart';

import 'package:offerji/storepage/dataclass/UserStore.dart';

class FetchStoreBloc extends Bloc<FetchStoreEvent, FetchStoreState> {

  UserStore? userStore;

  String? storeCurrency;

  final FetchStoreRepo repository;

  FetchStoreBloc({required this.repository})

      : super(FetchStoreInitState()) {

    on<FetchStoreEvent>(\_onCampainEvent);

  }

  FutureOr<void> \_onCampainEvent(

      FetchStoreEvent event, Emitter<FetchStoreState> emit) async {

    emit(FetchStoreLoadingState());

    try {

      userStore = await repository.fetchVariants();

      emit(FetchStoreLoadedState(userStore: userStore,storeCurrency:storeCurrency ));

    } catch (e) {

      emit(FetchStoreErrorState(

          error: UnkwonExeption(messege: "Unknown exception")));

    }

  }

}

class NoInterNetExeption {

  var messege;

  NoInterNetExeption({this.messege});

}

class UnkwonExeption {

  var messege;

  UnkwonExeption({this.messege});

}

class HttpExeption {

  var messege;

  HttpExeption({this.messege});

}

Repository.dart

import 'dart:convert';

import 'package:offerji/storepage/dataclass/UserStore.dart';

import 'package:shared\_preferences/shared\_preferences.dart';

import 'package:offerji/common/common\_constants.dart' as constants;

import 'package:http/http.dart' as http;

abstract class FetchStoreRepo {

  Future<UserStore?> fetchVariants();

  String? storeCurrency;

}

class FetchStoreApi implements FetchStoreRepo {

  UserStore? userStore;

  String? storeCurrency;

  @override

  Future<UserStore?> fetchVariants() async {

    SharedPreferences prefs = await SharedPreferences.getInstance();

    String url = constants.UserStore\_Url +

        "?requestfrom=" +

        constants.RequestFrom +

        "&accessuserid=" +

        prefs.getString(constants.App\_Userid).toString() +

        "&accesskey=" +

        prefs.getString(constants.App\_Accesskey).toString() +

        "&page=1";

    final response =

        await http.get(Uri.parse(url), headers: {"Accept": "application/json"});

    print(response.statusCode);

    if (response.statusCode == 200) {

      var data = jsonDecode(response.body);

      var rest = data['code'];

      if (response.statusCode == 200) {

        userStore = UserStore.fromJson(jsonDecode(response.body));

        storeCurrency=userStore!.success[0].storecurrency;

        return userStore;

      }

      return userStore;

    }

  }

}

Fetch Currency Api

Event.dart

 import 'package:equatable/equatable.dart';

class FetchCurrencyEvent extends Equatable{

  @override

  List<Object?> get props => [];

}

State.dart

import 'package:equatable/equatable.dart';

abstract class FetchCurrencyState extends Equatable{

  @override

  List<Object?> get props => [];

}

class FetchCurrencyInitState extends FetchCurrencyState{

  @override

  List<Object> get props => [];

}

class FetchCurrencyLoadingState extends FetchCurrencyState{

    @override

  List<Object> get props => [];

}

class FetchCurrencyLoadedState extends FetchCurrencyState{

final List ? offerDetails;

FetchCurrencyLoadedState({this.offerDetails});

 @override

  List<Object> get props => [];

}

class FetchCurrencyErrorState extends FetchCurrencyState{

    final error;

  FetchCurrencyErrorState({this.error});

  @override

  List<Object> get props =>[];

}

Bloc.dart

import 'dart:async';

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

import 'fetch\_currency\_event.dart';

import 'fetch\_currency\_repository.dart';

import 'fetch\_currency\_state.dart';

class FetchCurrencyBloc extends Bloc<FetchCurrencyEvent, FetchCurrencyState> {

  List data = [];

  final FetchCurrencyRepo repository;

  FetchCurrencyBloc({required this.repository})

      : super(FetchCurrencyInitState()) {

    on<FetchCurrencyEvent>(\_onCampainEvent);

