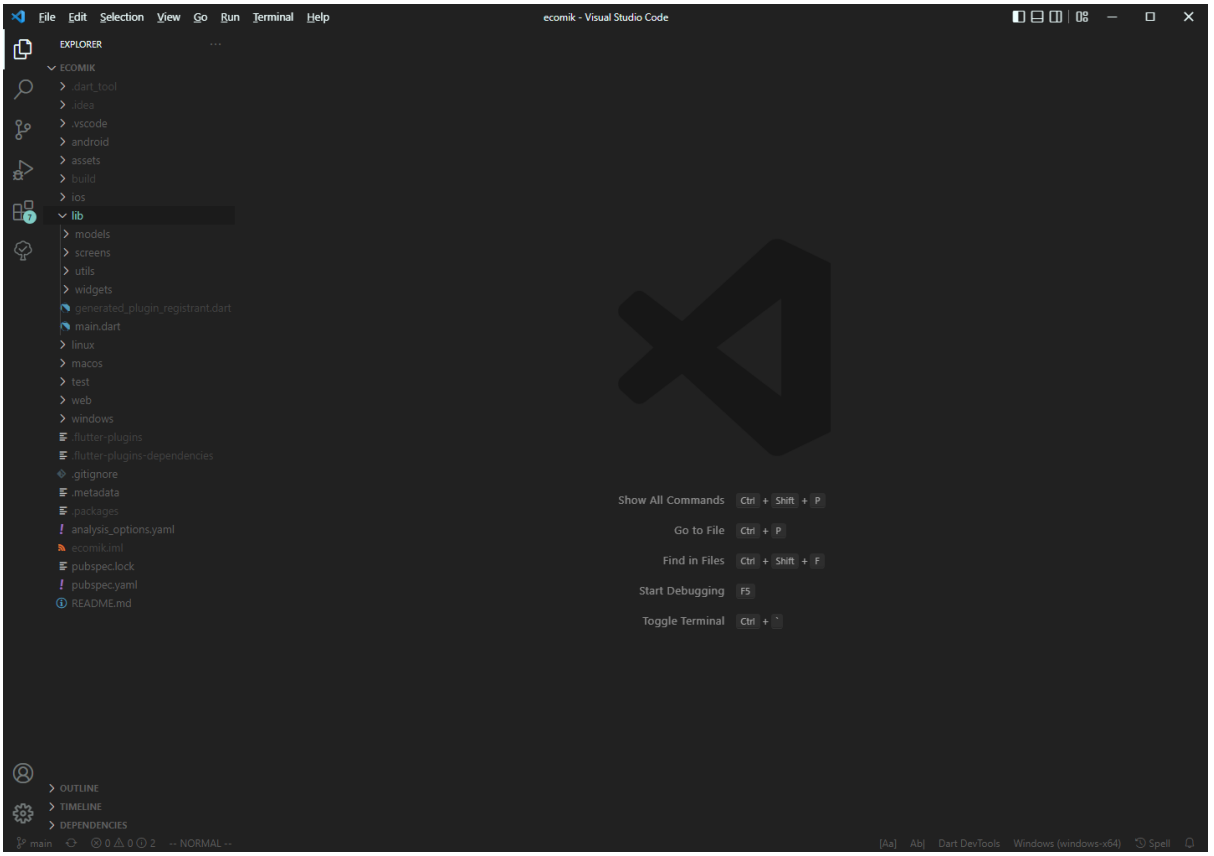
Hello everyone!! 👋👋

We implemented a clean folder structure here. We used “flutter_svg” for svg icons, “fl_chart” for graphs, “badges” for badges, i.e notification count, “dotted_border” for containers that have dotted borders, “flutter_slidable” for slidable option for a widget, “pinput” for OTP text field, “smooth_page_indicator” for circle page indicator, “flutter_switch” for custom toggle button, “sliding_up_panel” for persistent bottom sliding up panel.

Setting up

---

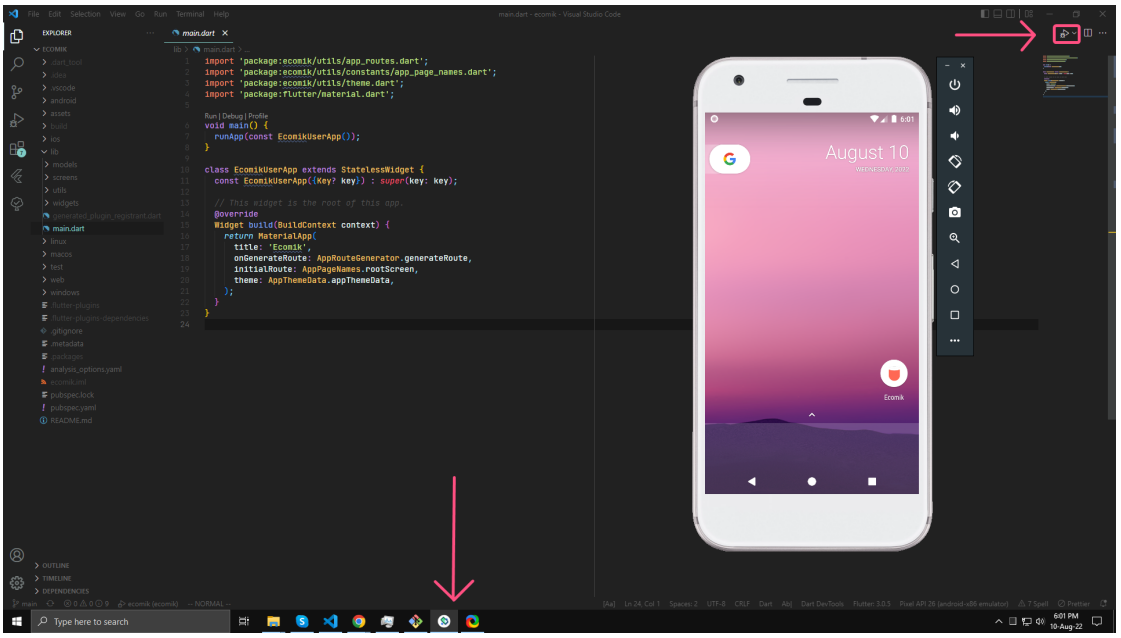
1. Follow the Flutter installation guide here.
2. After setting up the flutter environment, extract the sourcecode.zip.
3. Open your respected code editor (VSCode, Android Studio, etc).
4. Open the folder from your editor.

