TRƯỜNG CAO ĐẲNG LÝ TỰ TRỌNG TPHCM

KHOA CÔNG NGHỆ THÔNG TIN

ĐỒ ÁN CHUYÊN NGHÀNH

PHẦN MỀM THƯƠNG MẠI ĐIỆN TỬ GLOBLUX

GVGD: THẦY **NGUYỄN VĂN DANH**

NHÓM 3 :

- Võ Hồng Quân
- Cao Hồng Chương
- Nguyễn Hoàng Hải

MỤC LỤC

PHẦN 1: GIỚI THIỆU	6
1.1.Mở đầu	6
1.2.Giới thiệu về Flutter	6
1.3.Phạm vi công việc	7
PHẦN 2:PHÂN TÍCH – THIẾT KẾ	8
2.1.Phân tích yêu cầu	8
• Use Case	8
• Flow Chart	9
2.2.Thiết kế CSDL	9
PHẦN 3:HƯỚNG DẪN SỬ DỤNG	12
I. Các màn hình	13
• Màn hình đăng nhập	13

• Màn hình đăng kí	14
• Màn hình quên mật khẩu	15
• Màn hình chính	16
• Menu	17
• Lọc theo danh mục	18
• Tìm kiếm sản phẩm	19
• Thay đổi thông tin	20
• Quản lý địa chỉ	21
• Đơn đặt hàng	22
• Chi tiết sản phẩm	23
• Quản lý sản phẩm	24
• Thêm sản phẩm	25
• Giỏ hàng	26
• Thông tin chương trình	27

II. Source code	28
• Login Source code	28
• Register Source code	34
• Forgot password Source code	42
• Home Source code	47
• Search Source code	77
• Category Source code	80
• Change Name Source code	87
• Change Email Source code	91
• Change Password Source code	96
• Edit address Source code	101
• Edit product Source code	111
• Manager Address Source code	129
• Orders Screen Source code	141

• My product Source code	150
• Product Details Source code	157
• Các lớp model	178
• Các lớp services	187
• Các lớp Exceptions	214
• Các widget con khác	221
• Các lớp Utils	241
• Lớp Wrapper & Main	246
• Phiên bản SDK & Thư viện	247
PHẦN 4:BẢNG PHÂN CÔNG	248
• Phân công tổng quát	248
• Phân công chi tiết	249

PHẦN 1:GIỚI THIỆU

1.1.Mở đầu

- Trong những năm gần đây, sự phát triển mạnh mẽ của thương mại điện tử trên thế giới đã góp phần làm thay đổi cách thức kinh doanh, giao dịch truyền thống và đem lại những lợi ích to lớn cho xã hội.
- Sự ra đời của thương mại điện tử đã đánh dấu sự bắt đầu của một hệ thống mới nhằm phát triển kinh tế, một trong những điều kiện cơ bản và có ý nghĩa quyết định trong việc phát triển thương mại điện tử là việc hoàn thiện các dịch vụ thanh toán điện tử.
- Thực tế đã chứng minh thanh toán điện tử l à một trong những điều kiện cốt lõi để phát triển thương mại điện tử với vai trò là một khâu không thể tách rời của quy trình giao dịch và còn là biện pháp xác thực việc ký kết hợp đồng giữa ng ười bán và người mua.
- Qua đó thương mại điện tử tạo điều kiện cho việc thiết lập và củng cố mối quan hệ giữa các nhân tố tham gia v ào quá trình thương mại. Thông qua mạng, các đối tượng tham gia có thể giao tiếp trực tiếp v à liên tục với nhau nhờ đó sự hợp tác lẫn sự quản lý đều được tiến hành nhanh chóng và liên tục .
- Vì vậy nhóm 3 đã chọn đề tài phần mềm thương mại điện tử GLOBLUX để hiểu thêm và đóng góp một phần nhỏ vào việc hoàn thiện và phát triển hệ thống thương mại điện tử ở Việt Nam
- Trong quá trình làm đồ án, mặc dù đã tìm kiếm thông tin và đọc nhiều tài liệu tham khảo nhưng do kiến thức còn hạn chế, chưa nắm bắt được nhiều thông tin về thương mại điện tử nên sẽ gặp nhiều thiếu sót. Mong thầy và các bạn đóng góp ý kiến để nhóm hoàn thiện bài tiểu luận. Xin chân thành cảm ơn.

1.2.Giới thiệu về Flutter

1.2.1.Lời giới thiệu:

- Flutter là mobile **UI framework** của Google để tạo ra các giao diện chất lượng cao trên **iOS và Android** trong khoảng thời gian ngắn. Flutter hoạt động với những code sẵn có được sử dụng bởi các lập trình viên, các tổ chức.

1.2.2.Những điểm nổi bật

- Nếu bạn đang tìm kiếm các phương pháp thay thế để phát triển ứng dụng Android, bạn nên cân nhắc thử Flutter của Google, một framework dựa trên ngôn ngữ lập trình **Dart**.

- Các ứng dụng được xây dựng với Flutter hầu như không thể phân biệt với những ứng dụng được xây dựng bằng cách sử dụng Android SDK, cả về giao diện và hiệu suất. Hơn nữa, với những tinh chỉnh nhỏ, chúng có thể chạy trên thiết bị iOS.
- Chạy ở **60 fps**, giao diện người dùng được tạo ra với Flutter thực thi tốt hơn nhiều so với những ứng dụng được tạo ra với các framework phát triển đa nền tảng khác.
- Flutter sử dụng Dart, một ngôn ngữ nhanh, hướng đối tượng với nhiều tính năng hữu ích như mixin, generic, isolate, và static type.
- Flutter có các thành phần UI của riêng nó, cùng với một cơ chế để kết xuất chúng trên nền tảng Android và iOS. Hầu hết các thành phần giao diện người dùng, đều sẵn dùng, phù hợp với các nguyên tắc của Material Design.
- Các ứng dụng Flutter có thể được phát triển bằng cách sử dụng IntelliJ IDEA, Android Studio hoặc Visual Code.
- Fast Development: Tính năng **Hot Reload** hoạt động trong milliseconds để hiện thị giao diện tới bạn. Sử dụng tập hợp các widget có thể customizable để xây dựng giao diện trong vài phút. Ngoài ra Hot Reload còn giúp bạn thêm các tính năng, fix bug tiết kiệm thời gian hơn mà không cần phải thông qua máy ảo, máy android hoặc iOS.
- Có rất nhiều các thành phần để xây dựng giao diện của Flutter vô cùng đẹp mắt theo phong cách **Material Design và Cupertino**, hỗ trợ nhiều các APIs chuyển động, smooth scrolling...
- Các widget của fluter kết hợp các sự khác biệt của các nền tảng ví dụ như scrolling, navigation, icons, font để cung cấp một hiệu năng tốt nhất tới iOS và Android.

