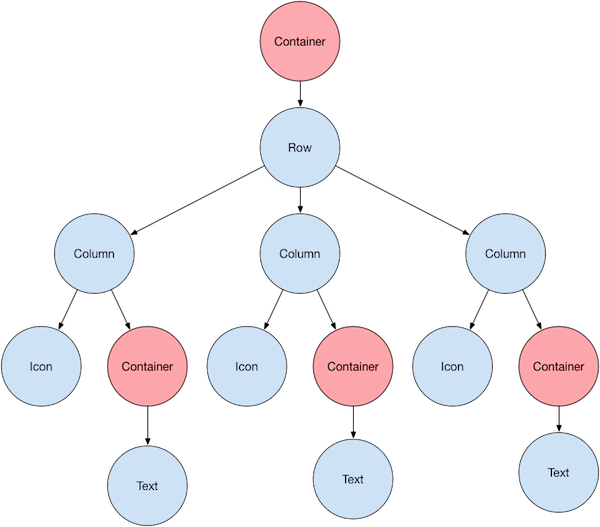
Flutter Basic Components

Structure of Flutter:



Flutter Life Cycle Methods:

1. **createState():** When the Framework is instructed to build a StatefulWidget, it immediately calls createState()
2. **mounted is true:** When createState creates your state class, a buildContext is assigned to that state. buildContext is, overly simplified, the place in the widget tree in which this widget is placed. Here's a longer explanation. All widgets have a bool this.mounted property. It is turned true when the buildContext is assigned. It is an error to call setState when a widget is unmounted.
3. **initState():** This is the first method called when the widget is created (after the class constructor, of course.) initState is called once and only once. It must call super.initState().
4. **didChangeDependencies():** This method is called immediately after initState on the first time the widget is built.
5. **build():** This method is called often. It is required, and it must return a Widget.
6. **didUpdateWidget(Widget oldWidget):** If the parent widget changes and has to rebuild this widget (because it needs to give it different data), but it's being rebuilt with the same runtimeType, then this method is called. This is because Flutter is re-using the state, which is long lived. In this case, you may want to initialize some data again, as you would in initState.
7. **setState():** This method is called often from the framework itself and from the developer. Its used to notify the framework that data has changed
8. **deactivate():** Deactivate is called when State is removed from the tree, but it might be reinserted before the current frame change is finished. This method exists basically because State objects can be moved from one point in a tree to another.
9. **dispose():** dispose() is called when the State object is removed, which is permanent. This method is where you should unsubscribe and cancel all animations, streams, etc.
10. **mounted is false:** The state object can never remount, and error will be thrown if setState is called.

Flutter Basic Flow

Any Flutter Programm Start executing from Main Flutter

void main() {

  runApp(const MyApp());

}

Then We need to Use Material App To Access the Material Properties Like Media Query and Material colors and etc.

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Flutter Demo',

      theme: ThemeData(

        primarySwatch: Colors.blue,

      ),

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

    );

  }

}

Important Widgets In Flutter

Column

Column is used to Add the Multiple Childrens in in Vertical Direction

Important Properties of Column:

1. mainAxisAlignment :The main Main Axis for Column is Vertical Direction ,The Main Axis Allignment is used to Allign the Coloum Properties.

 Column(

  crossAxisAlignment: CrossAxisAlignment.center,

  mainAxisAlignment: MainAxisAlignment.spaceEvenly,

  mainAxisSize: MainAxisSize.max,

  children: <Widget>[

    Text('Hello'),

    Text('World'),

  ],

)

Row

A Row in Flutter is a widget that displays its children in a horizontal arrangement. It's similar to a horizontal layout in other UI frameworks. The Row widget is commonly used to create forms, lists, and other UI elements that require a horizontal layout.

Row(

  crossAxisAlignment: CrossAxisAlignment.center,

  mainAxisAlignment: MainAxisAlignment.spaceEvenly,

  mainAxisSize: MainAxisSize.max,

  children: <Widget>[

    Text('Hello'),

    Text('World'),

  ],

)

Container

A Container in Flutter is a widget that provides a convenient way to create a visual element with customizable size, padding, margin, and decoration. It's a versatile widget that can be used to create various UI elements, such as buttons, cards, images, and text boxes.

Container(

  height: 100,

  width: 200,

  padding: EdgeInsets.all(16),

  margin: EdgeInsets.all(16),

  decoration: BoxDecoration(

    color: Colors.blue,

    border: Border.all(

      color: Colors.black,

      width: 2,

    ),

    borderRadius: BorderRadius.circular(8),

  ),

  child: Text('Hello World!'),

)

SizedBox

A SizedBox in Flutter is a widget that provides a fixed size box. It can be used to add spacing between UI elements, as well as to control the size of UI elements themselves.

To use a SizedBox in Flutter, you create an instance of the SizedBox widget and specify its width and/or height. Here's an example of using a SizedBox to add spacing between two Text widgets:

Column(

  children: <Widget>[

    Text('Hello'),

    SizedBox(height: 16),

    Text('World'),

  ],

)

Or

SizedBox(

  width: 100,

  height: 50,

  child: Text('Hello World'),

)

Icon in Flutter

In Flutter, an Icon is a graphical symbol that represents an action, a status, or an entity. Icons are commonly used in various UI elements, such as buttons, tabs, and menus, to help users understand the function or meaning of the UI element.

To use an Icon in Flutter, you create an instance of the Icon widget and specify the desired icon using an IconData object. The IconData object represents the icon data, such as its name, font family, and code point. Here's an example of using an Icon widget to display a favorite icon:

Icon(

  Icons.thumb\_up,

  size: 48,

  color: Colors.blue,

)

Image.asset

In Flutter, the Image.asset widget is used to display an image asset that is bundled with the app. An image asset can be a PNG, JPEG, GIF, WebP, or animated WebP file. To use an Image.asset widget, you must first add the image asset to the app's assets directory and then reference it using its asset name.

Image.asset(

  'assets/images/my\_image.png',

  width: 200,

  height: 200,

  fit: BoxFit.cover,

  alignment: Alignment.center,

)

Scaffold

In Flutter, a Scaffold widget is a basic material design layout that provides a structure for your app and contains several common elements, such as a top app bar, a bottom navigation bar, and a body. The Scaffold widget allows you to easily create a consistent app layout and provides a set of properties to customize its appearance and behavior.

Scaffold(

  appBar: AppBar(

    title: Text('My App'),

  ),

  body: Center(

    child: Text('Hello, World!'),

  ),

)

Floating Action Button In Flutter, a FloatingActionButton (FAB) is a button that appears above the content to provide the primary action for the current screen. A FAB is typically used for a high-priority, primary action, such as adding a new item or opening a new screen.