  }

  FutureOr<void> \_onCampainEvent(

      FetchCurrencyEvent event, Emitter<FetchCurrencyState> emit) async {

    emit(FetchCurrencyLoadingState());

    try {

      data = await repository.fetchVariants();

      emit(FetchCurrencyLoadedState(offerDetails: data));

    } catch (e) {

      emit(FetchCurrencyErrorState(

          error: UnkwonExeption(messege: "Unknown exception")));

    }

  }

}

class NoInterNetExeption {

  var messege;

  NoInterNetExeption({this.messege});

}

class UnkwonExeption {

  var messege;

  UnkwonExeption({this.messege});

}

class HttpExeption {

  var messege;

  HttpExeption({this.messege});

}

Repository.dart

import 'dart:convert';

import 'package:offerji/common/common\_constants.dart' as constants;

import 'package:shared\_preferences/shared\_preferences.dart';

import 'package:http/http.dart' as http;

abstract class FetchCurrencyRepo {

  Future<List<dynamic>> fetchVariants();

}

class FetchCurrencyApi implements FetchCurrencyRepo {

  List data = [];

  List catData = [];

  String catSelection = '';

  @override

  Future<List> fetchVariants() async {

    SharedPreferences prefs = await SharedPreferences.getInstance();

    String url1 = constants.CurrencyList\_Url +

        "?requestfrom=" +

        constants.RequestFrom +

        "&accesskey=" +

        prefs.getString(constants.App\_Accesskey).toString() +

        "&accessuserid=" +

        prefs.getString(constants.App\_Userid).toString();

    final response = await http

        .get(Uri.parse(url1), headers: {"Accept": "application/json"});

    final int statusCode = response.statusCode;

    print("session : " + response.statusCode.toString());

    print("body : " + response.body);

    if (statusCode == 401) {

      print('Session expired... please try relogging in.');

      // call logout api

    } else {

      String str = response.body;

      Map<String, dynamic> keys1 = json.decode(str);

      List myString = keys1["currency"].keys.toList();

      List catsList = keys1["category"].toList();

      data = keys1["currency"].keys.toList();

      catData = keys1["category"].toList();

      catSelection = prefs.getString(constants.Storecategory).toString();

    }

    return data;

  }

}

Edit offer Api

Event.dart

import 'package:equatable/equatable.dart';

import '../../models/drafts/OfferDetails.dart';

abstract class FetchOfferEvent extends Equatable{

  @override

  List<Object> get progs => [];

}

class InitFetch extends FetchOfferEvent{

  final OfferDetails ? offerDetails;

InitFetch({this.offerDetails});

  @override

  List<Object?> get props => [offerDetails];

}

State.dart

import 'package:equatable/equatable.dart';

import '../../models/drafts/OfferDetails.dart';

abstract class EditOfferState extends Equatable{

  @override

  List<Object?> get props => [];

}

class EditInitState extends EditOfferState{

  @override

  List<Object> get props => [];

}

class EditOfferLoadingState extends EditOfferState{

    @override

  List<Object> get props => [];

}

class EditOfferLoadedState extends EditOfferState{

final OfferDetails ? offerDetails;

final String? id;

EditOfferLoadedState({this.offerDetails,this.id});

 @override

  List<Object> get props => [];

}

class EditOfferErrorState extends EditOfferState{

    final error;

  EditOfferErrorState({this.error});

  @override

  List<Object> get props =>[];

}

Bloc.dart

import 'dart:async';

import 'dart:io';

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

import 'package:offerji/bloc/fetch\_varients/repository.dart';

import 'package:offerji/bloc/fetch\_varients/state.dart';

import 'package:offerji/models/drafts/OfferDetails.dart';

import 'event.dart';

class FetchBloc extends Bloc<FetchOfferEvent,EditOfferState> {

    OfferDetails? offerDetails;

    String? id;

  final SignInRepository repository;

  FetchBloc({required this.repository}) : super(EditInitState()) {

    on<FetchOfferEvent>(\_onCampainEvent);

  }

  // @override

  // Stream<EditOfferState> mapEventToState(FetchOfferEvent event) async\* {

  //  if (event is InitFetch) {

  //   yield EditOfferLoadingState();