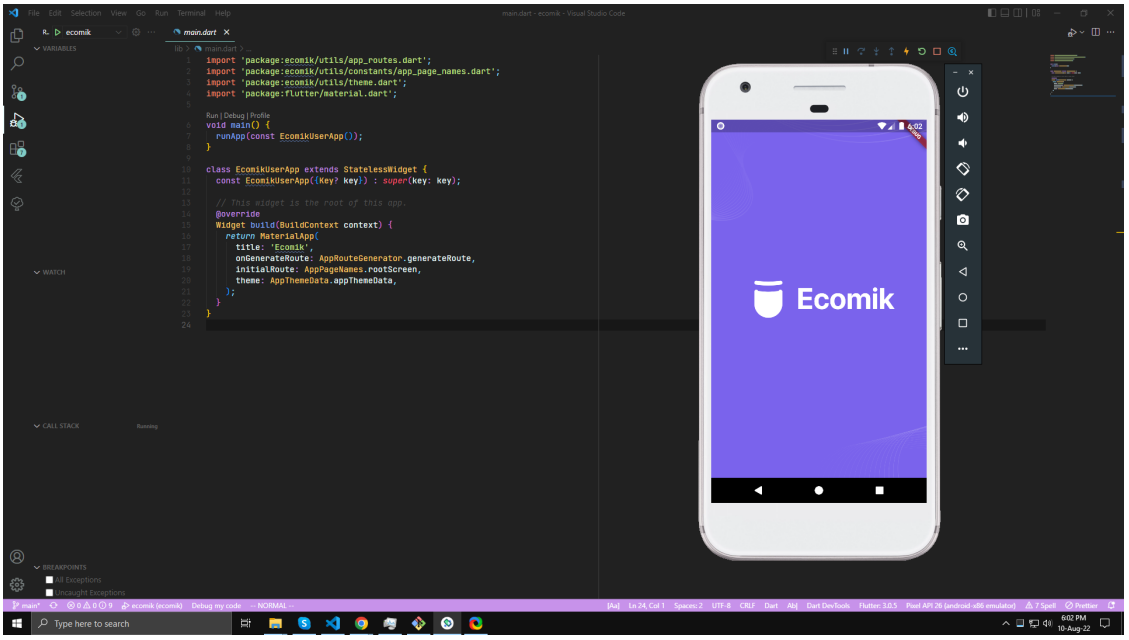


5. You will see something like this.
6. If you see some error, run " `flutter clean` ", then run " `flutter pub get` " on the terminal and finally, restart the code editor.
7. If you see any other errors after this, please contact us.

Running

1. Open and check if your emulator is detected by your code editor.
2. Then press the play button. It will run the app to the emulator.