To use a FloatingActionButton in Flutter, you create an instance of the FloatingActionButton widget and specify its properties, such as its icon and onPressed callback. Here's an example of using a FloatingActionButton to display an add icon

Scaffold(

  floatingActionButton: FloatingActionButton(

    onPressed: () {

      // Perform the FAB action

    },

    child: Icon(Icons.add),

  ),

)

TextField

In Flutter, a TextField widget is used to create a text input field that allows users to enter text. The TextField widget provides a range of properties and callbacks to customize the appearance and behavior of the text input field.

TextField(

  decoration: InputDecoration(

    labelText: 'Enter your password',

  ),

  obscureText: true,

  keyboardType: TextInputType.visiblePassword,

)

ListTile

In Flutter, a ListTile widget is used to create a single row in a list or a drawer, typically containing some text and an optional leading or trailing icon. The ListTile widget provides a range of properties and callbacks to customize the appearance and behavior of the list item.

ListTile(

  leading: Icon(Icons.person),

  title: Text('John Doe'),

  subtitle: Text('johndoe@example.com'),

  trailing: Icon(Icons.arrow\_forward),

)

Image.network

In Flutter, the Image.network widget is used to display an image from a network URL. This widget is useful when you need to display images that are not bundled with your app and need to be downloaded from a server.

Image.network(

  'https://example.com/images/myimage.jpg',

)

Listview

In Flutter, the ListView widget is used to display a scrollable list of widgets. This widget is useful when you need to display a large number of items that do not fit on the screen at once.

To use the ListView widget, you simply create an instance of the widget and pass a list of child widgets to the children property. Here's an example

ListView(

  children: [

    ListTile(title: Text('Item 1')),

    ListTile(title: Text('Item 2')),

    ListTile(title: Text('Item 3')),

    ListTile(title: Text('Item 4')),

    ListTile(title: Text('Item 5')),

  ],

)

Card

In Flutter, the Card widget is used to display a material design card, which is a rectangular piece of material that acts as an entry point to more detailed information. A card typically contains a header, a footer, and a body, which can include text, images, and other widgets.

Card(

  child: Column(

    children: [

      ListTile(

        title: Text('Card Title'),

        subtitle: Text('Card Subtitle'),

        leading: Icon(Icons.image),

      ),

    ],

  ),

)

Single Child ScrollView

In Flutter, the SingleChildScrollView widget is used to create a scrollable view that contains a single child. This widget is useful when you have a large widget that may not fit on the screen at once, such as a form or a list.

SingleChildScrollView(

  child: Column(

    children: [

      Text('Item 1'),

      Text('Item 2'),

      Text('Item 3'),

      Text('Item 4'),

      Text('Item 5'),

      Text('Item 6'),

      Text('Item 7'),

      Text('Item 8'),

      Text('Item 9'),

      Text('Item 10'),

    ],

  ),

)

TextButton

TextButton is a material design button widget that provides a simple flat button style without any background or elevation. It is commonly used for less important actions or secondary actions in an app's user interface.

TextButton(

  onPressed: () {

    // Action to perform when the button is pressed

  },

  child: Text('Press me!'),

)

Gesture Detector

GestureDetector is a widget in Flutter that allows you to recognize various touch events, such as taps, drags, and scrolls, on its child widget. It provides a way to add interactivity to your app's user interface and respond to user gestures.

GestureDetector(

  onTap: () {

    // Action to perform when the user taps on the widget

  },

  onDoubleTap: () {

    // Action to perform when the user double-taps on the widget

  },

  onLongPress: () {

    // Action to perform when the user long-presses on the widget

  },

  child: Container(

    width: 200,

    height: 200,

    color: Colors.blue,

    child: Center(

      child: Text(

        'Tap me!',

        style: TextStyle(

          color: Colors.white,

          fontSize: 20,

        ),

      ),

    ),

  ),

)

Padding

Padding is a widget in Flutter that provides padding to its child widget. It is commonly used to add spacing between widgets or to add padding around the edges of a widget.

Padding(

  padding: EdgeInsets.all(20),

  child: Text(

    'Hello, world!',

    style: TextStyle(fontSize: 20),

  ),

)

AppBar

AppBar is a widget in Flutter that provides a top app bar that can contain navigation buttons, titles, and other actions. It is typically used as a part of a Scaffold widget, which provides a basic structure for an app page with a top app bar, a body, and a bottom navigation bar.

Scaffold(

appBar: AppBar(

title: Text('My App'),

actions: [

IconButton(

icon: Icon(Icons.search),

onPressed: () {

// Action to perform when the search button is pressed

},

),

IconButton(

icon: Icon(Icons.settings),

onPressed: () {

// Action to perform when the settings button is pressed

},

),

],

),

body: Center(

child: Text('Hello, world!'),

),

)

Flutter Toast

dependencies:

fluttertoast: ^8.0.7

 Fluttertoast.showToast(

  msg: 'Hello, world!',

  toastLength: Toast.LENGTH\_SHORT,

  gravity: ToastGravity.BOTTOM,

  backgroundColor: Colors.grey,

  textColor: Colors.white,

  fontSize: 16.0,

);

Expanded

Expanded is a widget in Flutter that is used to make a child widget fill the available space along a specific axis within its parent widget. It is typically used as a child widget within a Row, Column, or Flex widget to ensure that the child widget takes up as much space as possible along a specific axis.

   Row(

  children: [

    Expanded(

      child: Container(

        color: Colors.blue,

        height: 100,

      ),

    ),

    Expanded(

      child: Container(

        color: Colors.green,

        height: 100,

      ),

    ),

    Expanded(

      child: Container(

        color: Colors.red,

        height: 100,

      ),

    ),

  ],

)

CircularProgressIndicator

CircularProgressIndicator is a built-in widget in Flutter that displays a circular progress indicator, typically used to indicate that a task is in progress and the app is not frozen. It is commonly used in combination with async operations such as network requests, database queries, or file system access.