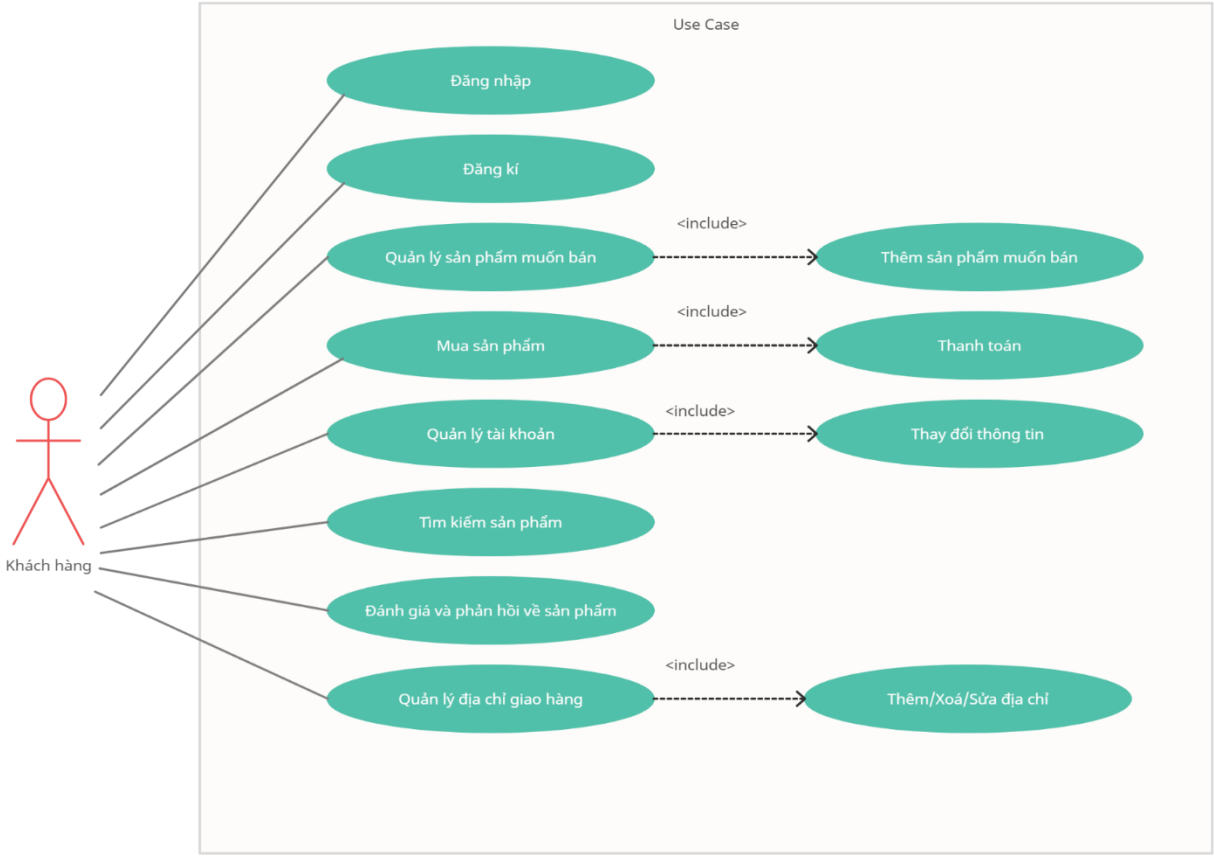
1.3.Phạm vi công việc

- Tra cứu,tìm kiếm sản phẩm.
- Chế độ bán hàng cho phép thêm mặt hàng và quản lý hàng hoá của tài khoản đó.
- Quản lý địa chỉ giao hàng của tài khoản(Không giới hạn số lượng địa chỉ thêm vào).
- Tra cứu và đánh giá đơn đặt hàng đã mua.

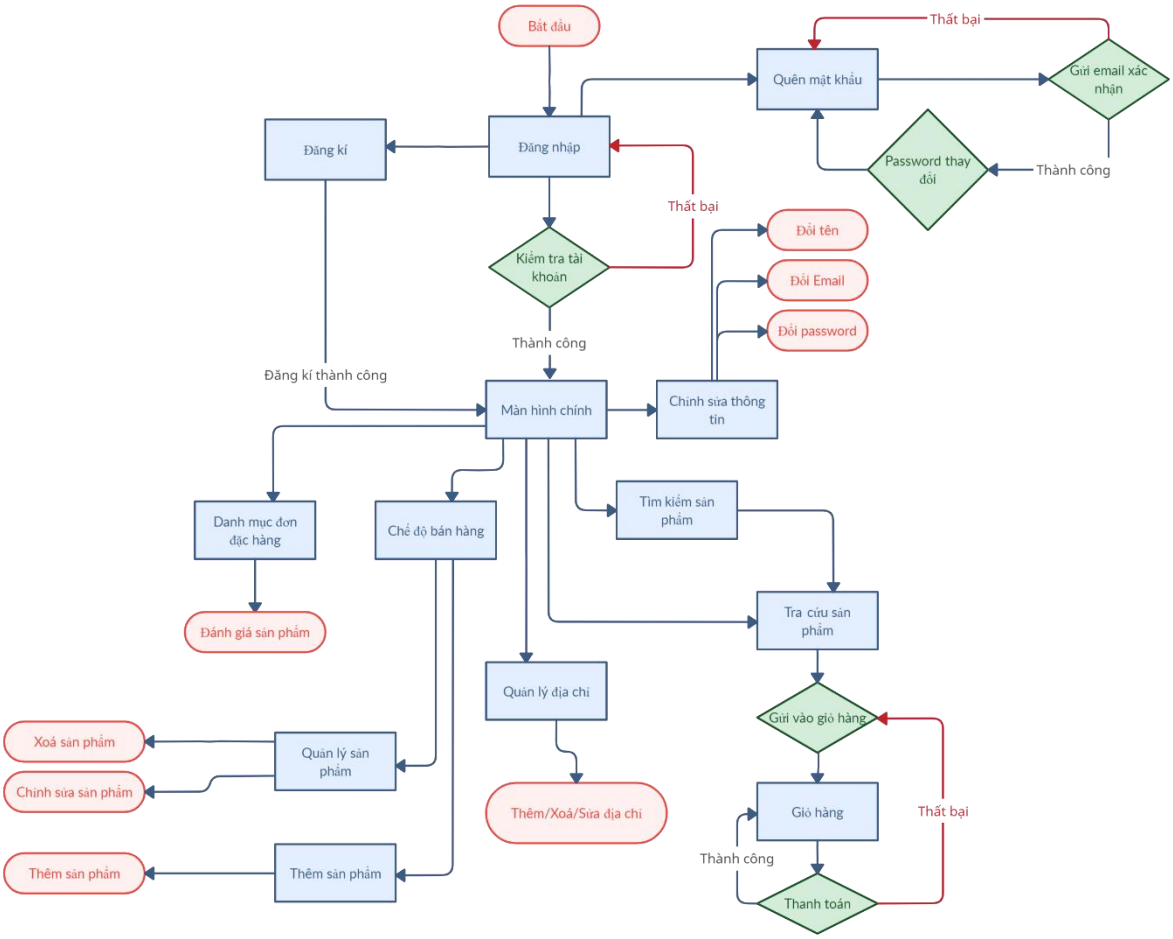
PHẦN 2:PHÂN TÍCH – THIẾT KẾ

2.1.Phân tích yêu cầu:

- Use Case:



○ Flow Chart:



2.2.Thiết kế CSDL:

Table Product(Sản phẩm):

Tên thuộc tính	Kiểu dữ liệu	Ghi chú
id	String	Được tạo bởi giá trị random của FireStore
title	String	Tiêu đề sản phẩm
variant	String	Sự khác nhau(Màu sắc)
seller	String	Người bán
image	List<String>	List Hình ảnh

highlight	String	Điểm nổi bật
description	String	Chi tiết sản phẩm
favourite	bool	Được yêu thích
owner	String	Chủ sở hữu(Công ty)
productType	ProcuctType(Model Class)	Danh mục của sản phẩm
searchTag	List<String>	Các thẻ tìm kiếm
rating	num	Số điểm đánh giá
discountPrice	num	Giảm giá
originalPrice	num	Giá gốc của sản phẩm

Table Adress(Địa chỉ giao hàng):

Tên thuộc tính	Kiểu dữ liệu	Ghi chú
id	String	Được tạo bởi giá trị random của FireStore
title	String	Tiêu đề
address_line_1	String	Địa chỉ thứ 1
address_line_2	String	Địa chỉ thứ 2
city	String	Thành phố
district	String	Quận/Huyện
landmark	String	Số nhà
phone	String	Số điện thoại
pincode	String	Mã PIN
receiver	String	Tên người nhận
state	String	Xã/Phường