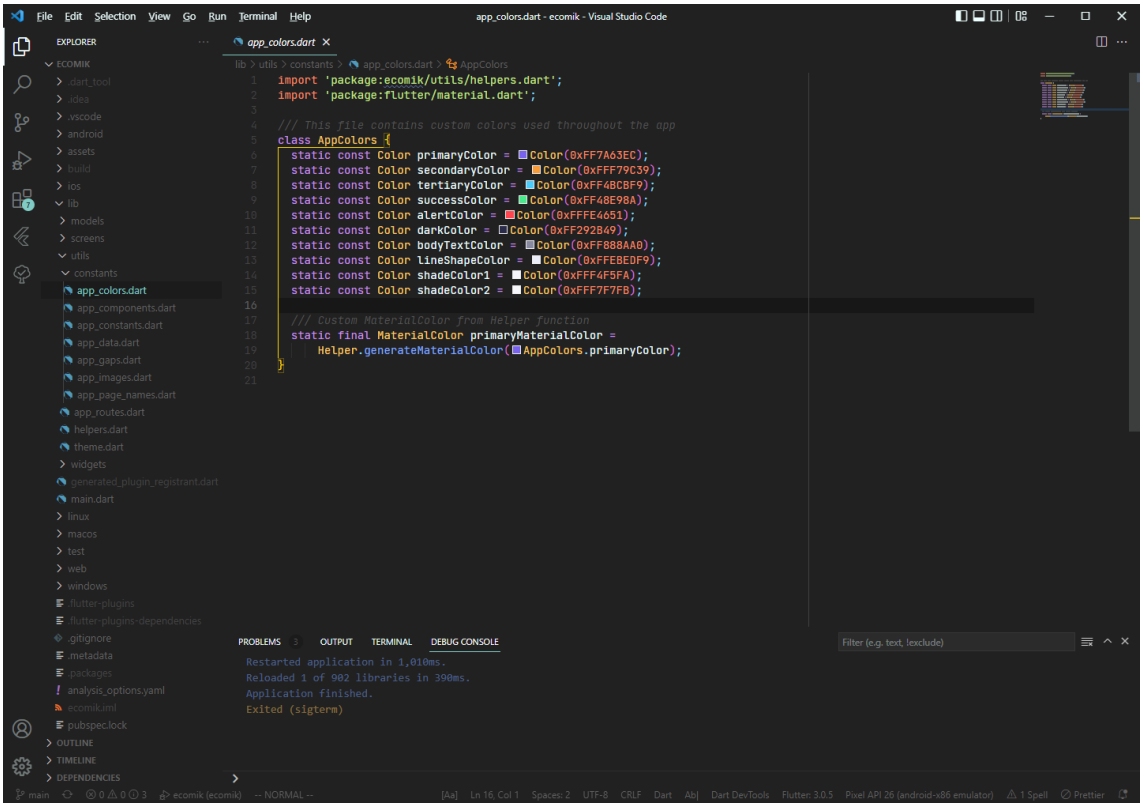




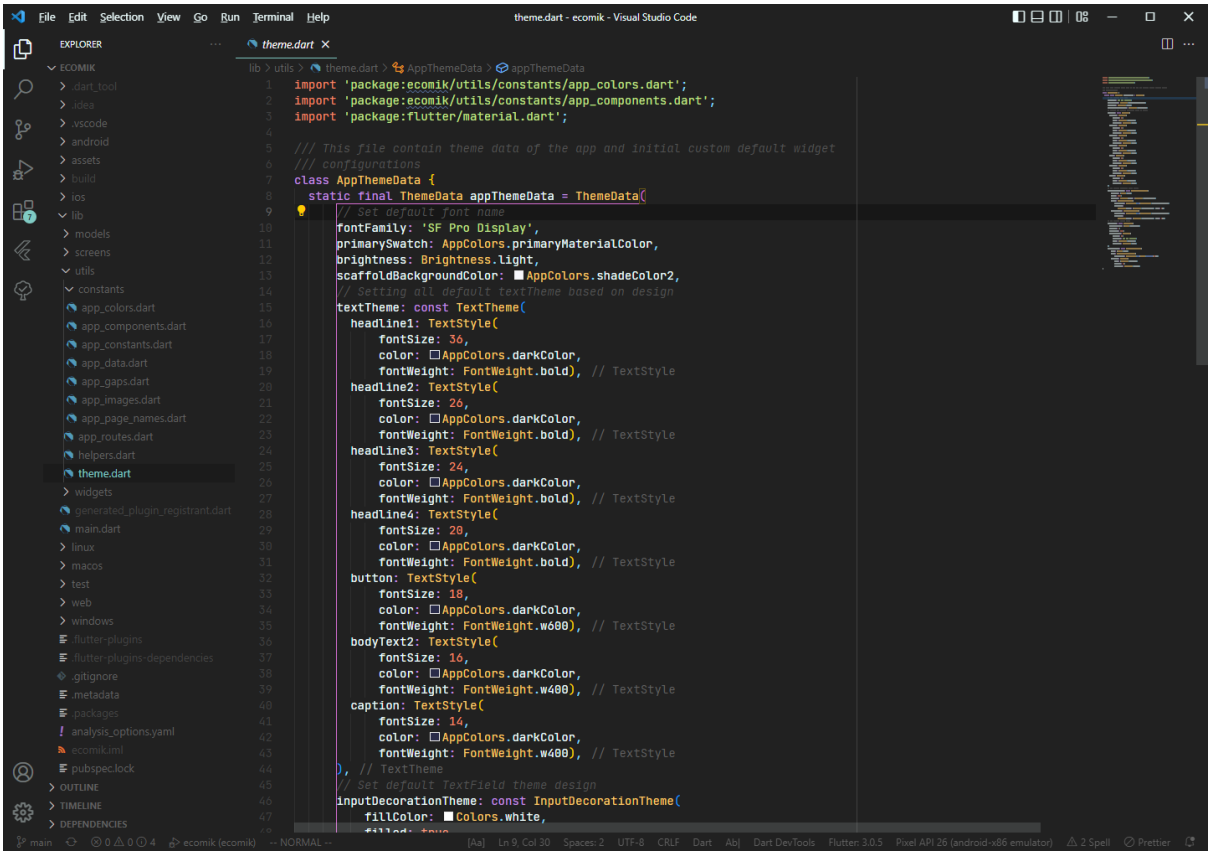
Customization

Change the primary color:

- Replace the primary color inside AppColors class (lib/utils/constants/app_colors.dart) with your brand color.

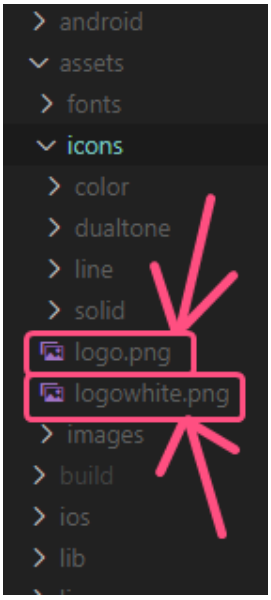


- You can change any other colors as you wish in this file.
- This app has a theme. You can find the theme in lib/utils/theme.dart
- You can change the default font name and style from here or use the default flutter font.




```
        color: AppColors.darkColor,
        fontWeight: FontWeight.w400), // TextStyle
    ), // TextTheme
    // Set default TextField theme design
    inputDecorationTheme: const InputDecorationTheme(
        fillColor: Colors.white,
        filled: true,
        hintStyle: TextStyle(color: AppColors.bodyTextColor),
        border: OutlineInputBorder(
            borderRadius: BorderRadius.all(AppComponents.defaultBorderRadius),
            borderSide:
                BorderSide(color: AppColors.lineShapeColor, width: 1)), // OutlineInputBorder
        enabledBorder: OutlineInputBorder(
            borderRadius: BorderRadius.all(AppComponents.defaultBorderRadius),
            borderSide:
                BorderSide(color: AppColors.lineShapeColor, width: 1))), // OutlineInputBorder // InputDecorat
    // Set default appbar theme
    appBarTheme: const AppBarTheme(
      backgroundColor: Colors.transparent,
      elevation: 0,
      centerTitle: true,
      titleTextStyle: TextStyle(
          fontSize: 24,
          fontFamily: 'SF Pro Display',
          color: AppColors.darkColor,
          fontWeight: FontWeight.bold), // TextStyle
    ), // AppBarTheme
    popupMenuTheme: const PopupMenuThemeData(
        color: Colors.white,
        shape: RoundedRectangleBorder(
            borderRadius: BorderRadius.all(Radius.circular(14))), // RoundedRectangleBorder
        textStyle: TextStyle(
            color: AppColors.darkColor,
            fontSize: 14,
            fontWeight: FontWeight.w500))); // TextStyle // PopupMenuThemeData // ThemeData
}
```

Changing the app logo



In the assets/icons/logo.png, replace the logo.png with your own logo image with the file name of "logo.png", the name should be exactly the same.

Do the same with the assets/icons/logowhite.png, replace the logowhite.png with your own logo in white color, with the file name of "logowhite.png".