Center(

  child: CircularProgressIndicator(),

)

DropDown

 String? \_chosenValue = "All";

  String fromdatelogin = '';

  List<String> itemstochoose = [

    "All",

    "Transferred",

    "Open",

    "Partial",

    "Executed"

  ];

Container(

                              padding: EdgeInsets.only(left: size.width \* 0.01),

                              decoration: BoxDecoration(

                                  borderRadius: BorderRadius.circular(10),

                                  color: Colors.white),

                              width: size.width \* 0.35,

                              height: size.height \* 0.054,

                              child: DropdownButton<String>(

                                icon: Icon(FontAwesomeIcons.angleDown),

                                underline: Container(color: Colors.transparent),

                                focusColor: Colors.white,

                                value: \_chosenValue,

                                //elevation: 5,

                                style: TextStyle(color: Colors.white),

                                iconEnabledColor: Colors.black,

                                items: itemstochoose

                                    .map<DropdownMenuItem<String>>((value) {

                                  return DropdownMenuItem<String>(

                                    value: value,

                                    child: Text(

                                      value,

                                      style: TextStyle(color: Colors.black),

                                    ),

                                  );

                                }).toList(),

                                hint: Text(

                                  "Select",

                                  style: GoogleFonts.roboto(

                                      color: Colors.black,

                                      fontSize: size.height\*0.017,

                                      fontWeight: FontWeight.w500),

                                ),

                                onChanged: (value) {

                                  setState(() {

                                    \_chosenValue = value;

                                  });

                                },

                              ),

                            ),

Stack

In Flutter, a Stack widget is used to place widgets on top of each other. It is similar to a RelativeLayout in Android or a ZStack in SwiftUI. The widgets are placed in the order in which they are added to the Stack, with the first widget added being at the bottom of the stack, and the last widget added being at the top of the stack.

body: Stack(

children: [

Container(

width: 100,

height: 100,

decoration: BoxDecoration(color: Colors.green),

),

Positioned(

left: 30,

child: Container(

width: 80,

height: 80,

color: Colors.grey,

)),

],

));

Linear Gradient

Linear gradients are a type of gradient that creates a smooth transition of colors or shades in a linear direction. The colors in a linear gradient blend together in a linear fashion, creating a smooth transition between them. The starting point and ending point of the gradient are usually specified, and the gradient is drawn between these two points.

import 'package:flutter/material.dart';

class MyGradientWidget extends StatelessWidget {

@override

Widget build(BuildContext context) {

return Container(

decoration: BoxDecoration(

gradient: LinearGradient(

begin: Alignment.topLeft,

end: Alignment.bottomRight,

colors: [

Colors.blue,

Colors.red,

],

),

),

child: Center(

child: Text(

'Hello, World!',

style: TextStyle(

color: Colors.white,

fontSize: 24,

fontWeight: FontWeight.bold,

),

),

),

);

}

}

Tab Bar View

return DefaultTabController(

length: 3,

child: Scaffold(

appBar: AppBar(

automaticallyImplyLeading: true,

bottom: const TabBar(

tabs: [

Tab(icon: Icon(Icons.flight)),

Tab(icon: Icon(Icons.directions\_transit)),

Tab(icon: Icon(Icons.directions\_car)),

],

),

title: Text('Tabs Demo'),

),

body: const TabBarView(

children: [

Icon(Icons.flight, size: 350),

Icon(Icons.directions\_transit, size: 350),

Icon(Icons.directions\_car, size: 350),

],

),

),

);

}

**Form Validation in Flutter**

**class** HomePage extends StatefulWidget {

  @override

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

}

**class** \_HomePageState extends State<HomePage> {

  var \_formKey = GlobalKey<FormState>();

  var isLoading = **false**;

**void** \_submit() {

    final isValid = \_formKey.currentState.validate();

**if** (!isValid) {

**return**;

    }

    \_formKey.currentState.save();

  }

  @override

  Widget build(BuildContext context) {

**return** Scaffold(

      appBar: AppBar(

        title: Text("Form Validation"),

        leading: Icon(Icons.filter\_vintage),

      ),

      //body

      body: Padding(

        padding: **const** EdgeInsets.all(16.0),

        //form

        child: Form(

          key: \_formKey,

          child: Column(

            children: <Widget>[

              Text(

                "Form-Validation In Flutter ",

                style: TextStyle(fontSize: 24.0, fontWeight: FontWeight.bold),

              ),

              //styling

              SizedBox(

                height: MediaQuery.of(context).size.width \* 0.1,

              ),

              TextFormField(

                decoration: InputDecoration(labelText: 'E-Mail'),

                keyboardType: TextInputType.emailAddress,

                onFieldSubmitted: (value) {

                  //Validator

                },

                validator: (value) {

**if** (value.isEmpty ||

                      !RegExp(r"^[a-zA-Z0-9.a-zA-Z0-9.!#$%&'\*+-/=?^\_`{|}~]+@[a-zA-Z0-9]+\.[a-zA-Z]+")

                          .hasMatch(value)) {

**return** 'Enter a valid email!';

                  }

**return** null;

                },

              ),

              //box styling

              SizedBox(

                height: MediaQuery.of(context).size.width \* 0.1,

              ),

              //text input

              TextFormField(

                decoration: InputDecoration(labelText: 'Password'),

                keyboardType: TextInputType.emailAddress,

                onFieldSubmitted: (value) {},

                obscureText: **true**,

                validator: (value) {

**if** (value.isEmpty) {

**return** 'Enter a valid password!';

                  }

**return** null;

                },

              ),

              SizedBox(

                height: MediaQuery.of(context).size.width \* 0.1,

              ),

              RaisedButton(

                padding: EdgeInsets.symmetric(

                  vertical: 10.0,

                  horizontal: 15.0,

                ),

                child: Text(

                  "Submit",

                  style: TextStyle(

                    fontSize: 24.0,

                  ),

                ),

                onPressed: () => \_submit(),

              )

            ],

          ),

        ),

      ),

    );

  }

}

Animations Flutter

Hero Animations:

Step1:we need to create a Stateless Widget for Hero Animation we can re use when ever Required

**class** HeroAnimation **extends** StatelessWidget {  
 HeroAnimation({ Key key, **this**.**photo**, **this**.**onTap**, **this**.**width** ,**this**.**index**}) : **super**(key: key);  
 **final** String **photo**;  
 **final** VoidCallback **onTap**;  
 **final** double **width**;  
 int **index**;  
 @override  
 Widget build(BuildContext context) {  
 **return** Container(  
 height: 150,  
 margin: EdgeInsets.all(5.0),  
 child: Hero(  
 flightShuttleBuilder: (  
 BuildContext flightContext,  
 Animation<double> animation,  
 HeroFlightDirection flightDirection,  
 BuildContext fromHeroContext,  
 BuildContext toHeroContext,  
 )  
 {  
 **final** Hero toHero = toHeroContext.**widget**;  
 **return** RotationTransition(  
  
 turns: animation,  
 child: toHero.**child**,  
 );  
  
 },  
 tag: **photo**+**index**.toString(),  
 child: Material(  
 color: Colors.*transparent*,  
 child: InkWell(  
 onTap: **onTap**,  
 child: Image.asset(  
 **photo**,  
 fit: BoxFit.**contain**,  
 ),  
 ),  
 ),  
 ),  
 );  
 }  
}

Note:here flutter shuttle builder is optional we can use this for spectial effects only

step2: Use this Hero Animation Where ever required in this case we are using inside List view builder.