Table Cart(Giỏ hàng):

Tên thuộc tính	Kiểu dữ liệu	Ghi chú
id	String	Được tạo bởi giá trị random của FireStore
itemCount	int	Số lượng trong giỏ hàng

Table OrderedProduct(Sản phẩm đã thanh toán):

Tên thuộc tính	Kiểu dữ liệu	Ghi chú
id	String	Được tạo bởi giá trị random của FireStore
productUid	String	ID của sản phẩm
orderDate	String	Ngày thanh toán

Table Review(Đánh giá sản phẩm):

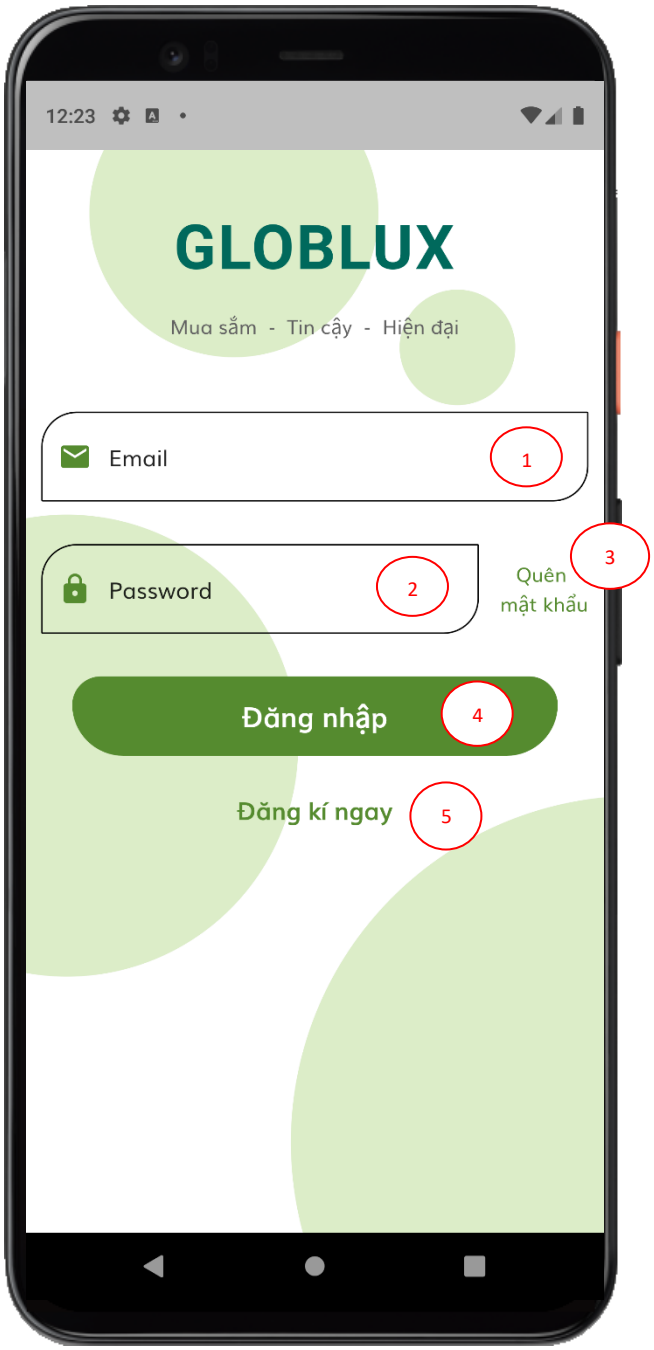
Tên thuộc tính	Kiểu dữ liệu	Ghi chú
id	String	Được tạo bởi giá trị random của FireStore
reviewerUid	String	ID của khách hàng đánh giá
rating	int	Số sao đánh giá(Max:5)
feedback	String	Phản hồi của khách hàng

Table AppReview(Đánh giá phần mềm):

Tên thuộc tính	Kiểu dữ liệu	Ghi chú
id	String	Được tạo bởi giá trị random của FireStore
liked	bool	Đã thích
feedback	String	Phản hồi của khách hàng

PHẦN 3:HƯỚNG DẪN SỬ DỤNG

I. CÁC MÀN HÌNH CÓ TRONG CHƯƠNG TRÌNH

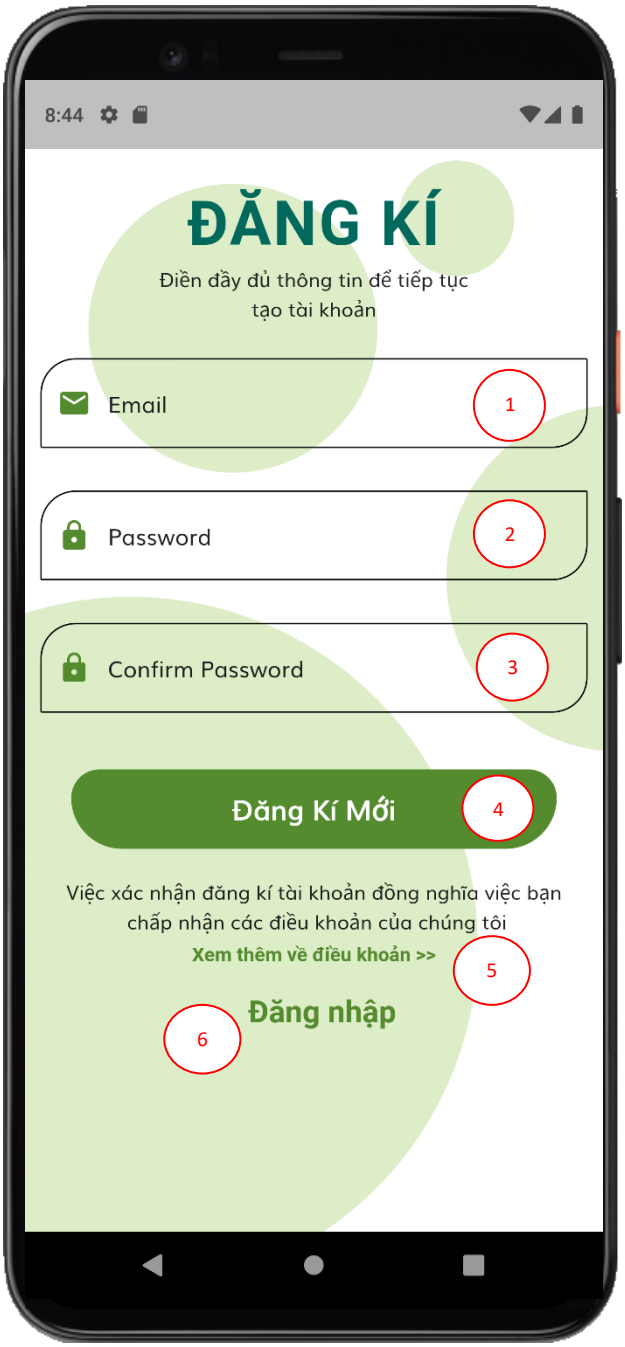


LOGIN SCREEN

(Đăng nhập)

Chú thích:

