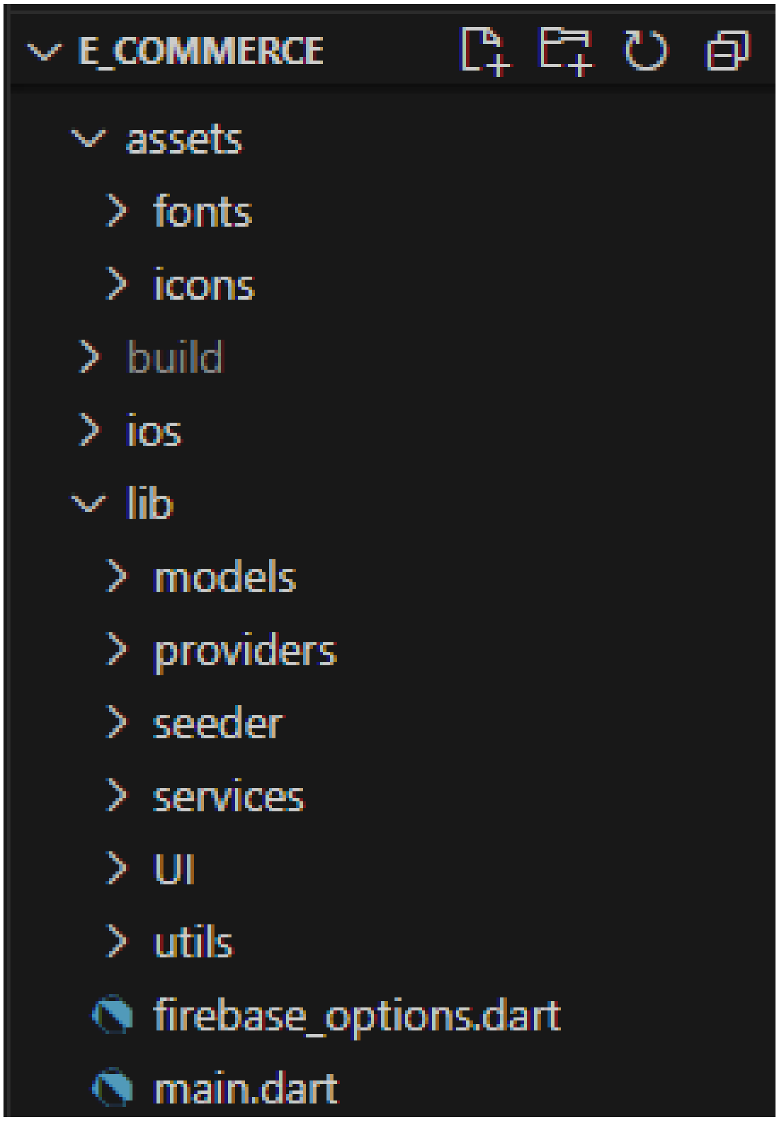
Project Report

Project structure



```
E_COMMERCE
  assets
    fonts
    icons
  build
  ios
  lib
    models
    providers
    seeder
    services
    UI
    utils
    firebase_options.dart
    main.dart
```

Models' folder: contains modeling files of dart dojects

Providers folder: contains Providers file

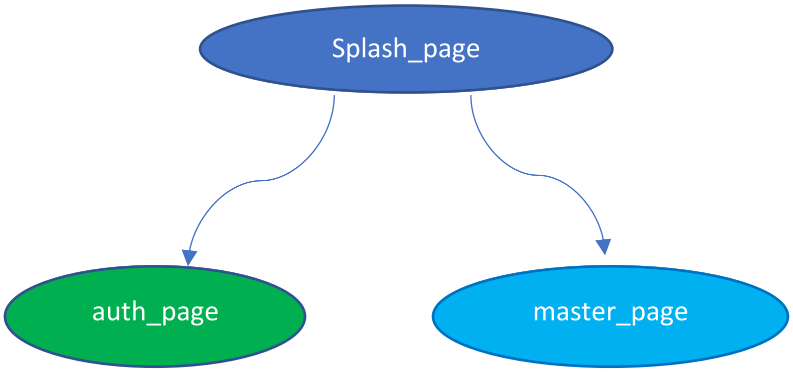
Seeders folder: contains widget data seeder file

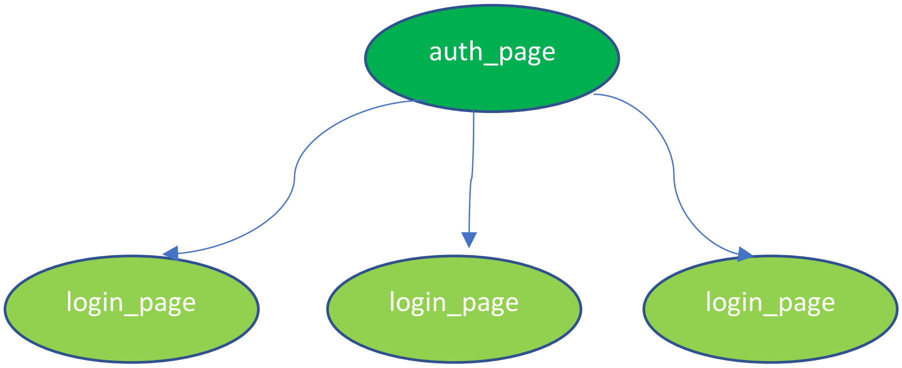
Service folder: contains notification Service file

