Install the Bloc extension on VScode

Bloc

Understanding Bloc (Project Based learning)

Bloc Listener Bloc Consumer Bloc Builder

1. features folder → Home Folder, Wishlist Folder, Cart Folder
2. data folder → groceryData.dart
   ↓
   Create a class GroceryData which will contain the list.

Features

Home → Ui Folder, Bloc Folder
- home_bloc.dart
- home_event.dart
- home_state.dart

Wishlist → Ui Folder

Cart → Ui Folder

Right Click on the folder & choose the Add Bloc option.

Every feature will have a seperate Bloc logic. It will have the logic of the feature.

So whenever we want to do anything on the UI
Eg: Liking a post.

This will be the flow:

event will be created → bloc will do the logical part → And a state is passed

↓

State will then update the UI.

- flutter pub add bloc
- flutter pub add flutter_bloc

Eg: when we click on the heart button, the post gets liked (the event of liking has taken place), the BLoC does the backend addition & returns a State that the post is liked.

1. Firstly we'll think of the possible events of a feature & make classes for the same in the home_event.dart.
   - For clicking the wishlist button.

```
class HomeProductWishListButtonClickedEvent
    extends HomeEvent {}
```

Similarly, for CartButtonClick

WishlistButtonNavigate

CartButtonNavigate

2. Lets now think about the States which will be emitted from the block.
   There are two types of states:
   1. Simple states - will build a UI portion
   2. Actionable states - will perform some action.

So we'll add one more abstract class in home_state.dart.

```
abstract class HomeActionState extends HomeState {}
```

Even for creating states we'll have classes.

1. class HomeLoadingState extends HomeState()
   → for showing the loading sign till we get the data.
2. class HomeLoadedState extends HomeState()
   → for displaying the items as soon as we open the app
3. class HomeErrorState extends " " " "
4. class HomeNavigateWishlistPage extends HomeActionState()
   → for Navigating
5. class HomeNavigateCartPage " " " "
   → for Navigating

THE UI should:
- Listen to the states emitted by the bloc.
- Accept events which the user is passing.

So we'll wrap our home.dart's Scaffold with BlocConsumer which will
- listen to events
- listen to states

BlocConsumer<HomeBloc, HomeState>

Create an instance of HomeBloc outside build

final HomeBloc homeBloc = HomeBloc();

i.e on this event, run this function

parameters of BlocConsumer:
bloc: homeBloc,

Add Favourite icon button & Cart Button to the appbar.

. Now we have to add an event which is to Navigate to the Cart / Wishlist Screen when it is pressed.

↳ onPressed: () {
homeBloc.add(HomeWishlistButtonNavigateEvent());
}
// Adding event

↳ Same for Cart

We'll now create the event in Bloc.
Inside the HomeBloc class

HomeBloc(): super(HomeInitial()) {
on<HomeProductWishlistButtonNavigateEvent>(homeProductWishlistButtonClickedEvent)
}

Add the method for it; simply click on the bulb & add the method.

Same for Cart Clicked
Wishlist clicked
Cart Navigate

In the methods created above we'll add
emit(HomeNavigateToWishlistPageActionState());

↳ This will emit the state

Now we have to actual listen to the States emitted by Bloc.
↳ We'll listen to it in the BlocProvider's listener : argument.

listenWhen : (current, previous) : => current is HomeActionState

buildWhen : (current, previous) : => current is !HomeActionState

> listen when the emitted state is ActionState build when the emitted states is not an Action state.

---

lets defining the listener :-

```
listener : (context, state) {
if (state is HomeNavigateToCartPageActionState)
{{
    Navigator.of(context).push(MaterialPageRoute
                builder(context) =>
                Cart()))
  }
else if
  {
    same for wishlist
  }}
```

We'll Now ~~get~~ to work to the product list.

First create a models folder in home folder & a file which will contain the definition of the product.

~~class~~

class ProductDataModal {

final String id;
final String ~~name~~ name,
description
Price, ImageUrl }

and also make a constructor.

When I am getting the state for HomeLoadedSuccessState, I should get a list of ProductDataModal

~~In th~~ In the class of HomeLoadedSuccessState we'll declare a list of ProductDataModel & its constructor.

Note: We'll also add more event that is HomeInitialEvent in the event class for the initial part also registers it in the Bloc.

homeInitialEvent will emit the HomeLoading State ();

Note We aren't fetching anything from the server but still we'll show a loader for 3 sec using async await.

await Future.delayed(Duration(seconds: 3))
& then emit LoadingSuccessState which will require 'products'.

This should be a list of ProductDataModel
So well map() it: GroceryData.groceryProducts
.map((e) => ProductDataModel(
id: e['id']
,
,
,
).toList();

For HomeInitialEvent to trigger at the start, we'll have InitState & inside it we'll add(HomeInitialEvent());

Initially we can get any state so we'll have to build accordingly using switch case.

```
switch (state.runTimeType){
  case HomeLoadingState:
    return Scaffold(body Centre(CircularProgressIndicator));

case HomeLoadedSuccessState:
  return
    Scaffold
(body appbar:
      Appbar, TwoIcons (Cart, Wishlist)
    )
```

We'll now work on creating a widget for displaying the product (product_tile_widget.dart) Build a Stateless Widget the way you want.

Things to remember:

- In home.dart have a list_view builder
- Make an instance of ProductDataModel in Product-tile widget.
- In HomeLoadedSuccessState make a variable to take the state.

```
final SuccessState = state as HomeLoadedSuccessState;
```

- Use this variable in ListView builder to get the products from list
  ProductDataModel: successState.products[index]
  itemCount: successState.products.length
- Use NetworkImage in the product-tile
- Add Buttons in the product-tile to add the product to wishlist or Cart.

Now We'll have to see how we can add the items to the wishlist or Cart.

- For that we'll first declare two empty lists in two seperate files in the data folder. (of type ProductDataModel)
- Create an instance of HomeBloc
  final HomeBloc homeBloc;
- In the onPressed section we'll use the instance to add event
  homeBloc.add(HomeProductWishlistButtonClickedEvent)

- We'll now be passing data through events. So in the hom_event.dart, we'll create an instance of ProductDataModal in both, the Card & the Wishlist Class.
- The onPressed will now ask for a value for clicked Product (the instance we created in the event class) ~~to us~~.

H.W. Manage removing the items.

- In the ~~Bloc~~. home_bloc we'll add the value to ~~the~~ the list.
  wishlistItems.add(event.clickedProduct)
  Same for Cart.

We also ~~want~~ want to see a Snakbar showing if an item is added.
- For that, we'll create an action State & emit it in the home_bloc.dart
- In the home.dart add two else if for the new action State we added & show the respective Snackbars.

Flow till Now → When we click on ~~on~~ one of the icons ~~the~~ the event will take place & by taking ProductDataModel data. Event given to BLoC. State will be emitted after addition of the event to the list By BLoC.

Lets now work on displaying the cartlist
- We'll create a ~~stateless~~ Stateful widget with the BlocConsumer as the body.
- Create an instance of CartBloc.
- Create event in the cart_event.dart.
  class CartInitialEvent extends CartEvent {}

- In the CartBloc mention the function on< ___ > (Here) & create it,
  emit(CartSuccessState(cartItems: cartItems));
- In the cart.dart we'll add the CartInitial Event in the InitState
- We'll listen when: State is an ActionState & build when not.
- Go ahead with the Switch case in the builder:
  case CartSuccessState:
    Take the state in a variable.
  & return a ~~List~~ ListView builder - Copy it from home.dart & make the required changes
- Also create a new file for CartTile which will be a copy of ~~Home~~ ProductTile.