1. Nhập email
2. Nhập khẩu
3. Chuyển đến màn hình lấy mật khẩu
4. Nút đăng nhập
5. Chuyển đến màn hình đăng kí

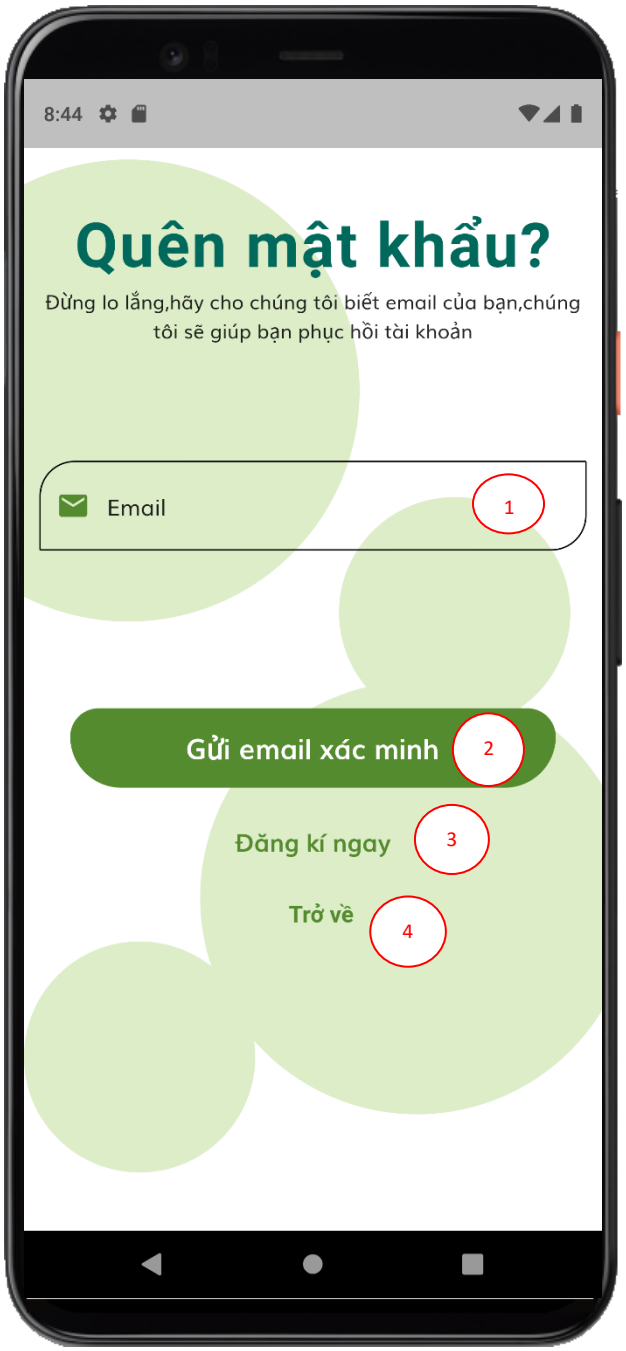


REGISTER SCREEN

(Đăng kí)

Chú thích:

1. Nhập email
2. Nhập khẩu
3. Xác nhận mật khẩu
4. Nút đăng kí
5. Xem thêm về điều khoản
6. Trở về màn hình đăng nhập



FORGOT PASSWORD SCREEN

(Quên mật khẩu)

Chú thích:

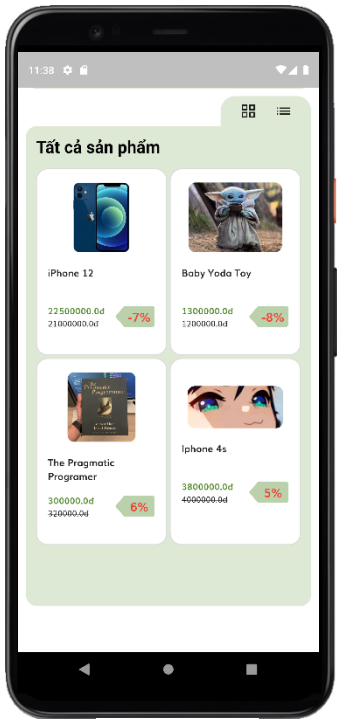
1. Nhập email đã đăng ký
2. Nút gửi email xác minh quên mật khẩu
3. Chuyển đến màn hình đăng kí
4. Quay về trang đăng nhập

MAIN SCREEN

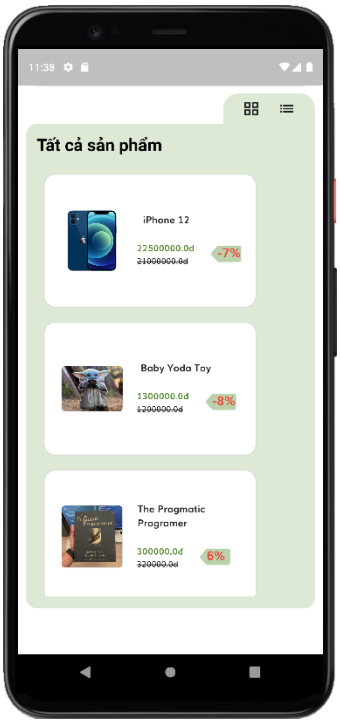
(Màn hình chính)

Chú thích:

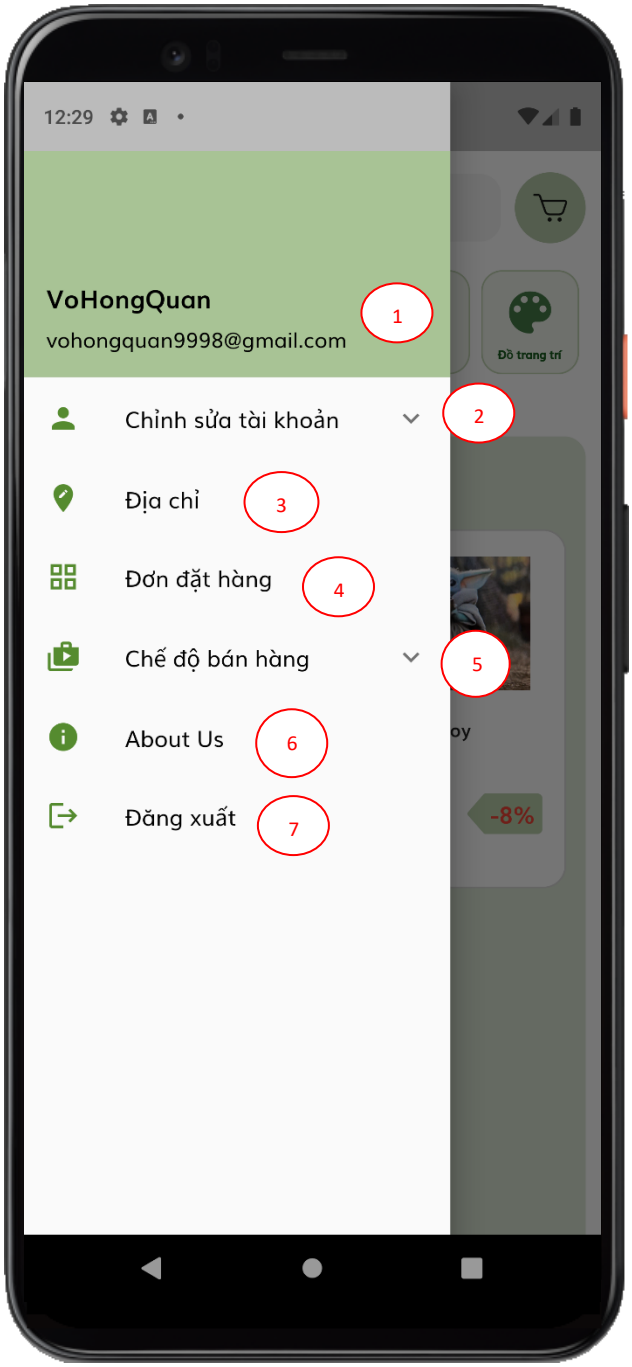
1. Menu
2. Khung tìm kiếm sản phẩm
3. Giỏ hàng
4. Danh mục
5.List sản phẩm cùng với đánh giá
6. Chuyển chế độ GridView và Listview
7.List tất cả sản phẩm