  //   try {

  //    offerlist=await signInRepository.fetchVariants();

  //    yield EditOfferLoadedState(offerDetails: offerlist,);

  //   } on SocketException {

  //    yield EditOfferErrorState(error: NoInterNetExeption(messege:"No internet"));

  //        }

  //        catch (e) {

  //         yield EditOfferErrorState(error: UnkwonExeption(messege: "Uknownexeption"));

  //        }

  //   }

    FutureOr<void> \_onCampainEvent(

      FetchOfferEvent event, Emitter<EditOfferState> emit) async {

    emit(EditOfferLoadingState());

    try {

      offerDetails = await repository.fetchVariants();

      id=await repository.id;

      emit(EditOfferLoadedState(offerDetails: offerDetails,id: id));

    } catch (e) {

      emit(EditOfferErrorState(

          error: UnkwonExeption(messege: "Unknown exception")));

    }

  }

}

 class NoInterNetExeption {

 var messege;

 NoInterNetExeption({this.messege});

 }

 class UnkwonExeption{

 var messege;

 UnkwonExeption({this.messege});

 }

 class HttpExeption{

  var messege;

  HttpExeption({this.messege});

}

Repository.dart

import 'dart:convert';

import 'package:shared\_preferences/shared\_preferences.dart';

import '../../models/drafts/OfferDetails.dart';

import 'package:offerji/common/common\_constants.dart' as constants;

import 'package:http/http.dart' as http;

abstract class SignInRepository {

  Future<OfferDetails?> fetchVariants();

  String ?id;

}

class FetchApi implements SignInRepository {

  OfferDetails? fp;

  String? id;

  @override

  Future<OfferDetails?> fetchVariants() async {

    try {

      SharedPreferences prefs = await SharedPreferences.getInstance();

      var headers = {

        "Accept": "application/json",

        "Authorization": "key=${prefs.getString(constants.App\_Accesskey)}"

      };

      String url = constants.FetchOfferDetails\_Url +

          "?requestfrom=" +

          constants.RequestFrom +

          "&accessuserid=" +

          prefs.getString(constants.App\_Userid).toString() +

          "&accesskey=" +

          prefs.getString(constants.App\_Accesskey).toString() +

          "&storeid=" +

          prefs.getString(constants.Storeid).toString() +

          "&offerid=" +

          constants.currOfferId;

      final drafts = await http.get(Uri.parse(url), headers: headers);

      print("variants: " + drafts.body + url + headers.toString());

      if (drafts.statusCode == 200) {

        fp = OfferDetails.fromJson(json.decode(drafts.body));

        id="123";

      }

    } on Exception catch (e) {

      print(e);

    }

    return fp;

  }

}

Ui Side

//iniit state

 context.read<FetchBloc>().add(InitFetch());

SingleChildScrollView(

            child: BlocListener<FetchBloc, EditOfferState>(

          listener: (context, state) {

            if (state is EditOfferErrorState) {

              Fluttertoast.showToast(msg: "Error Occured is" + state.error);

            }

            else if(state is EditOfferLoadedState){

              print("id value is"+state.id.toString());

              etofferNameController.text=state.offerDetails!.success![0].offertitle.toString();

              etofferTypeController.text=state.offerDetails!.success![0].offertype.toString();

              etofferDescController.text=state.offerDetails!.success![0].offerdescription.toString();

              etstartDateController.text=state.offerDetails!.success![0].offerstartdate.toString().substring(0,19);

              etendDateController.text=state.offerDetails!.success![0].offerenddate.toString().substring(0,19);

            }

          },

          child: BlocBuilder<FetchBloc, EditOfferState>(

              builder: ((context, state) {

            if (state is EditOfferErrorState) {

            return Center(child: Container(child: Text(state.error)),);

            }

            else if(state is EditOfferLoadingState){

           return Center(

                child: ColorLoader(

              dotIcon: Icon(Icons.adjust),

            ));

            }else if (state is EditOfferLoadedState){

              offerDetailsUi=state.offerDetails;

              return buildWidget(offerDetailsUi);