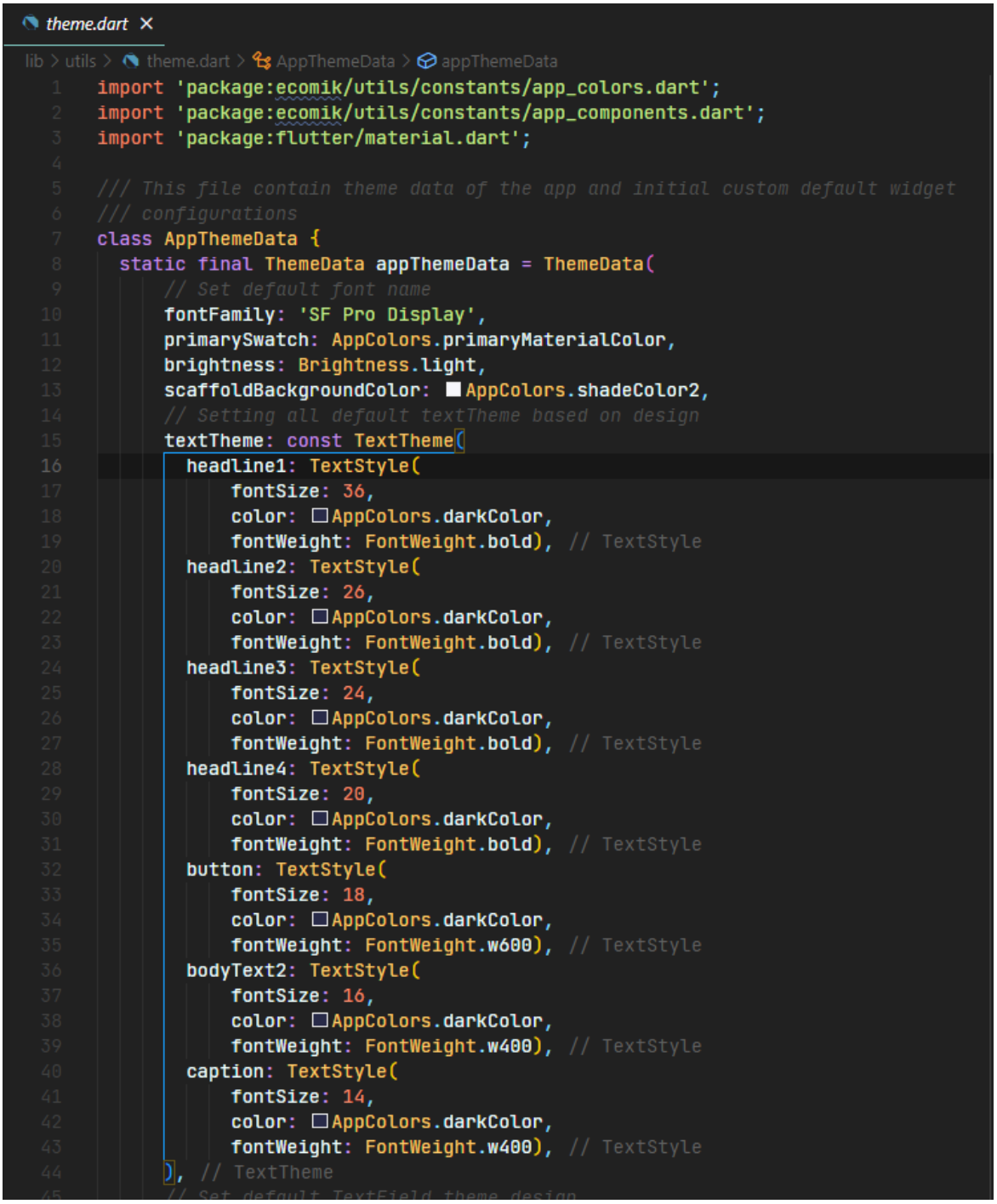
Then run this command on your terminal " `flutter pub get` "

And then run " `flutter pub run flutter_launcher_icons:main` "

For detailed instructions, follow this guide.

Textstyles

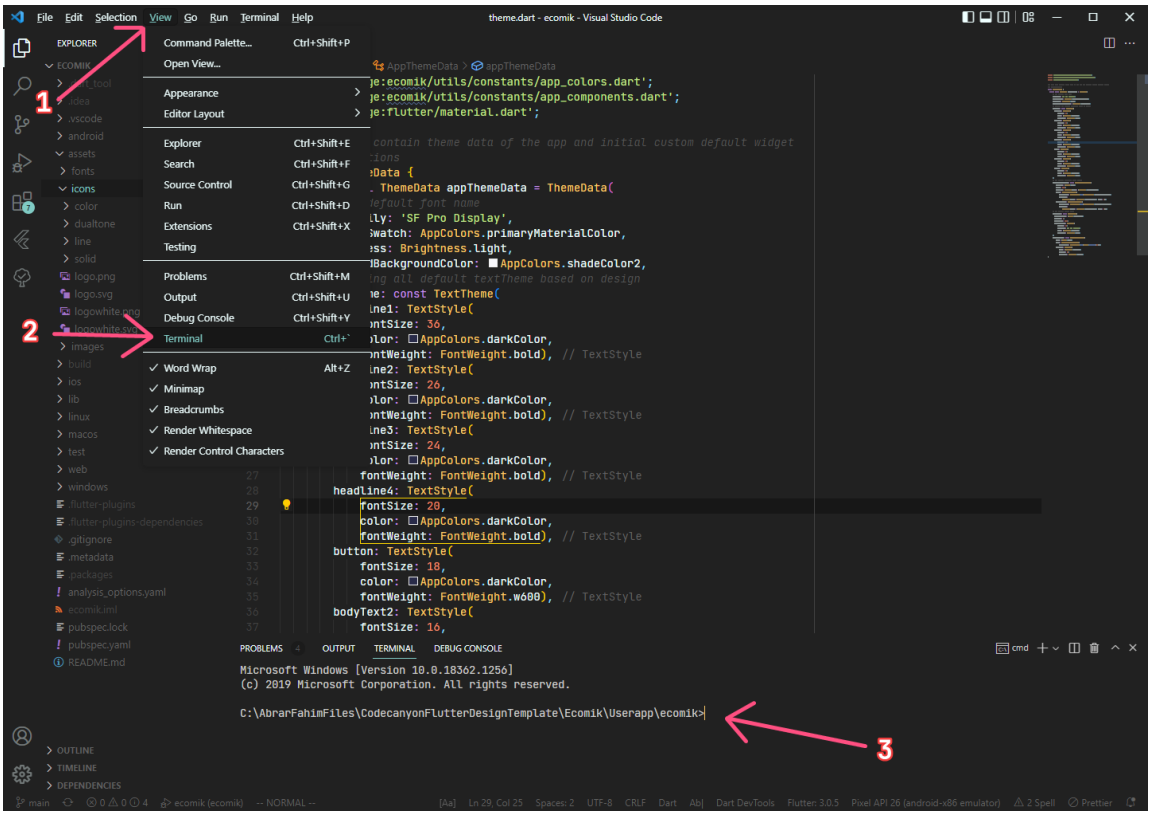
We are modifying the default TextTheme of flutter inside the theme to use textStyles. You can find the text theme inside of the “lib/utils/theme.dart“ file, the appThemeData class contains all the Text themes and more theme properties.