GridView



ListView

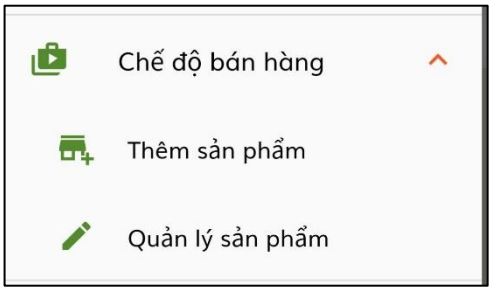


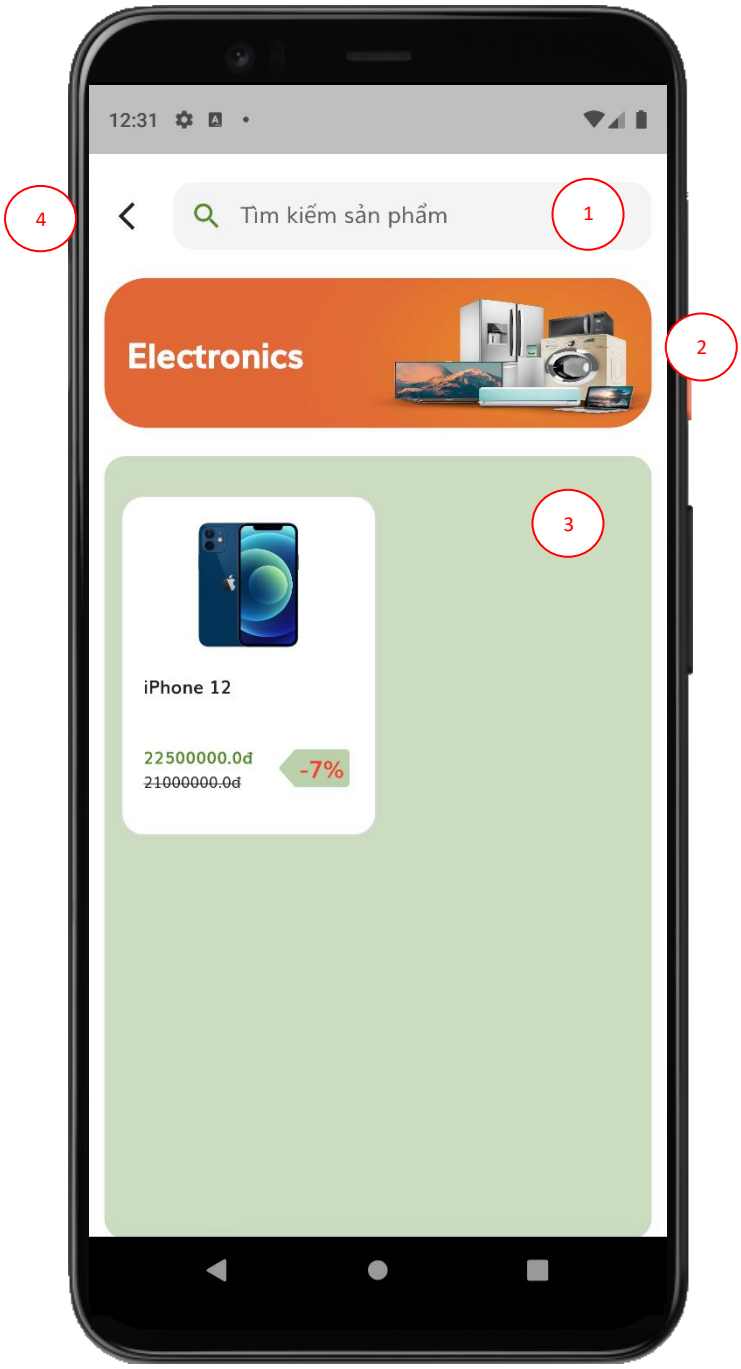
DRAWER MENU SCREEN

(Menu vuốt)

Chú thích:

1. Tên tài khoản & Email
2. Chỉnh sửa thông tin tài khoản
3. Địa chỉ giao hàng
4. Đơn đặt hàng
5. Chế độ bán hàng
6. Thông tin ứng dụng
7. Đăng xuất chương trình



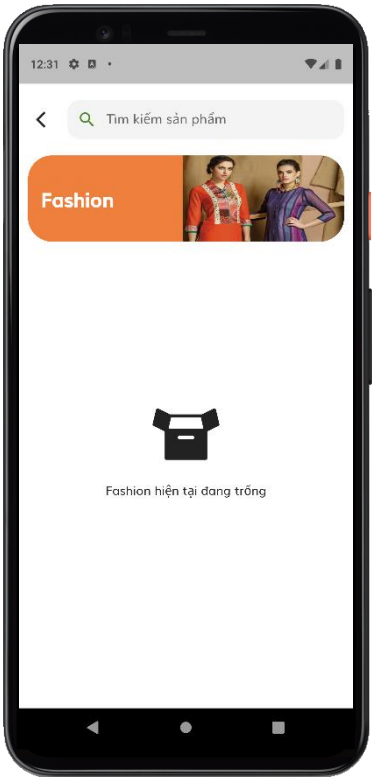


CATEGORY SCREEN

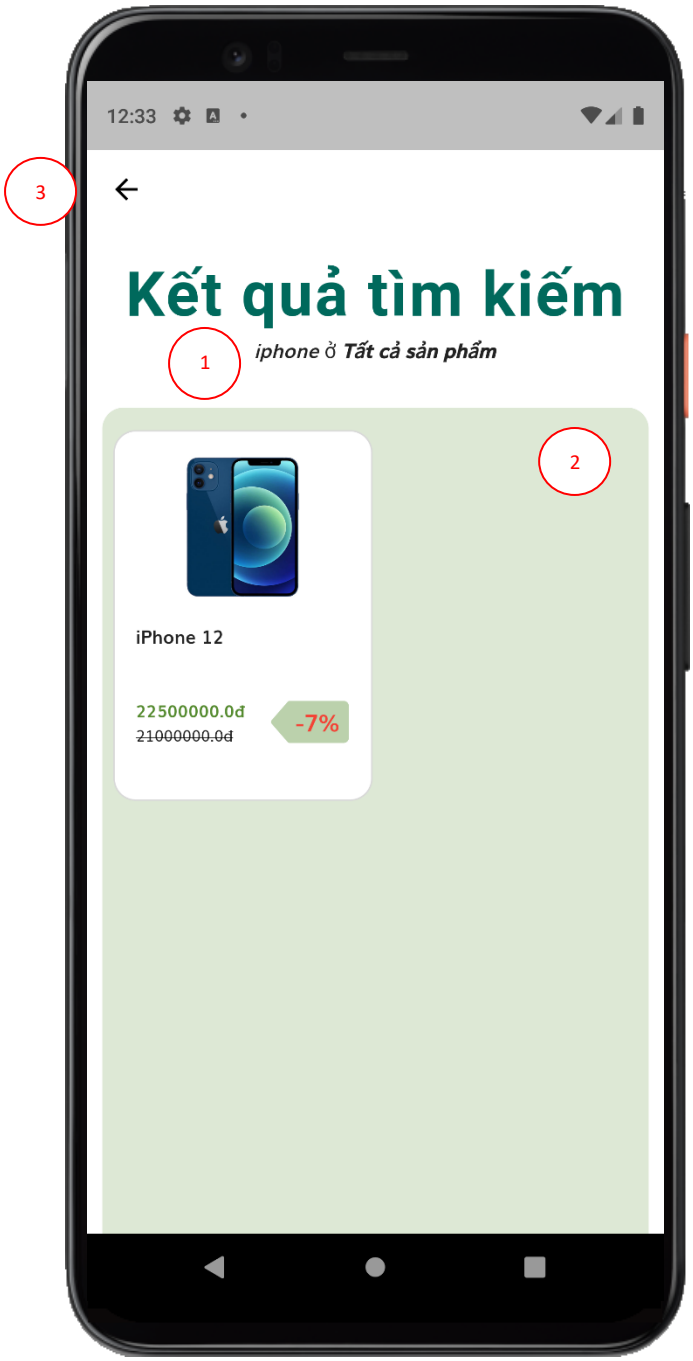
(Lọc theo danh mục)