            }

          else{

            return Container(child: Text("none of above"),);

          }

          })),

        )));

  }

Adding Bloc Testing With Sample Data Without Event

bloc.dart

class WetherBloc extends Bloc<WeatherEvent, WeatherState> {

  Weather? weatherModel;

  WeatherRepository? weatherRepository;

  WetherBloc({this.weatherRepository}) : super(WeatherInitial());

  @override

  WeatherState get initialState => WeatherInitial();

  void onWetherEvent() async {

    emit(WeatherLoadInProgress());

    try {

      print("case executed");

      weatherModel = await weatherRepository!.getWeather(city: "hyderabad");

      emit(WeatherLoadSuccess(weather: weatherModel!));

    } catch (e) {

      emit(WeatherLoadFailure());

    }

  }

}

class NoInterNetExeption {

  var messege;

  NoInterNetExeption({this.messege});

}

class UnkwonExeption {

  var messege;

  UnkwonExeption({this.messege});

}

class HttpExeption {

  var messege;

  HttpExeption({this.messege});

}

State.dart

import 'package:bloc\_tes/bloc/wether\_model.dart';

import 'package:equatable/equatable.dart';

import 'package:meta/meta.dart';

abstract class WeatherState extends Equatable {

  const WeatherState();

  @override

  List<Object> get props => [];

}

class WeatherInitial extends WeatherState {}

class WeatherLoadInProgress extends WeatherState {}

class WeatherLoadSuccess extends WeatherState {

  final Weather weather;

  const WeatherLoadSuccess({required this.weather});

  @override

  List<Object> get props => [weather];

}

class WeatherLoadFailure extends WeatherState {

}

Model.dart

// weather.dart

import 'package:equatable/equatable.dart';

enum Condition { sunny, rainy, cloudy }

class Weather extends Equatable {

  final double temperature;

  final Condition condition;

  const Weather({

    required this.temperature,

    required this.condition,

  });

  @override

  List<Object> get props => [temperature, condition];

}

Repository.dart

import 'package:bloc\_tes/bloc/wether\_model.dart';

class WeatherRepository {

  Future<Weather> getWeather({required String city}) async {

    await Future.delayed(Duration(seconds: 1));

    return const Weather(

      temperature: 30,

      condition: Condition.sunny,

    );

  }

}

Test bloc.dart

import 'package:bloc\_tes/bloc/wether\_bloc.dart';

import 'package:bloc\_tes/bloc/wether\_event.dart';

import 'package:bloc\_tes/bloc/wether\_model.dart';

import 'package:bloc\_tes/bloc/wether\_repo.dart';

import 'package:bloc\_tes/bloc/wetherstate.dart';

import 'package:bloc\_test/bloc\_test.dart';

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

import 'package:mockito/mockito.dart';

class MockWetherRepo extends Mock implements WeatherRepository {

  Future<Weather> getWeather({required String city}) async {

    await Future.delayed(Duration(seconds: 1));

    return const Weather(

      temperature: 30,

      condition: Condition.sunny,

    );

  }

}

repoTest() {

  late WetherBloc wetherBloc;

  late WeatherRepository mockWetherRepo;

  setUp(() {

    mockWetherRepo = MockWetherRepo();

    wetherBloc = WetherBloc(weatherRepository: mockWetherRepo);

  });

  tearDown(() {

    wetherBloc.close();

  });

  group('bloc Test', () {

    blocTest<WetherBloc, WeatherState>(

        'emits [QuoteEmpty, QuoteLoading, QuoteLoaded] when FetchQuote is added and fetchQuote succeeds',

        build: () => wetherBloc,

        act: (bloc) => wetherBloc.onWetherEvent(),

    expect:() => <WeatherState>[

          WeatherLoadInProgress(),

          const WeatherLoadSuccess(

              weather: Weather(condition: Condition.sunny,temperature: 30))

        ]);

  });

}

Maintest.dart

import 'api\_testing.dart';

import 'counter\_cubit\_test.dart';

void main() {

  mainCubit();

  repoTest();

}