**class** \_ChewieDemoState **extends** State<ChewieDemo> {  
String **imagepath**=**'images/pppp.jpg'**;  
List<String>**list**=[**'images/pppp.jpg'**,**'images/ramya.jpg'**];  
 @override  
 **void** initState() {  
 **super**.initState();  
  
 }  
   
 @override  
 Widget build(BuildContext context) {  
 **return** MaterialApp(  
 home: Scaffold(  
 appBar: AppBar(  
 title: Text(**widget**.**title**),  
 ),  
 body: Column(  
 children: [  
 Expanded(child: ListView.builder(  
 itemCount: **list**.**length**,  
 itemBuilder: (context,index){  
 **return** HeroAnimation(  
 index: index,  
 photo: **list**[index],  
 onTap: () {  
 *// Navigator.push(context, new MaterialPageRoute( builder: (context)=>SecondPage(  
 // list[index]  
 // )));* Navigator.*push*(  
 context,  
 PageRouteBuilder(  
 transitionDuration:  
 Duration(seconds: 1),  
 pageBuilder: (\_, \_\_, \_\_\_) =>  
 HeroAnimation(  
 index: index,  
 photo: **list**[index],  
 onTap: (){  
 Navigator.*pop*(context);  
 },  
 )));  
 });  
  
  
  
 }))],  
 ),  
 ));  
 }  
}

Note:In this example we display the hero image in the same page. we can also navigate to next page and display our hero image

Step:3

**class** SecondPage **extends** StatelessWidget{  
 String **path**;  
 SecondPage(**this**.**path**);  
 @override  
 Widget build(BuildContext context) {  
 **return** Scaffold(  
 body: Column(  
 children: [  
 Center(  
 child:HeroAnimation(  
  
 index: 1,  
 photo:**path** ,  
 ) ,  
 )  
 ],  
 ),  
 );  
 }  
  
}

Note: just we pass the selected index image to this page and dislayed here

Tween Animation:

class SecondScreenWidget extends State<SecondScreen>

with SingleTickerProviderStateMixin {

late Animation<double> animation;

late AnimationController controller;

late Animation colorAnimation;

@override

void initState() {

super.initState();

controller =

AnimationController(duration: const Duration(seconds: 3), vsync: this);

animation = Tween<double>(begin: 200, end: 300).animate(controller);

colorAnimation =

ColorTween(begin: Colors.red, end: Colors.green).animate(controller);

animation.addListener(() {

setState(() {});

});

controller.forward();

}

@override

void dispose() {

controller.dispose();

super.dispose();

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: const Text('tween Animation'),

centerTitle: true,

backgroundColor: Colors.teal,

),

body: Column(

children: [

Container(

padding: const EdgeInsets.all(16),

height: animation.value,

width: animation.value,

child: Center(

child: Image.network(

"https://www.hdwallpapers.in/download/pink\_cosmos\_flowers-wide.jpg")),

),

Container(

margin: const EdgeInsets.all(10),

height: animation.value,

width: animation.value,

color: colorAnimation.value,

),

],

),

);

}

}

Pick Image From Gallery and camera

import 'dart:io';

import 'package:flutter/material.dart';

import 'package:image\_picker/image\_picker.dart';

import 'package:sampleapp/second\_screen.dart';

import 'package:sampleapp/third\_screen.dart';

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

import 'fourth\_screen.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

home: MyPickImageScreen(),

);

}

}

class MyPickImageScreen extends StatefulWidget {

@override

\_MyPickImageScreenState createState() => \_MyPickImageScreenState();

}

class \_MyPickImageScreenState extends State<MyPickImageScreen> {

File? imgFile;

final imgPicker = ImagePicker();

Future<void> showOptionsDialog(BuildContext context) {

return showDialog(

context: context,

builder: (BuildContext context) {

return AlertDialog(

title: Text("Options"),

content: SingleChildScrollView(

child: ListBody(

children: [

GestureDetector(

child: Text("Capture Image From Camera"),

onTap: () {

openCamera();

},

),

Padding(padding: EdgeInsets.all(10)),

GestureDetector(

child: Text("Take Image From Gallery"),

onTap: () {

openGallery();

},

),

],

),

),

);

});

}

void openCamera() async {

var imgCamera = await imgPicker.getImage(source: ImageSource.camera);

setState(() {

imgFile = File(imgCamera!.path);

});

Navigator.of(context).pop();

}

void openGallery() async {

var imgGallery = await imgPicker.pickImage(source: ImageSource.gallery);

setState(() {

imgFile = File(imgGallery!.path);

});

Navigator.of(context).pop();

}

Widget displayImage() {

if (imgFile == null) {

return Text("No Image Selected!");

} else {

return Image.file(imgFile!);

}

}

@override

Widget build(BuildContext context) {

return Scaffold(

appBar: AppBar(

title: Text("Hello"),

),

body: Center(

child: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

displayImage(),

SizedBox(height: 30),

TextButton(

onPressed: () {

showOptionsDialog(context);

},

child: Text("Select Image"),

)

],

),

),

);

}

}

Async and Await

In Flutter, async and await are used to handle asynchronous operations.

async is used to define a function that will perform an asynchronous operation. When you mark a function as async, you can use the await keyword to wait for the completion of a future or stream.

await is used inside an async function to wait for a future to complete before continuing execution of the function. When you use await, the function will pause execution until the future completes and returns a value.

Here's an example of how to use async and await in Flutter

Future<String>fetchData() async {

final response = await http.get(Uri.parse('https://example.com/api/data'));

if (response.statusCode == 200) {

return response.body;

} else {

throw Exception('Failed to load data');

}

}

BottomSheet

void \_showPicker(context) {

showModalBottomSheet(

context: context,

builder: (BuildContext bc) {

return SafeArea(

child: Container(

child: new Wrap(

children: <Widget>[

new ListTile(

leading: new Icon(Icons.photo\_library),

title: new Text('Photo Library'),

onTap: () {

\_imgFromGallery();

Navigator.of(context).pop();

}),

new ListTile(

leading: new Icon(Icons.photo\_camera),

title: new Text('Camera'),

onTap: () {

\_imgFromCamera();

Navigator.of(context).pop();

} ),], ),),);} );}}

Snackbar in Flutter

**final**snackBar = SnackBar(

                    content: Text(**'Yay! A SnackBar!'**),

                    action: SnackBarAction(

                      label: **'Undo'**,

                      onPressed: () {

                        Text(**'hello'**);

                      },

                    ),);

GestureDetector(

child: Text("Take Image From Gallery"),

onTap: () {

Navigator.of(context).pop();

ScaffoldMessenger.of(context).showSnackBar(snackBar);

// openGallery();

},

),

Search Filter

import 'dart:io';

import 'package:flutter/material.dart';

import 'package:image\_picker/image\_picker.dart';

import 'package:sampleapp/second\_screen.dart';

import 'package:sampleapp/third\_screen.dart';