Chú thích:

1. Khung tìm kiếm
2. Tiêu đề danh mục đã lọc
3. Tất cả sản phẩm thuộc danh mục đó
4. Nút trở về



Giao diện danh mục nếu không có sản phẩm nào

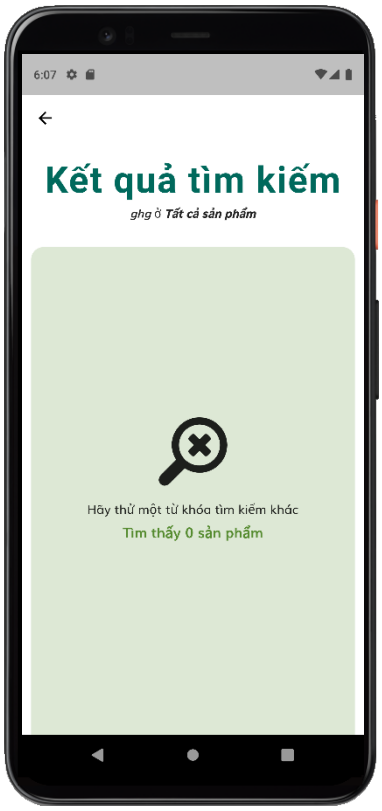


SEARCH SCREEN

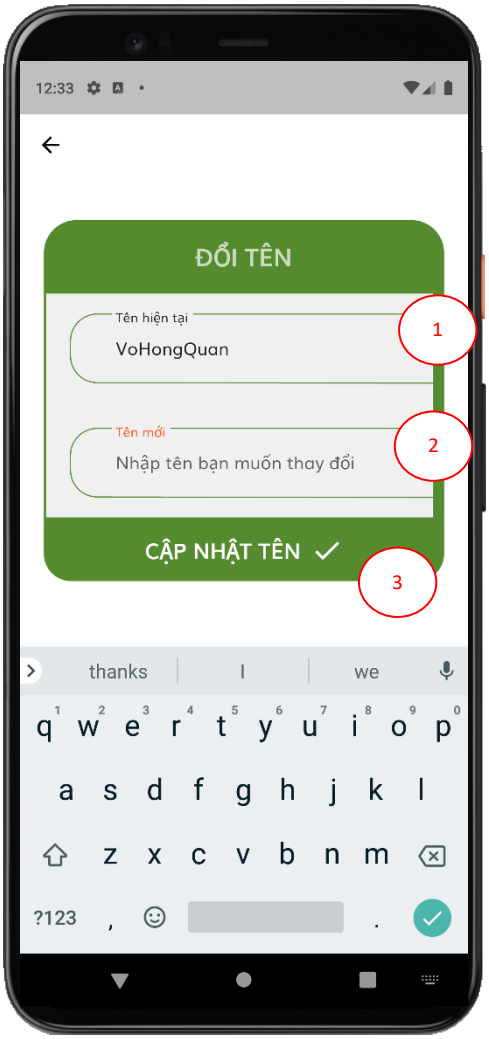
(Tìm kiếm)

Chú thích:

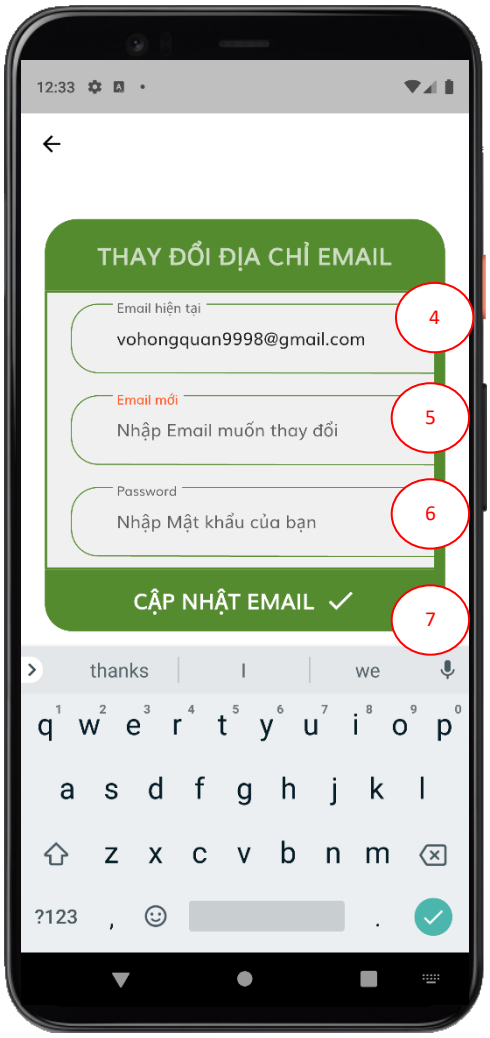
1. Từ "iphone" là từ bạn muốn tìm kiếm
2. Danh sách các Sản phẩm trùng khớp
3. Nút trở về



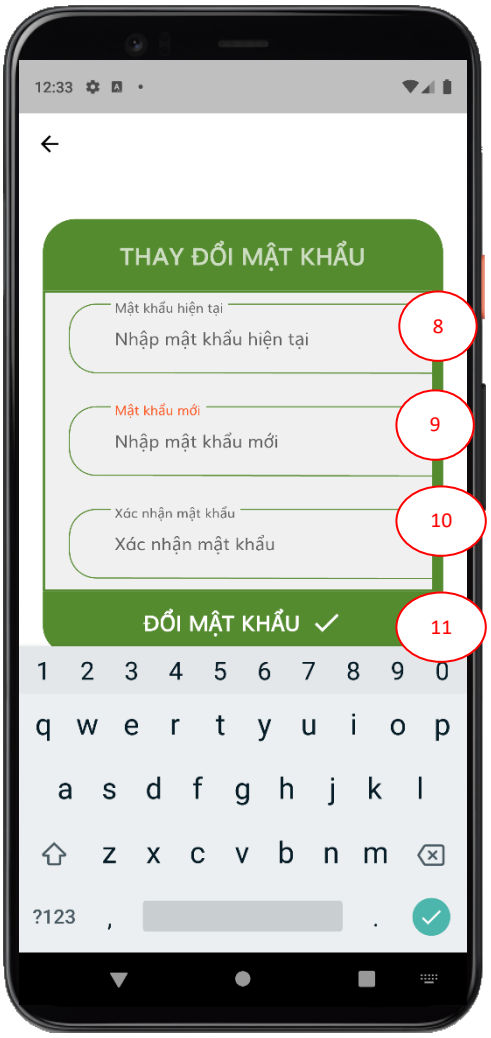
Giao diện nếu không có kết quả tìm kiếm trùng khớp