Renaming the package and the app

---

1. If you installed flutter this step is done, or follow this,
2. Run this command on your terminal " `flutter pub global activate rename` "
3. Then go to the projects root directory, where the pubspec.yaml is
4. Run this command to change the package name " `flutter pub global run rename --bundleId com.yourpackage.app` "
5. Run this command to change the App name " `flutter pub global run rename --appname "Ecomik two"` "
6. You can follow the more detailed instructions here.
7. If you are unsure where to run the commands, you can do it here

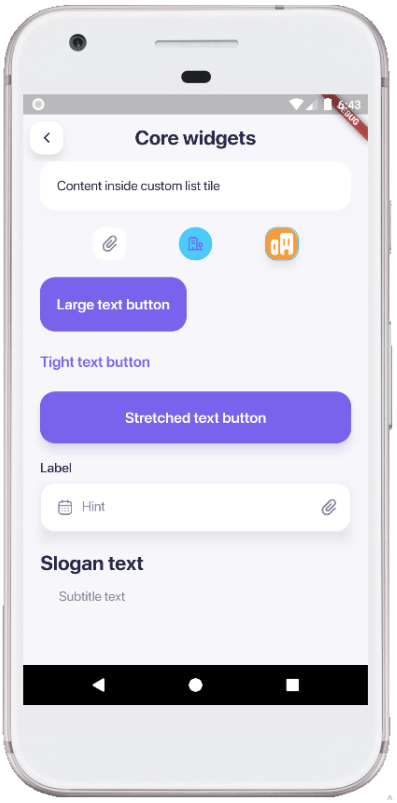


The custom widgets

You can find the custom widgets in the lib/widgets/ folder.
There are core widgets and screen widgets inside the widgets folder. These widgets serve their purpose across all the three apps.

Core widgets

There are a lot of core widgets used throughout the project, defined inside the lib/widgets/core_widgets.dart file.



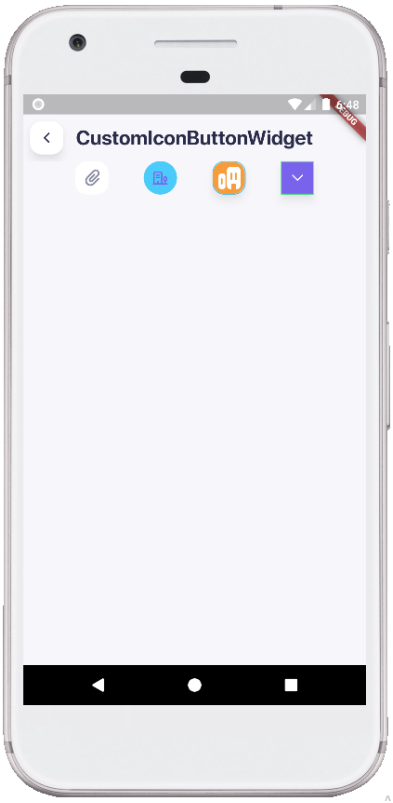
Custom buttons

There are several button widgets used here. These are the types of buttons used- Icon button, stretched button, text button.

CustomIconButtonWidget

Attributes:

```dart
/// Custom IconButton widget various attributes
class CustomIconButtonWidget extends StatelessWidget {
  final void Function()? onTap;
  final Border? border;
  final Widget child;
  final Color backgroundColor;
  final Size fixedSize;
  final Radius borderRadiusRadiusValue;
  final bool isCircleShape;
  final bool hasShadow;
  const CustomIconButtonWidget(
      {Key? key,
      this.onTap,
      required this.child,
      this.backgroundColor = Colors.white,
      this.fixedSize = const Size(40, 40),
      this.borderRadiusRadiusValue = const Radius.circular(14),
      this.border,
      this.isCircleShape = false,
      this.hasShadow = false})
      : super(key: key);
```

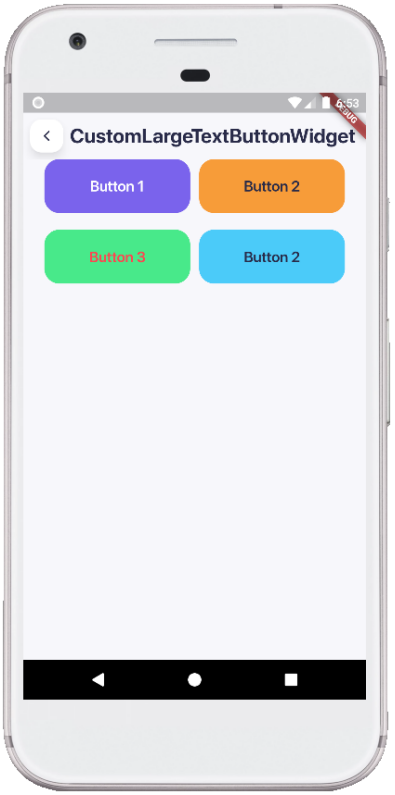


This button is used as an icon button.
You can resize the button height and width and can be set to a circular icon button.

CustomLargeTextButtonWidget

Attributes:

```dart
/// Custom large text button widget
class CustomLargeTextButtonWidget extends StatelessWidget {
  final bool isSmallScreen;
  final void Function()? onTap;
  final String text;
  final Color backgroundColor;
  final Color textColor;
  const CustomLargeTextButtonWidget({
    Key? key,
    this.onTap,
    required this.text,
    this.backgroundColor = AppColors.primaryColor,
    this.textColor = Colors.white,
    this.isSmallScreen = false,
  }) : super(key: key);
```

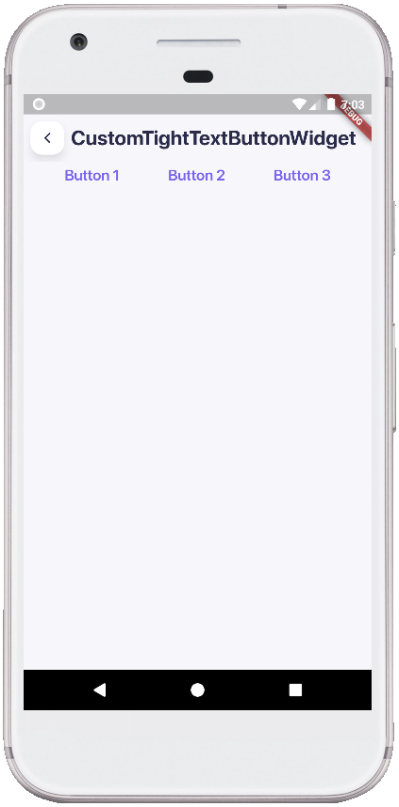


This text button uses a large padding around the text.