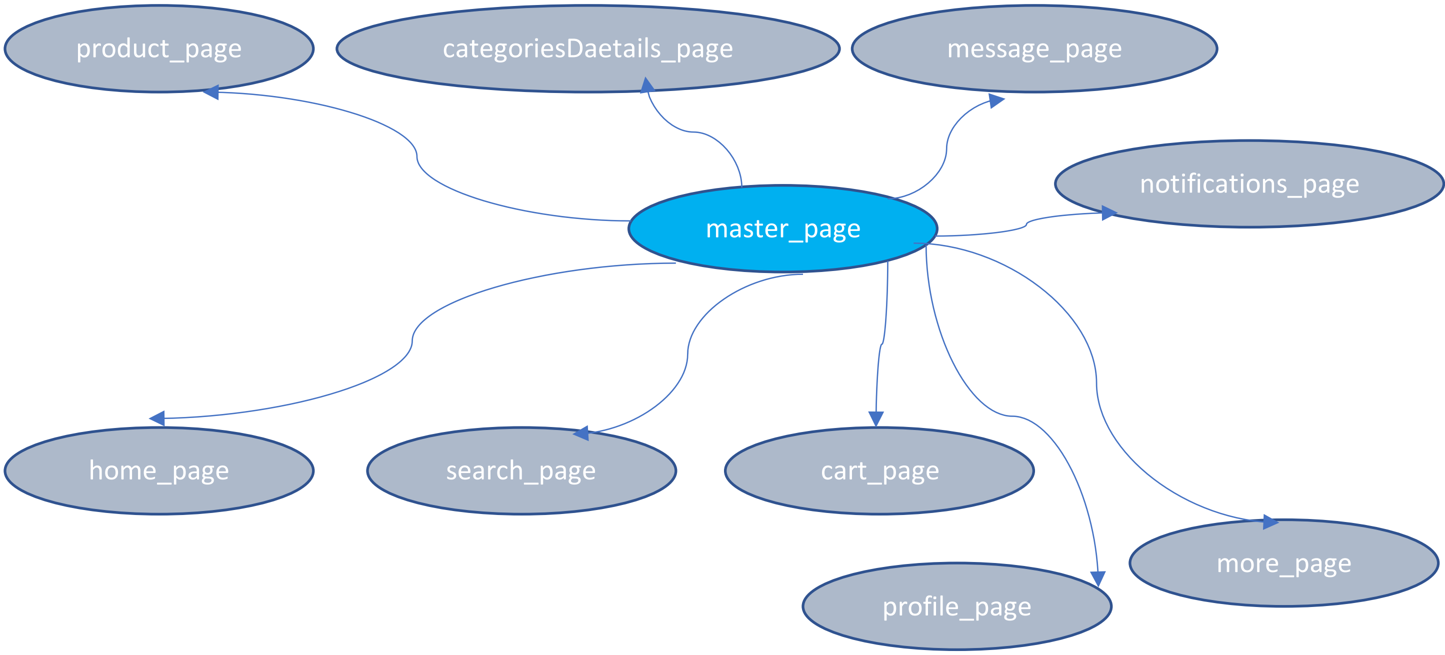
UI folder: user interface folder which contains page folder and widget folder

Utils folder contains the utilities (authenum,color,theme,collectionenum)

Pages Tree


Splash_page
auth_page
master_page


auth_page
login_page
login_page
login_page


product_page
categoriesDaetails_page
message_page
notifications_page
master_page
home_page
search_page
cart_page
more_page
profile_page

Used Packages:

shared_preferences: ^2.2.2

Wraps platform-specific persistent storage for simple data (NSUserDefaults on iOS and macOS, SharedPreferences on Android, etc.). Data may be persisted to disk asynchronously, and there is no guarantee that writes will be persisted to disk after returning, so this plugin must not be used for storing critical data.