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

import 'fourth\_screen.dart';

void main() {

runApp(MyApp());

}

class MyApp extends StatelessWidget {

@override

Widget build(BuildContext context) {

return MaterialApp(

debugShowCheckedModeBanner: false,

home: MyPickImageScreen(),

);

}

}

class MyPickImageScreen extends StatefulWidget {

@override

\_MyPickImageScreenState createState() => \_MyPickImageScreenState();

}

class \_MyPickImageScreenState extends State<MyPickImageScreen> {

List<Product> data = [

Product(id: "1", name: "rakesh", type: "Developer"),

Product(id: "2", name: "Tejas", type: "Devops"),

Product(id: "3", name: "Karthikeya", type: "Testing"),

];

List<Product> emptyList = [];

TextEditingController searchController = TextEditingController();

@override

Widget build(BuildContext context) {

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

return Scaffold(

appBar: AppBar(

leading: CircleAvatar(

radius: 16,

child: Icon(Icons.abc),

backgroundColor: Colors.green,

),

),

body: Column(

mainAxisAlignment: MainAxisAlignment.center,

children: [

Container(

width: size.width \* 0.9,

height: size.height \* 0.06,

child: TextField(

controller: searchController,

decoration: InputDecoration(hintText: "Search Item With Name"),

onChanged: (text) {

searchData(text);

},

),

),

Expanded(

child: ListView.builder(

itemCount:

searchController.text.isNotEmpty || emptyList.isNotEmpty

? emptyList.length

: data.length,

itemBuilder: (context, index) {

return emptyList.isNotEmpty

? Container(

child: Text(emptyList[index].name),

)

: Container(

child: Text(data[index].name),

);

}),

)

],

),

);

}

searchData(text) {

emptyList.clear();

if (text == null || text == "") {

setState(() {});

return;

} else {

data.forEach((element) {

if (element.name.toLowerCase().contains(text)) {

emptyList.add(element);

setState(() {});

}

});

}

}

}

class Product {

String id;

String name;

String type;

Product({required this.id, required this.name, required this.type});

}

Maps in Flutter

**initMap**:

Map<String, dynamic> user = {};

**Add Individual element in Map:**

Map<String, dynamic> persion1 = {

"timeintervalstart": widget.date

};

**Add all individual maps to user:**

user.addAll(persion1);

CLIP Rect

Refresh Indicator

Fontawesome

Device info plus

import 'dart:convert';

import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

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

import 'package:fluttertoast/fluttertoast.dart';

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

import 'package:hvpnl/business\_logic/blocs/log\_data\_operations/final\_logdatbase\_bloc/insert\_final\_logdata/insert\_log\_data\_bloc.dart';

import 'package:hvpnl/business\_logic/blocs/log\_data\_operations/final\_logdatbase\_bloc/insert\_final\_logdata/insert\_log\_data\_event.dart';

import 'package:hvpnl/business\_logic/blocs/log\_data\_operations/temp\_database/fetch\_temp\_data\_bloc/fetch\_temp\_data\_event.dart';

import 'package:hvpnl/business\_logic/blocs/log\_data\_operations/temp\_database/fetch\_temp\_data\_bloc/fetch\_temp\_state.dart';

import 'package:hvpnl/business\_logic/blocs/log\_data\_operations/temp\_database/insert\_temp\_database/temp\_data\_bloc.dart';

import 'package:hvpnl/business\_logic/blocs/operations/log\_data\_bloc/log\_data\_bloc.dart';

import 'package:hvpnl/business\_logic/blocs/operations/log\_data\_bloc/log\_data\_event.dart';

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

import 'package:hvpnl/business\_logic/blocs/operations/log\_data\_bloc/log\_data\_state.dart';

import 'package:hvpnl/business\_logic/blocs/operations/log\_data\_sync\_bloc/log\_data\_sync\_bloc.dart';

import 'package:hvpnl/business\_logic/blocs/operations/log\_data\_sync\_bloc/log\_data\_sync\_event.dart';

import 'package:sqflite/sqflite.dart';

import 'dart:math' as math;

import '../../../../business\_logic/blocs/log\_data\_operations/temp\_database/fetch\_temp\_data\_bloc/fetch\_temp\_bloc.dart';

import '../../../../business\_logic/blocs/log\_data\_operations/temp\_database/insert\_temp\_database/temp\_data\_event.dart';

import '../../../../business\_logic/cubits/connectivity\_cubit/connectivity\_cubit.dart';

import '../../../../database/temp\_databse.dart';

import '../../../../models/log\_data\_model.dart';

import '../../../../models/temp\_log\_db\_model.dart';

import '../../../../utils/helper.dart';

class BaySelectScreen extends StatefulWidget {

  final String frequencyId;

  final String subStationId;

  final String roasterId;

  final String timeSlot;

  const BaySelectScreen(

      {super.key,

      required this.frequencyId,

      required this.subStationId,

      required this.timeSlot,

      required this.roasterId});

  @override

  State<StatefulWidget> createState() {

    return BaySelectScreenWidget();

  }

}

class BaySelectScreenWidget extends State<BaySelectScreen> {

  Map<String, dynamic> data = {};

  List<Map<String, dynamic>> secondMap = [];

  List<Map<String, Object?>>? result;

  Map<String, dynamic> dataMap = {};

  String bayTypeName = "";

  @override

  void initState() {

    super.initState();

    fetchDataBase();

    BlocProvider.of<LogDataBloc>(context).onLogDataBloc(LogDataEvent(

        frequencyId: widget.frequencyId, subStationId: widget.subStationId));

  }

  Future fetchDataBase() async {

    Database? db = await DatabaseHelper.instance.database;

    result = await db!.query(

      DatabaseHelper.table,

      columns: [

        DatabaseHelper.bayId,

        DatabaseHelper.dataPart,

        DatabaseHelper.id,

        DatabaseHelper.frequencyId,

        DatabaseHelper.subStationId,

      ],

      where: 'frequency\_id = ? AND substation\_id =? AND time\_slot =?',

      whereArgs: [widget.frequencyId, widget.subStationId, widget.timeSlot],

    );

    print("result" + result.toString());

    result!.forEach((element) {

      TempLogTableModel cartModel = TempLogTableModel.fromMap(element);

      print(cartModel);

      var dataPart = jsonDecode(cartModel.dataPart.toString());

      // secondMap = [dataMap];

      dataMap[dataPart['bay\_type\_name']] = dataPart['data'];

    });

    print("dataMap");

    print(dataMap);

    return result;