Giao diện Thay đổi tên



Giao diện Thay đổi Email



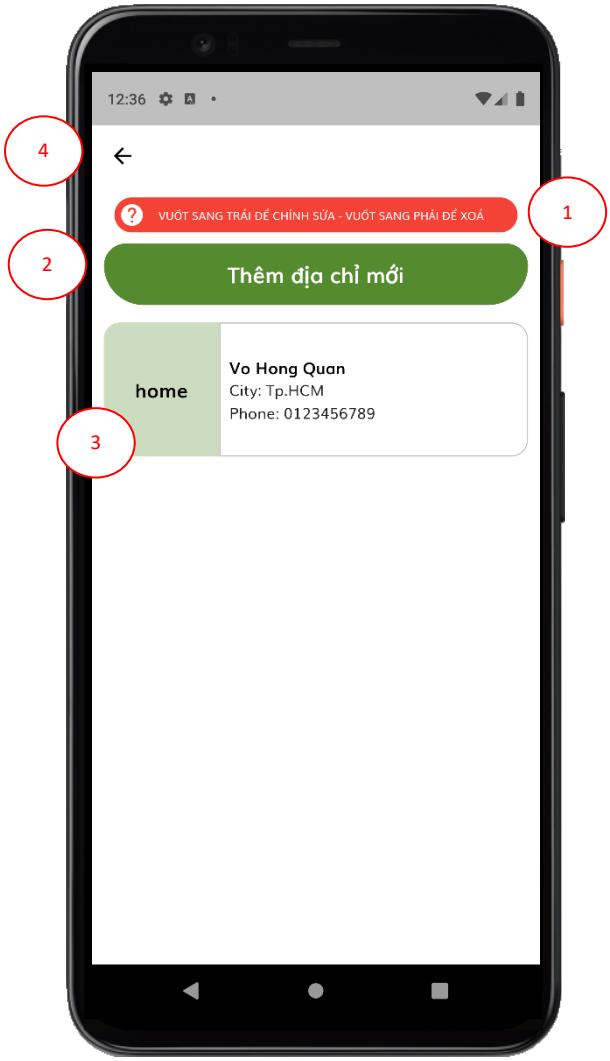
Giao diện Thay đổi Mật khẩu

CHANGE INFOMATION SCREEN

(Thay đổi thông tin)

Chú thích:

1. Tên tài khoản hiện tại (Tự động điền)
2. Tên tài khoản muốn đổi
3. Nút Cập nhật Tên tài khoản
4. Email hiện tại (Tự động điền)
5. Email mới
6. Mật khẩu xác nhận để đổi email
7. Nút Cập nhật Email
8. Mật khẩu hiện tại
9. Mật khẩu mới
10. Xác nhận mật khẩu mới
11. Nút Cập nhật mật khẩu



ADDRESS SCREEN

(Thông tin địa chỉ)

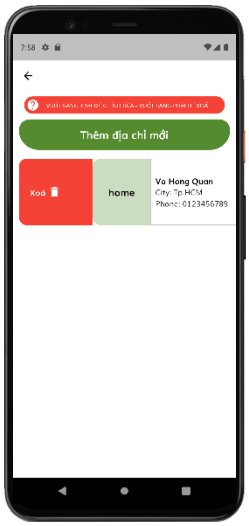
Chú thích:

1. Thanh hướng dẫn (Vuốt trái : Sửa – Vuốt phải : Xoá)

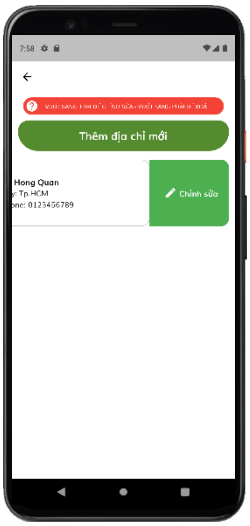
2. Thêm địa chỉ (Chuyển đến màn hình thêm địa chỉ)

3. Địa chỉ đã thêm (Không giới hạn số lượng thêm vào)