CustomTightTextButtonWidget

Attributes:

```dart
class CustomStretchedButtonWidget extends StatelessWidget {
  final Widget child;
  final Size buttonFixedSize;
  final Color backgroundColor;
  final Function()? onTap;
  const CustomStretchedButtonWidget({
    Key? key,
    required this.child,
    this.buttonFixedSize = const Size(50, 55),
    this.onTap,
    this.backgroundColor = AppColors.primaryColor,
  }) : super(key: key);
```



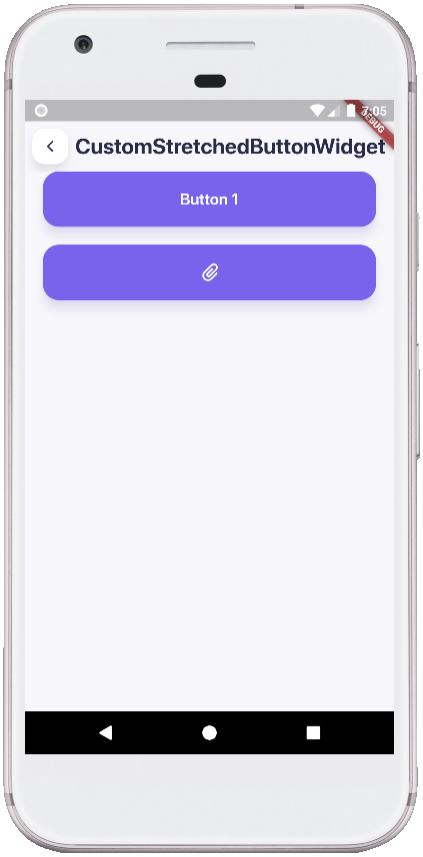
This text button does not have any background color and the button size is exactly as the text takes spaces.

CustomStretchedButtonWidget and CustomStretchedTextButtonWidget

Attributes:

```dart
/// Custom TextButton stretches the width of the screen with small elevation
/// shadow with custom child widget
class CustomStretchedButtonWidget extends StatelessWidget {
  final Widget child;
  final void Function()? onTap;
  const CustomStretchedButtonWidget({
    Key? key,
    this.onTap,
    required this.child,
  }) : super(key: key);
```

```dart
/// Custom TextButton stretches the width of the screen with small elevation
/// shadow
class CustomStretchedTextButtonWidget extends StatelessWidget {
  final String buttonText;
  final void Function()? onTap;
  const CustomStretchedTextButtonWidget({
    Key? key,
    this.onTap,
    required this.buttonText,
  }) : super(key: key);
```

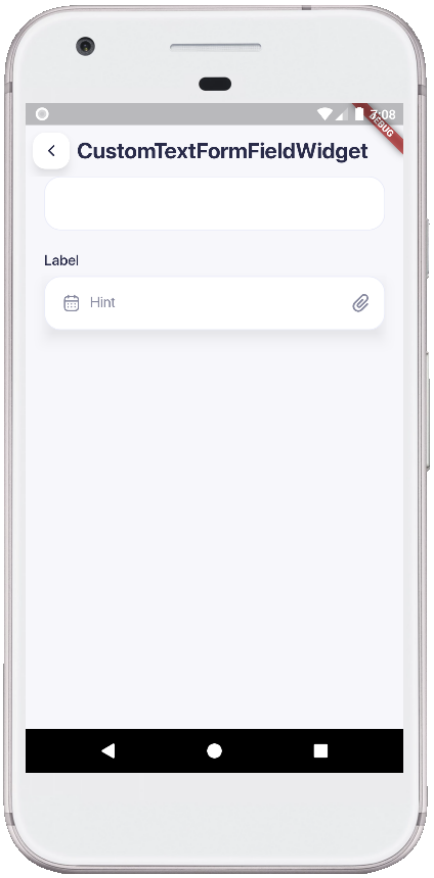


These button's width stretched across the screen. Their difference is the CustomStretchedButtonWidget needs to provide a child widget while CustomStretchedTextButtonWidget needs to provide the button text attribute to settext to the button.

CustomTextFormFieldWidget

Attributes:

```dart
/// Custom TextFormField configured with Theme.
class CustomTextFormField extends StatelessWidget {
  final String? labelText;
  final String? hintText;
  final Widget? prefixIcon;
  final Widget? suffixIcon;
  final bool isPasswordTextField;
  final bool hasShadow;
  final bool isReadOnly;
  final BoxConstraints prefixIconConstraints;
  final TextInputType? textInputType;
  final BoxConstraints suffixIconConstraints;
  final TextEditingController? controller;
  final int? minLines;
  final int maxLines;
  final void Function()? onTap;
  const CustomTextFormField({
    Key? key,
    this.labelText,
    this.prefixIcon,
    this.suffixIcon,
    this.hintText,
    this.isPasswordTextField = false,
    this.hasShadow = false,
    this.prefixIconConstraints =
        const BoxConstraints(maxHeight: 48, maxWidth: 48),
    this.suffixIconConstraints =
        const BoxConstraints(maxHeight: 48, maxWidth: 48),
    this.isReadOnly = false,
    this.textInputType,
    this.controller,
    this.minLines,
    this.maxLines = 1,
    this.onTap,
  }) : super(key: key);
```



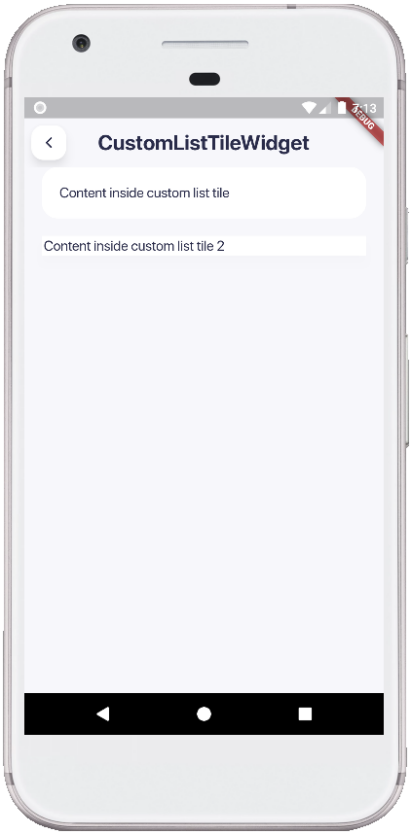
- There are a lot of attributes used for customizing this widget.
- Most of the attributes are optional and maintain the intended text field design flow.

This custom text field widget is used for various use cases as it offers a lot of attributes to control.