  }

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      resizeToAvoidBottomInset: false,

      appBar: AppBar(

        toolbarHeight: 10.h,

        backgroundColor: Colors.white,

        flexibleSpace: Row(

          children: [

            SizedBox(

              width: 8.w,

            ),

            Container(

                margin: EdgeInsets.only(top: 1.h),

                child: Image.asset("assets/images/hvpnl\_logo.jpeg")),

            SizedBox(

              width: 15.w,

            ),

            Text(

              "Operation",

              style: GoogleFonts.sourceSans3(

                  fontWeight: FontWeight.w600,

                  color: Theme.of(context).primaryColor,

                  fontSize: 3.h),

            ),

            SizedBox(

              width: 20.w,

            ),

          ],

        ),

      ),

      body: Column(

        children: [

          logSheetHeader(context, "Shift Time : ${widget.timeSlot}",

              "${DateTime.now().toString().substring(0, 10)}"),

          BlocConsumer<LogDataBloc, LogDataState>(

            listener: (context, state) {},

            builder: (context, state) {

              if (state is LogDataErrorState) {

                return Container(

                  child: Text("some thing went wrong"),

                );

              }

              if (state is LogDataInitState || state is LogDataLoadingState) {

                return CircularProgressIndicator();

              }

              if (state is LogDataLoadedState) {

                return Expanded(

                    child: ListView.builder(

                        itemCount: state.loginDataModel.success.length,

                        itemBuilder: (context, index) {

                          return LogListWidget(

                            timeSlot: widget.timeSlot,

                            frequencyId: widget.frequencyId,

                            subStationId: widget.subStationId,

                            result: result!,

                            data: data,

                            secondMap: secondMap,

                            bayId:

                                state.loginDataModel.success[index].bayTypeName,

                            values: state.loginDataModel.success[index].values,

                            dataMap: dataMap,

                            bayName:

                                state.loginDataModel.success[index].bayName,

                            bayTypeName:

                                state.loginDataModel.success[index].bayTypeName,

                            buildContext: context,

                          );

                        }));

              }

              return Container();

            },

          ),

          Container(

              width: 50.w,

              height: 5.7.h,

              decoration: BoxDecoration(

                  borderRadius: BorderRadius.circular(10),

                  color: Theme.of(context).primaryColor),

              child: TextButton(

                  onPressed: () async {

                    result = await fetchDataBase();

                    List<TempLogTableModel> cartItems = [];

                    for (var element in result!) {

                      TempLogTableModel cartModel =

                          TempLogTableModel.fromMap(element);

                      cartItems.add(cartModel);

                    }

                    Map<String, dynamic> tempMap = {};

                    for (int i = 0; i < cartItems.length; i++) {

                      secondMap

                          .add(json.decode(cartItems[i].dataPart.toString()));

                    }

                    if (secondMap.isEmpty) {

                      Fluttertoast.showToast(

                          msg: "Please first save your data and submit");

                      return;

                    }

                    var connectionStatus =

                        BlocProvider.of<ConnectivityCubit>(context)

                            .state

                            .status;

                    print(widget.timeSlot);

                    if (connectionStatus == "ConnectivityResult.none") {

                      BlocProvider.of<InsertFinalLogDataBloc>(context)

                          .onInsertFinalLogData(InsertFinalLogDataEvent(

                              frequencyId: widget.frequencyId,

                              bayTypeId: "",

                              subStationId: widget.subStationId,

                              parameterValues: secondMap,

                              entryDate: DateTime.now().toString(),

                              entryTime:

                                  "${DateTime.now().hour}:${DateTime.now().minute}",

                              roasterId: widget.roasterId,

                              syncStatus: "false",

                              timeSlot: widget.timeSlot));

                    } else {

                      print(secondMap);

                      BlocProvider.of<InsertFinalLogDataBloc>(context)

                          .onInsertFinalLogData(InsertFinalLogDataEvent(

                              frequencyId: widget.frequencyId,

                              bayTypeId: "",

                              subStationId: widget.subStationId,

                              parameterValues: secondMap,

                              entryDate: DateTime.now().toString(),

                              entryTime:

                                  "${DateTime.now().hour}:${DateTime.now().minute}",

                              roasterId: widget.roasterId,

                              syncStatus: "true",

                              timeSlot: widget.timeSlot));

                      var res = await BlocProvider.of<LogDataSyncBloc>(context)

                          .onLogDataSuccess(LogDataSyncEvent(

                              entryDate: DateTime.now().toString(),

                              entryTime:

                                  "${DateTime.now().hour}:${DateTime.now().minute}",

                              frequencyId: widget.frequencyId,

                              paramValues: secondMap,

                              roasterId: widget.roasterId,

                              subStationId: widget.subStationId));

                      if (res!.code == 200) {

                        Fluttertoast.showToast(msg: "successful");

                        Navigator.of(context).pop();

                      } else {

                        Fluttertoast.showToast(msg: "some thing went wrong");

                      }

                    }

                  },

                  child: Text(

                    "Submit",

                    style: GoogleFonts.sourceSans3(

                        color: Colors.white, fontWeight: FontWeight.w600),

                  ))),

          SizedBox(

            height: 2.h,

          )

        ],

      ),

    );

  }

}

class LogListWidget extends StatefulWidget {

  final BuildContext buildContext;

  final String bayTypeName;

  final String bayName;

  final List<Value> values;

  final Map<String, dynamic> data;

  final String frequencyId;

  final String bayId;

  final Map<String, dynamic> dataMap;

  final List<Map<String, dynamic>> secondMap;

  final String subStationId;

  final List<Map<String, Object?>> result;

  final String timeSlot;

  const LogListWidget(

      {Key? key,

      required this.values,

      required this.frequencyId,

      required this.subStationId,

      required this.dataMap,

      required this.bayId,

      required this.secondMap,

      required this.timeSlot,

      required this.bayName,

      required this.data,

      required this.bayTypeName,

      required this.buildContext,

      required this.result})

      : super(key: key);

  @override

  \_\_ProductListWidgetState createState() => \_\_ProductListWidgetState();

}

class \_\_ProductListWidgetState extends State<LogListWidget>

    with TickerProviderStateMixin {

  late ThemeData themeData;

  Map<String, dynamic> insertedValues = {};

  List<bool> basicDetails = [true, false];

  Color? colorValue;

  bool valueEmpty = false;

  bool isSelected = false;

  List<Color> saveColor = [];

  List<TextEditingController> testController = [];

  GlobalKey<FormState> formKey = GlobalKey<FormState>();

  List<bool> validationList = [];

  late final AnimationController \_controller = AnimationController(

      vsync: this, duration: const Duration(milliseconds: 500));

  @override

  void initState() {

    super.initState();

    //  initialValue:widget

    //                                                 .dataMap[widget.bayTypeName] ==null?"" : widget

    //                                                 .dataMap[widget.bayTypeName]

    //                                             [widget.values[index]

    //                                                 .parameterName],

    // get

    if (widget.dataMap[widget.bayTypeName] != null) {

      insertedValues[widget.bayTypeName] = widget.dataMap[widget.bayTypeName];