In flutter are used to store data locally in key-value pairs in android and ios apps. We might have to save data locally while developing applications. Data like login credentials or a flag etc. Sharedpreferences comes in handy in such situations. For example, we have to save login credentials or a flag locally when the user logs in for the first time. This data will help the user access his account without logging in every time the app launches.

email_validator: ^2.1.17

A simple (but correct) Dart class for validating email addresses without using RegEx. Can also be used to validate emails within Flutter apps

carousel_slider: ^4.2.1

A carousel slider widget, support infinite scroll and custom child widget

dots_indicator: ^3.0.0

Dots indicator to show progression of a PageView for example

animated_bottom_navigation_bar: ^1.3.0

BottomNavigationBar is a widget that displays a row of small widgets at the bottom of a Flutter app. Usually, it’s used to show around three to five items. Each item must have a label and an icon. BottomNavigationBar allows you to select one item at a time and quickly navigate to a given page

Line_icons: ^2.0.3

Another icon library, based on Awesome Line Icons (by www.icons8.com) and with the help of FlutterIcon.com

get_it: ^7.6.4

This is a simple Service Locator for Dart and Flutter projects with some additional goodies highly inspired by Splat. It can be used instead of Inherited Widget or Provider to access objects e.g. from your UI

provider: ^6.1.1
Most Famous Flutter Package, it is a mix between State Management and Dependency Injection

- Separates your state from your UI
- Manages rebuilding UI based on state changes
- To share information between the widget tree

By using provider instead of manually writing Inherited Widget, you get:

- simplified allocation/disposal of resources
- lazy-loading
- a vastly reduced boilerplate over making a new class every time
- devtool friendly – using Provider, the state of your application will be visible in the Flutter devtool
- a common way to consume these Inherited Wid
- increased scalability for classes with a listening mechanism that grows exponentially in complexity

flutter_screenutil: ^5.9.0
A flutter plugin for adapting screen and font size.Guaranteed to look good on different models

firebase_core: ^2.24.2
A Flutter plugin to use the Firebase Core API, which enables connecting to multiple Firebase apps.

firebase_auth: ^4.15.3
Flutter plugin for Firebase Auth, enabling authentication using passwords, phone numbers and identity providers like Google, Facebook and Twitter.

cloud_firestore: ^4.13.6
Flutter plugin for Cloud Firestore, a cloud-hosted, noSQL database with live synchronization and offline support on Android and iOS.

firebase_storage: ^11.5.6

Flutter plugin for Firebase Cloud Storage, a powerful, simple, and cost-effective object storage service for Android and iOS.

firebase_messaging: ^14.7.10
Flutter plugin for Firebase Cloud Messaging, a cross-platform messaging solution that lets you reliably deliver messages on Android and iOS.

quickalert: ^1.0.2
With QuickAlert, instantly display animated alert dialogs such as success, error, warning, confirm, loading or even a custom dialog.

file_picker: ^6.1.1
A package that allows you to use a native file explorer to pick single or multiple absolute file paths, with extension filtering support.

cached_network_image: ^3.3.0
A flutter library to show images from the internet and keep them in the cache directory.

flexible_grid_view: ^0.0.2
Flexible grid view is a Flutter package that simplifies the process of creating a grid view. With FlexibleGridView, you don't need to specify the height of your children widgets.

uuid: ^4.2.2

Simple, fast generation of RFC4122 UUIDs.

Features:

- Generate RFC4122 version 1, version 4, or version 5 UUIDs
- Supports draft version 6, version 7, and version 8
- Runs in web, server, and flutter

order_tracker: ^0.0.2
A Flutter plugin for iOS and Android for order tracker same as flipkart order tracker ui.

overlay_support: ^2.1.0

Provider support for overlay, make it easy to build toast and In-App notification.this library support ALL platform

Used Flutter Widgets

ValueListenableBuilder Widget

ValueListenableBuilderwidget. It is an amazing widget. It builds the widget every time the valueListenable value changes. Its values remain synced with there listeners i.e. whenever the values change the ValueListenable listen to it. It updates the UI without using setState() or any other state management technique

Page Viewer Widget

A PageView allows the user to swipe between different screens in your app. All you need to set it up are a PageViewController and a PageView

LayoutBuilder Widget

The LayoutBuilderwidget allows you to build different UI layouts based on the parent widget's constraints. You can use this to customize the UI based on available width and height.

Positioned and Stack Widgets

These allow for precise positioning. The positioning will adapt to the container's dimensions, making it responsive to different screen sizes.

Consumer widget

Once our designed Model is provided to the desired widgets in our app through the ChangeNotifierProvider declaration at the top, the Consumer widgets subscribed to the notifications can use it

TabBar Widget

Apps often have different categories of content or features available to users. Naturally, you want your users to be able to quickly glance at each option and move between categories with a simple swipe. That’s where the Flutter TabBar class comes in

MediaQuery Widget

the MediaQuerywidget plays a pivotal role in responsive design. It provides access to the device's dimensions, orientation, and other characteristics. By wrapping relevant parts of the UI in a MediaQuery widget, developers can obtain contextual information and make decisions about how to adjust the app's layout and components accordingly.

FutureBuilder Widget

FutureBuilderis a widget that uses the result of a Future to build itself.

StreamBuilder widget

An asynchronous process may need some time to finish. Sometimes, there can be some values emitted before the process finishes. In Dart, you can create a function that returns a Stream, which can emit some values while the asynchronous process is active. If you want to build a widget in Flutter based on the snapshots of a Stream, there's a widget called StreamBuilder. This tutorial explains how to use the widget.