Custom list tile

Attributes:

```dart
class CustomListTileWidget extends StatelessWidget {
  final bool hasShadow;
  final double elevation;
  final void Function()? onTap;
  final void Function()? onLongPress;
  final Widget child;
  final EdgeInsets paddingValue;
  final BorderRadius borderRadius;
  const CustomListTileWidget(
      {Key? key,
      required this.child,
      this.onTap,
      this.paddingValue = const EdgeInsets.all(AppGaps.screenPaddingValue),
      this.onLongPress,
      this.borderRadius =
          const BorderRadius.all(AppComponents.defaultBorderRadius),
      this.hasShadow = false,
      this.elevation = 10})
      : super(key: key);
```

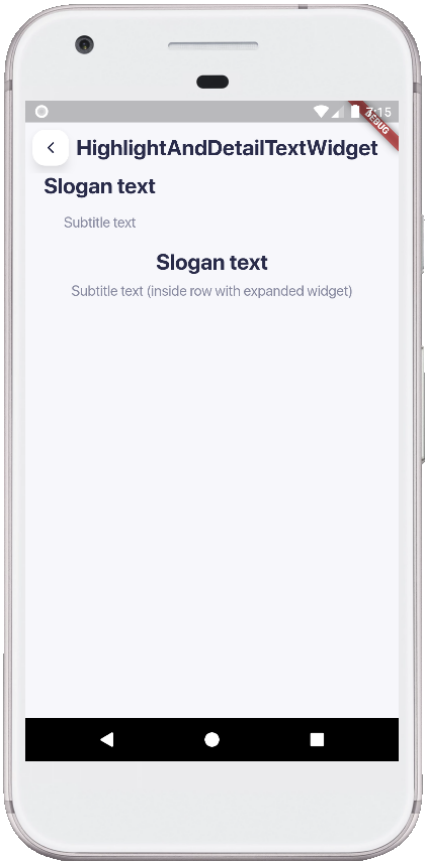


One of the major building block widget used throughout the app. It can act as a button.

HighlightAndDetailTextWidget

Attributes:

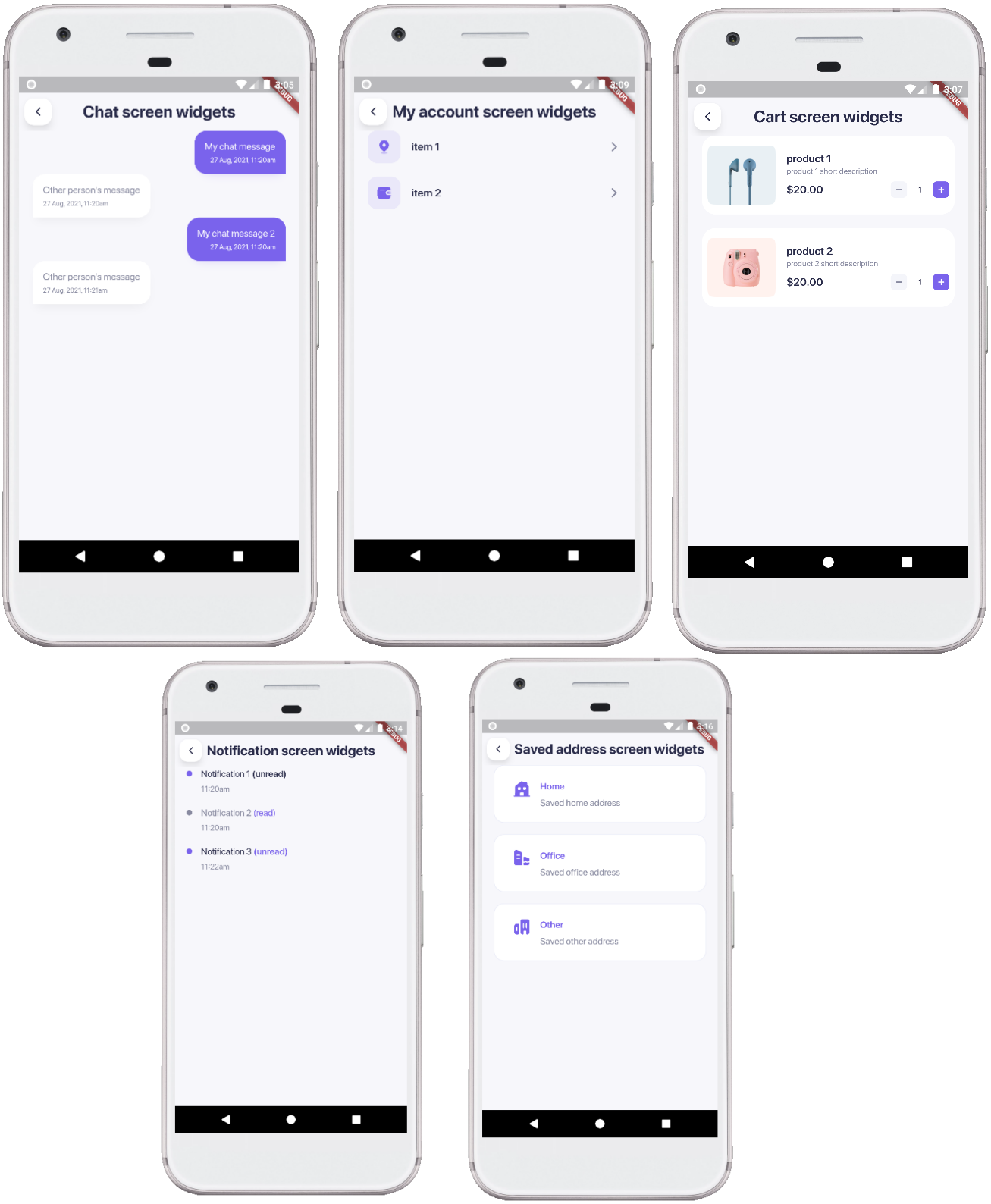
```dart
/// Custom large text button widget
class CustomLargeTextButtonWidget extends StatelessWidget {
  final bool isSmallScreen;
  final void Function()? onTap;
  final String text;
  final Color backgroundColor;
  final Color textColor;
  const CustomLargeTextButtonWidget({
    Key? key,
    this.onTap,
    required this.text,
    this.backgroundColor = AppColors.primaryColor,
    this.textColor = Colors.white,
    this.isSmallScreen = false,
  }) : super(key: key);
```