    }

    BlocProvider.of<FetchTempDataBloc>(context).onFetchTempLogBloc(

        FetchTempDataEvent(

            frequencyId: widget.frequencyId,

            subStationId: widget.subStationId,

            timeSlot: widget.timeSlot));

  }

  // getFromLocalDB(){

  //     for (int i = 0;

  //                                   i < state.cartItems.length;

  //                                   i++) {

  //                                 if (state.cartItems[i]!.bayId ==

  //                                     widget.bayId) {

  //                                   print(state.cartItems);

  //                                   Map<String, dynamic> myMap = json.decode(

  //                                       state.cartItems[i]!.dataPart

  //                                           .toString());

  //                                   for (int k = 0;

  //                                       k < widget.values.length;

  //                                       k++) {

  //                                     for (int j = 0;

  //                                         j < myMap['data'].length;

  //                                         j++) {

  //                                       MapEntry<String, dynamic> entry =

  //                                           myMap['data'].entries.elementAt(j);

  //                                       String key = entry.key;

  //                                       dynamic value = entry.value;

  //                                       if (widget.values[k].parameterName

  //                                               .toString() ==

  //                                           key) {

  //                                         testController[k].text = value;

  //                                       }

  //                                     }

  //                                   }

  //                                 }

  //                               }

  // }

  @override

  Widget build(BuildContext context) {

    themeData = Theme.of(context);

    return BlocConsumer<FetchTempDataBloc, FetchTempDataState>(

      listener: (context, state) {},

      builder: (context, state) {

        return ExpansionPanelList(

            expandedHeaderPadding: const EdgeInsets.all(0),

            elevation: 0,

            expansionCallback: (int index, bool isExpanded) {

              basicDetails[index] = !isExpanded;

              isExpanded ? \_controller.forward() : \_controller.reverse();

              setState(() {});

            },

            animationDuration: const Duration(milliseconds: 1000),

            children: <ExpansionPanel>[

              ExpansionPanel(

                isExpanded: basicDetails[0],

                hasIcon: false,

                canTapOnHeader: true,

                headerBuilder: (BuildContext context, bool isExpanded) {

                  return Container(

                    decoration: const BoxDecoration(

                      color: Colors.white70,

                    ),

                    child: Column(

                        crossAxisAlignment: CrossAxisAlignment.start,

                        children: [

                          SizedBox(height: 1.h),

                          Row(

                            children: [

                              Container(

                                width: 30.w,

                                margin: EdgeInsets.only(left: 4.w),

                                child: Text(

                                  widget.bayTypeName,

                                  style: GoogleFonts.sourceSansPro(

                                    color: colorValue,

                                    fontWeight: FontWeight.w600,

                                    fontSize: 2.5.h,

                                  ),

                                  maxLines: 1,

                                  overflow: TextOverflow.ellipsis,

                                ),

                              ),

                              SizedBox(

                                width: 1.w,

                              ),

                              Container(

                                width: 40.w,

                                child: Text(

                                  "(${widget.bayName})",

                                  overflow: TextOverflow.ellipsis,

                                ),

                              ),

                              Expanded(child: Container()),

                              AnimatedBuilder(

                                animation: \_controller,

                                builder: (BuildContext context, Widget? child) {

                                  return Transform.rotate(

                                    angle: \_controller.value \* 0.25 \* math.pi,

                                    child: Icon(

                                      Icons.close,

                                      size: 2.3.h,

                                      color: Theme.of(context).primaryColor,

                                    ),

                                  );

                                },

                              ),

                              SizedBox(

                                width: 5.w,

                              )

                            ],

                          ),

                        ]),

                  );

                },

                body: Container(

                  decoration: BoxDecoration(color: Colors.white),

                  child: Column(

                    crossAxisAlignment: CrossAxisAlignment.start,

                    children: [

                      SizedBox(

                          height: 7.h,

                          child: ListView.builder(

                              scrollDirection: Axis.horizontal,

                              itemCount: widget.values.length,

                              itemBuilder: (context, index) {

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

                                  testController.add(TextEditingController());

                                  validationList.add(false);

                                }

                                return Form(

                                  child: Column(

                                    children: [

                                      SizedBox(

                                        height: 1.h,

                                      ),

                                      Container(

                                        alignment: Alignment.center,

                                        margin: EdgeInsets.only(left: 5.w),

                                        width: 30.w,

                                        height: 5.h,

                                        decoration: BoxDecoration(

                                            color: const Color(0xffF2F2F2),

                                            borderRadius:

                                                BorderRadius.circular(3)),

                                        child: TextFormField(

                                            initialValue: widget.dataMap[

                                                        widget.bayTypeName] ==

                                                    null

                                                ? ""

                                                : widget.dataMap[

                                                        widget.bayTypeName][

                                                    widget.values[index]

                                                        .parameterName],

                                            key: ObjectKey("key$index"),

                                            inputFormatters: [

                                              FilteringTextInputFormatter(

                                                  RegExp(r'[0-9+.]'),

                                                  allow: true)

                                            ],

                                            onChanged: (value) {

                                              if (insertedValues.containsKey(

                                                  widget.bayTypeName

                                                      .toString())) {

                                                insertedValues[

                                                    widget.bayTypeName

                                                        .toString()][widget

                                                    .values[index].parameterName

                                                    .toString()] = value;

                                              } else {

                                                insertedValues[widget

                                                    .bayTypeName

                                                    .toString()] = {};

                                                insertedValues[

                                                    widget.bayTypeName

                                                        .toString()][widget

                                                    .values[index].parameterName

                                                    .toString()] = value;

                                              }

                                              // widget.data[widget

                                              //     .values[index].parameterName

                                              //     .toString()] = value;

                                            },

                                            keyboardType: TextInputType.number,

                                            decoration: InputDecoration(

                                              hintText: widget

                                                  .values[index].parameterName,

                                              hintStyle: const TextStyle(

                                                  color: Colors.grey),

                                              fillColor:

                                                  const Color(0xffF2F2F2),

                                              enabledBorder: OutlineInputBorder(

                                                borderSide: BorderSide(

                                                    color: Theme.of(context)

                                                        .primaryColor),

                                              ),

                                              contentPadding: EdgeInsets.only(

                                                  top: 3.h, left: 10.w),

                                              focusedBorder: OutlineInputBorder(

                                                borderSide: BorderSide(

                                                    color: Theme.of(context)

                                                        .primaryColor),

                                              ),

                                            )),

                                      ),

                                    ],

                                  ),

                                );