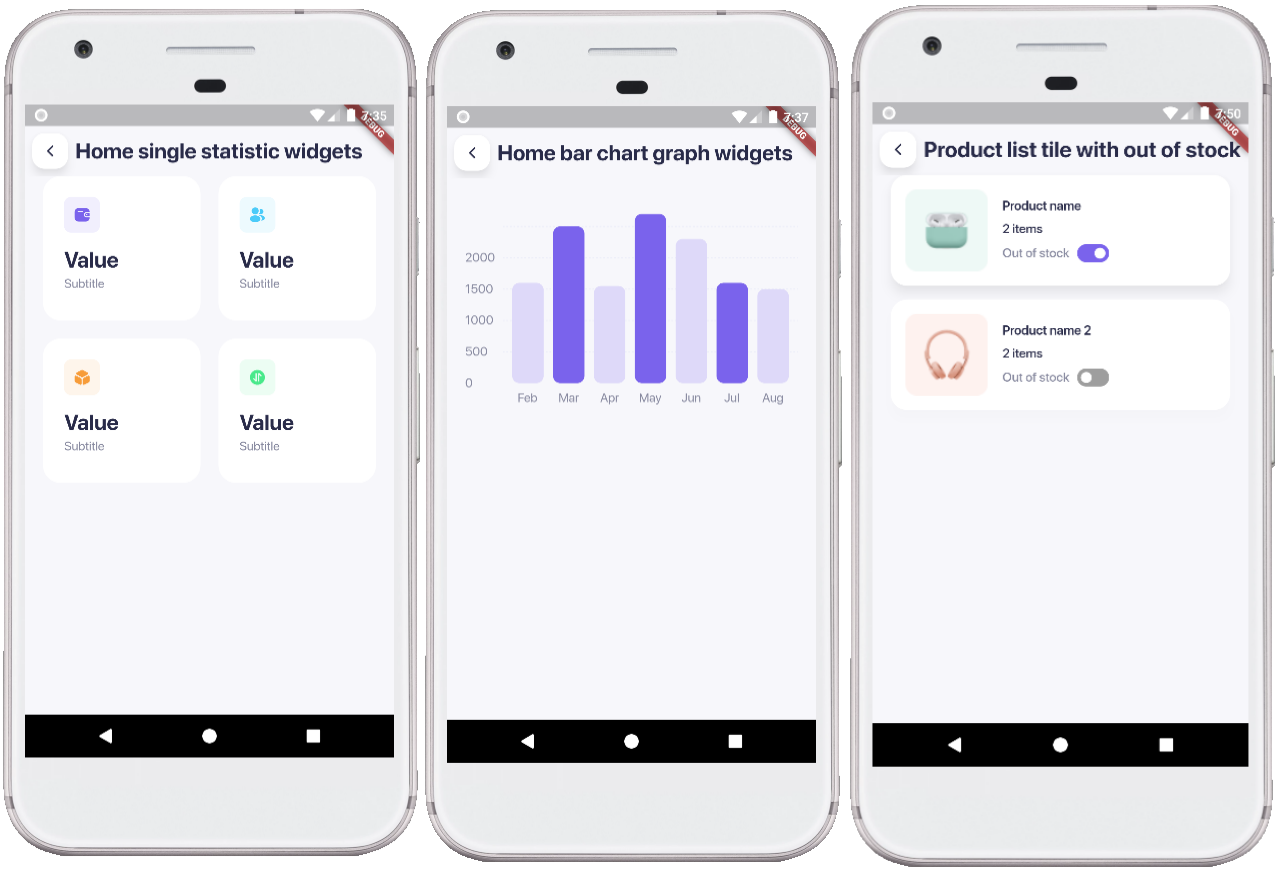
This widget shows highlighted slogan text and subtitle text.

Screen widgets

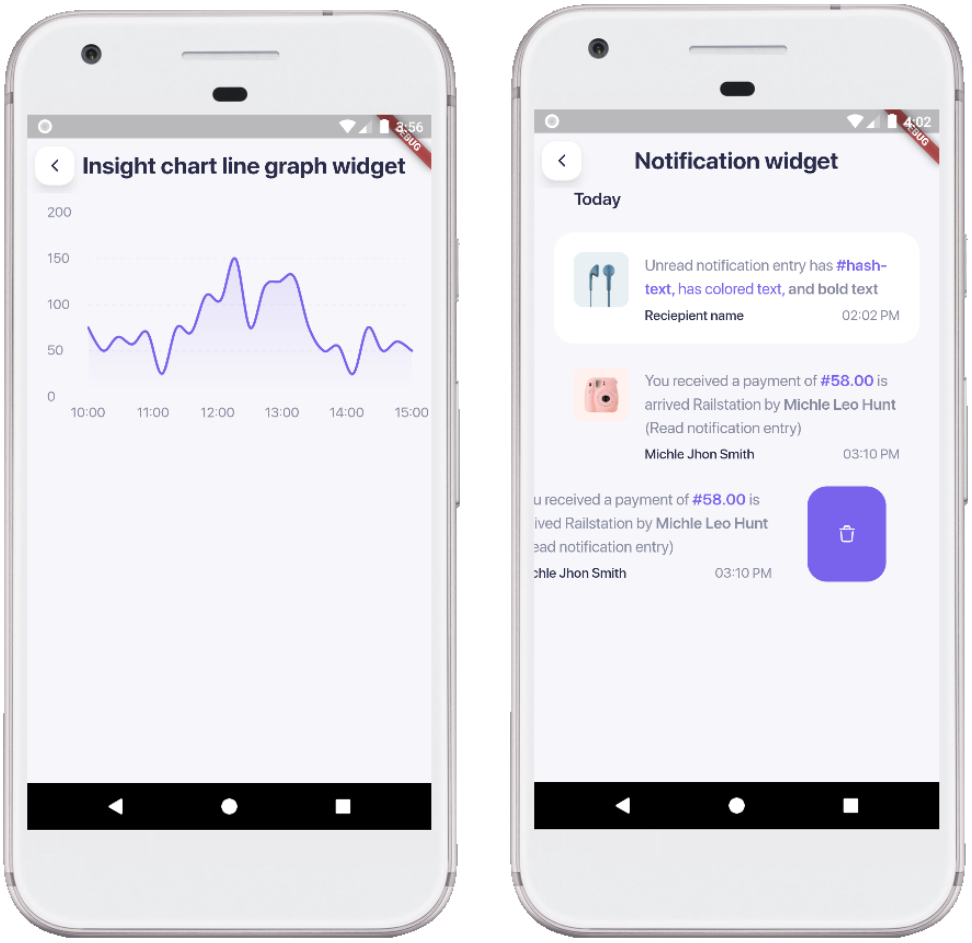
User app



Store app


Home single statistic widgets
Value
Subtitle
Value
Subtitle
Value
Subtitle
Value
Subtitle
Home bar chart graph widgets
2000
1500
1000
500
0
Feb
Mar
Apr
May
Jun
Jul
Aug
Product list tile with out of stock
Product name
2 items
Out of stock
Product name 2
2 items
Out of stock

Delivery app



The modular widgets of particular screens defined inside of this folder inside per app project.

THANK YOU

If you face any issues, please contact us in support.
We will try our best to help you out.