                              })),

                      Row(

                        children: [

                          Expanded(child: Container()),

                          TextButton(

                              onPressed: () {},

                              child: Text(

                                "Cancel",

                                style: GoogleFonts.sourceSans3(

                                    color: Colors.black,

                                    fontWeight: FontWeight.w500,

                                    fontSize: 2.8.h),

                              )),

                          SizedBox(

                            width: 9.w,

                          ),

                          GestureDetector(

                            onTap: () async {

                              Map<String, dynamic> overAllMap = {};

                              overAllMap["bay\_type\_name"] = widget.bayTypeName;

                              overAllMap["bay\_name"] = widget.bayName;

                              overAllMap["data"] =

                                  insertedValues[widget.bayTypeName];

                              print(overAllMap);

                              var id =

                                  await BlocProvider.of<TempDataBloc>(context)

                                      .onInsertData(TempDataEvent(

                                          bayId: widget.bayTypeName,

                                          timeSlot: widget.timeSlot,

                                          data: overAllMap,

                                          frequencyId: widget.frequencyId,

                                          subStationId: widget.subStationId));

                              if (id > 0) {

                                Fluttertoast.showToast(

                                    toastLength: Toast.LENGTH\_SHORT,

                                    msg: "Successfully saved");

                              }

                            },

                            child: Container(

                              alignment: Alignment.center,

                              width: 18.w,

                              height: 5.5.h,

                              decoration: BoxDecoration(

                                  borderRadius: BorderRadius.circular(5),

                                  color: Theme.of(context).primaryColor),

                              child: Text(

                                "Save",

                                style: GoogleFonts.sourceSans3(

                                    color: Colors.white,

                                    fontWeight: FontWeight.w600,

                                    fontSize: 2.6.h),

                              ),

                            ),

                          ),

                          SizedBox(

                            width: 8.w,

                          )

                        ],

                      )

                    ],

                  ),

                ),

              ),

            ]);

      },

    );

  }

}

if (state is FetchEventsLoadedState) {

                return

                 ListView.builder(

                    itemCount: state.fetchEventsModel.success!.length,

                    itemBuilder: (context, index) {

                      return GestureDetector(

                        onTap: () {

                          Navigator.push(

                              context,

                              MaterialPageRoute(

                                  builder: (context) => EventDetailsScreen(

                                        mnpStatus: state.fetchEventsModel

                                            .success![index].isMpRequired

                                            .toString()

                                            .toLowerCase(),

                                        subStationId: widget.subStationId,

                                        eventId: state.fetchEventsModel

                                            .success![index].eventId

                                            .toString(),

                                        eventType: "BREAKDOWN",

                                      )));

                        },

                        child: Container(

                          width: 90.w,

                          height: 19.h,

                          decoration: BoxDecoration(

                            borderRadius: BorderRadius.circular(19),

                          ),

                          child: Card(

                            child: Container(

                              margin: EdgeInsets.only(left: 2.w),

                              child: Column(

                                crossAxisAlignment: CrossAxisAlignment.start,

                                children: [

                                  SizedBox(

                                    height: 1.h,

                                  ),

                                  Row(

                                    children: [

                                      Container(

                                        width: 44.2.w,

                                        margin: EdgeInsets.only(left: 3.w),

                                        child: Text(

                                          "Substation bay Name :",

                                          style: GoogleFonts.aBeeZee(

                                              color: Colors.black),

                                        ),

                                      ),

                                      Flexible(

                                        child: Text(

                                          state.fetchEventsModel.success![index]

                                              .substationbayname

                                              .toString(),

                                          style: GoogleFonts.aBeeZee(

                                              fontSize: 2.h,

                                              fontWeight: FontWeight.w500),

                                          overflow: TextOverflow.ellipsis,

                                        ),

                                      ),

                                    ],

                                  ),

                                  SizedBox(

                                    height: 0.9.h,

                                  ),

                                  Row(

                                    children: [

                                      Container(

                                        width: 44.2.w,

                                        margin: EdgeInsets.only(left: 3.w),

                                        child: Text(

                                          "Created On :",

                                          style: GoogleFonts.aBeeZee(

                                              color: Colors.black),

                                        ),

                                      ),

                                      Flexible(

                                        child: Text(

                                          state.fetchEventsModel.success![index]

                                              .created

                                              .toString().substring(0,16),

                                          style: GoogleFonts.aBeeZee(

                                              fontSize: 2.h,

                                              fontWeight: FontWeight.w500),

                                          overflow: TextOverflow.ellipsis,

                                        ),

                                      ),

                                    ],

                                  ),

                                  SizedBox(

                                    height: 0.9.h,

                                  ),

                                  Row(

                                    children: [

                                      Container(

                                        width: 44.2.w,

                                        margin: EdgeInsets.only(left: 3.w),

                                        child: Text(

                                          "Is Mnp Required :",

                                          style: GoogleFonts.aBeeZee(

                                              color: Colors.black),

                                        ),

                                      ),

                                      Flexible(

                                        child: Text(

                                          state.fetchEventsModel.success![index]

                                              .isMpRequired

                                              .toString(),

                                          style: GoogleFonts.aBeeZee(

                                              fontSize: 2.h,

                                              fontWeight: FontWeight.w500),

                                          overflow: TextOverflow.ellipsis,

                                        ),

                                      ),

                                    ],

                                  ),

                                  SizedBox(

                                    height: 0.9.h,

                                  ),

                                  Row(

                                    children: [

                                      Container(

                                        width: 44.2.w,

                                        margin: EdgeInsets.only(left: 3.w),

                                        child: Text(

                                          "Event Status :",

                                          style: GoogleFonts.aBeeZee(

                                              color: Colors.black),

                                        ),

                                      ),

                                      Flexible(

                                        child: Text(

                                          state.fetchEventsModel.success![index]

                                              .eventStatus

                                              .toString(),

                                          style: GoogleFonts.aBeeZee(

                                              fontSize: 2.h,

                                              fontWeight: FontWeight.w500),

                                          overflow: TextOverflow.ellipsis,

                                        ),

                                      ),

                                    ],

                                  ),

                                  SizedBox(

                                    height: 0.9.h,

                                  ),

                                  Row(

                                    children: [

                                      Container(

                                        width: 44.2.w,

                                        margin: EdgeInsets.only(left: 3.w),

                                        child: Text(

                                          "Reason :",

                                          style: GoogleFonts.aBeeZee(

                                              color: Colors.black),

                                        ),

                                      ),

                                      Flexible(

                                        child: Text(

                                          state.fetchEventsModel.success![index]

                                              .reason

                                              .toString(),

                                          style: GoogleFonts.aBeeZee(

                                              fontSize: 2.h,

                                              fontWeight: FontWeight.w500),

                                          overflow: TextOverflow.ellipsis,

                                        ),

                                      ),

                                    ],

                                  ),

                                ],

                              ),

                            ),

                          ),

                        ),

                      );

                    });

              }