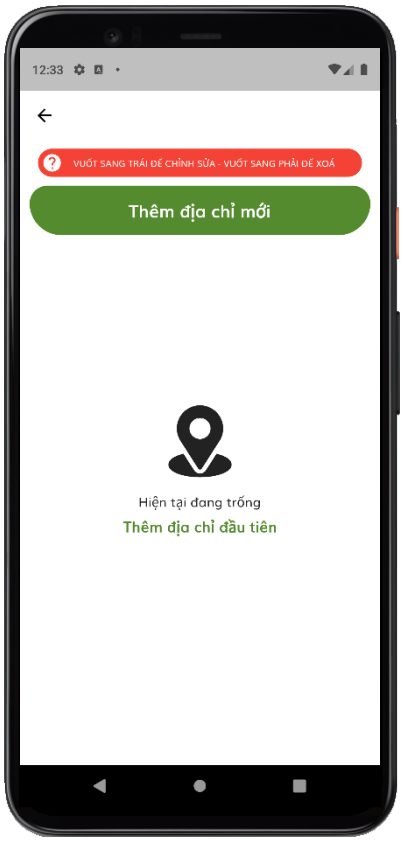
4. Nút trở về Màn hình chính



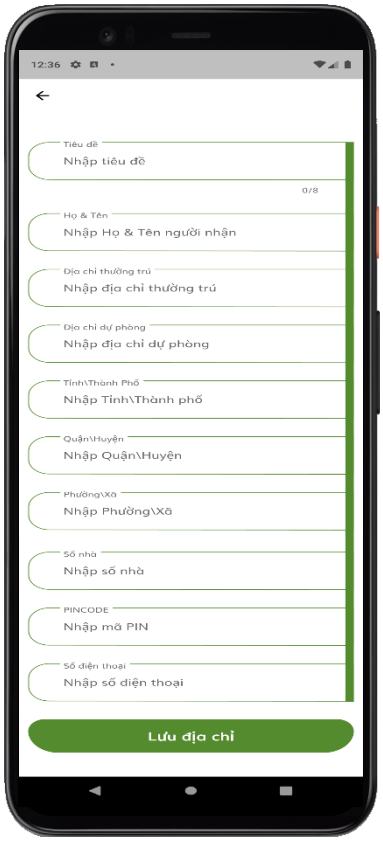
Vuốt qua phải để Xoá



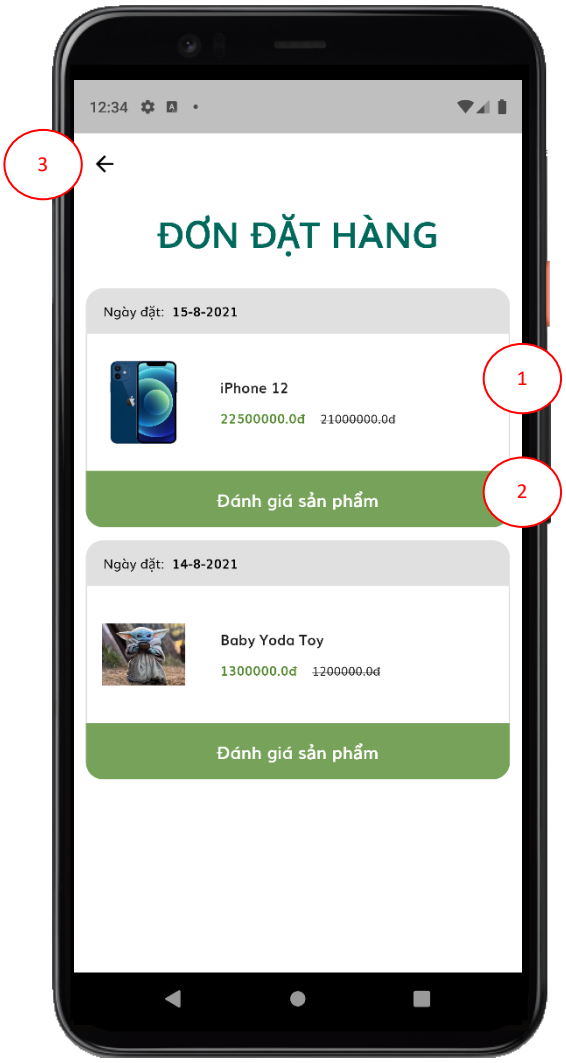
Vuốt qua trái để Sửa



Giao diện nếu không có địa chỉ nào



Giao diện Thêm địa chỉ

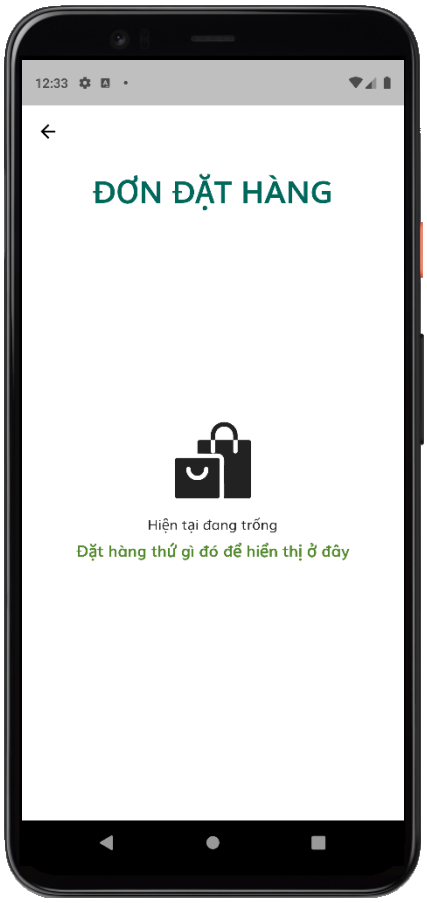


THE ORDERS SCREEN

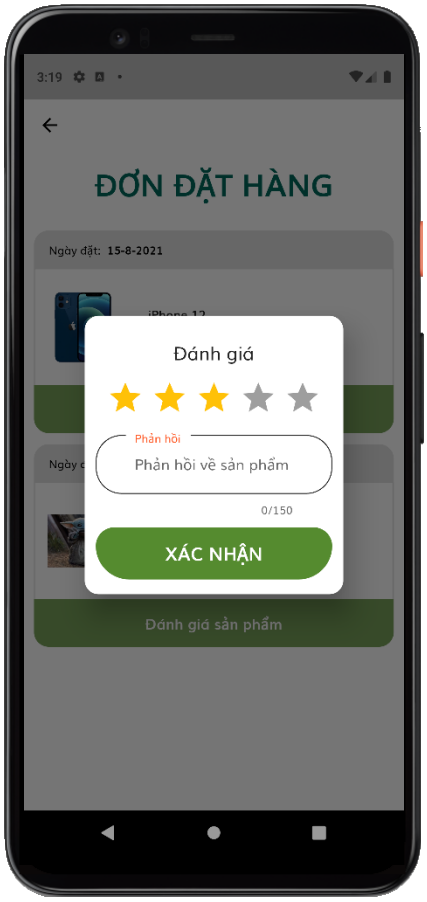
(Đơn đặt hàng)

Chú thích:

1. Sản phẩm đã đặt hàng
2. Nút đánh giá sản phẩm
3. Nút trở về Màn hình chính



Giao diện nếu không đơn đặt hàng nào



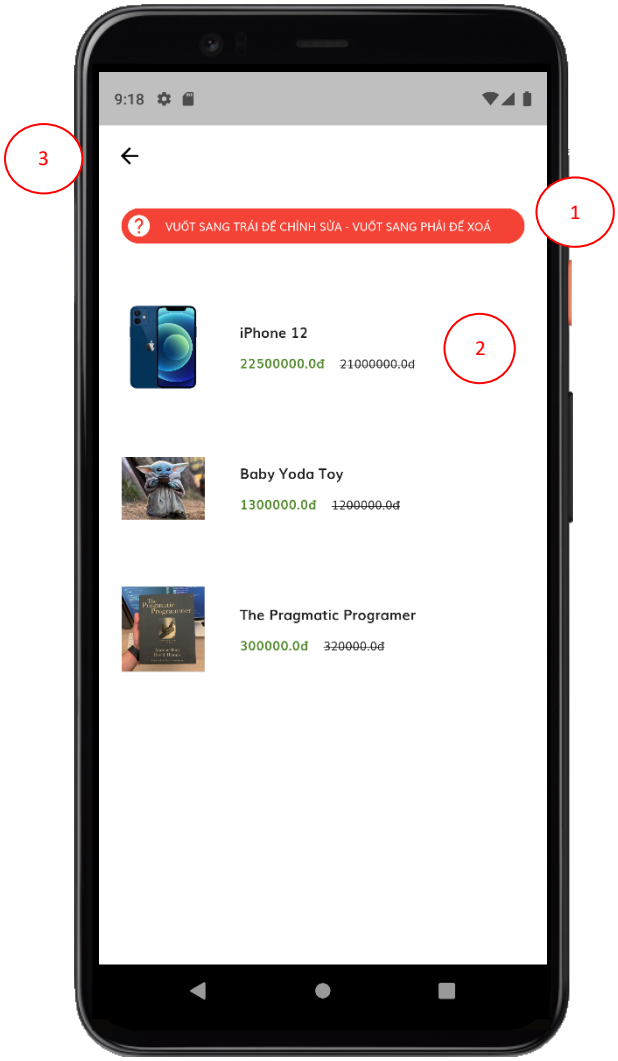
Giao diện Đánh giá sản phẩm

DETAILS SCREEN

(Thông tin sản phẩm)

Chú thích:

1. Hình ảnh sản phẩm
2. Tất cả hình ảnh về sản phẩm
3. Thông tin sản phẩm
4. Thông tin đánh giá
5. Nút thêm giỏ hàng
6. Nút trở về màn hình chính



PRODUCT MANAGER SCREEN

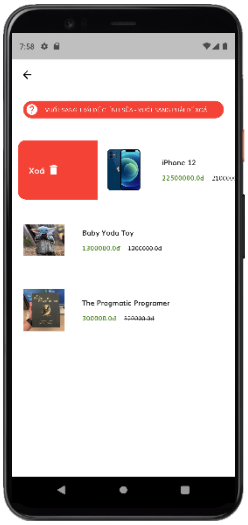
(Quản lý sản phẩm)

Chú thích:

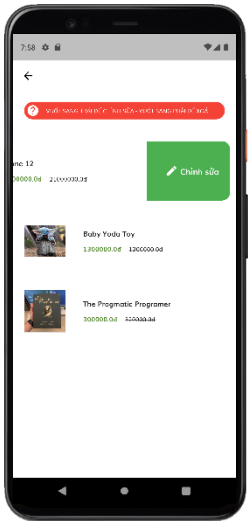
1. Thanh hướng dẫn (Vuốt trái : Sửa – Vuốt phải : Xoá)

2. List các sản phẩm đã thêm

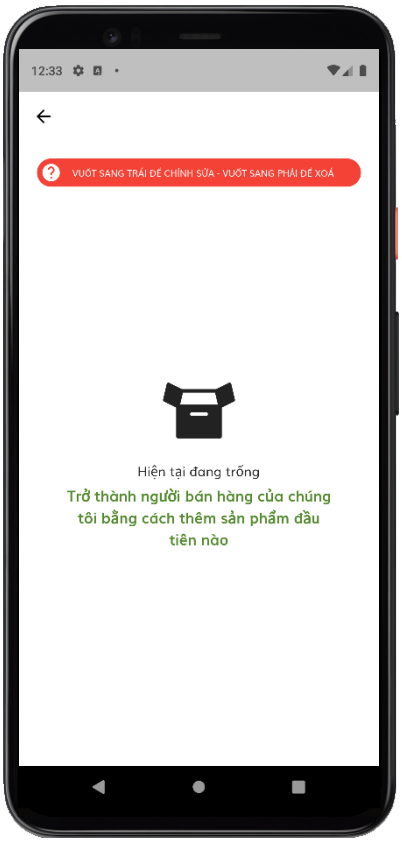
3. Nút trở về Màn hình chính



Vuốt qua phải để Xoá



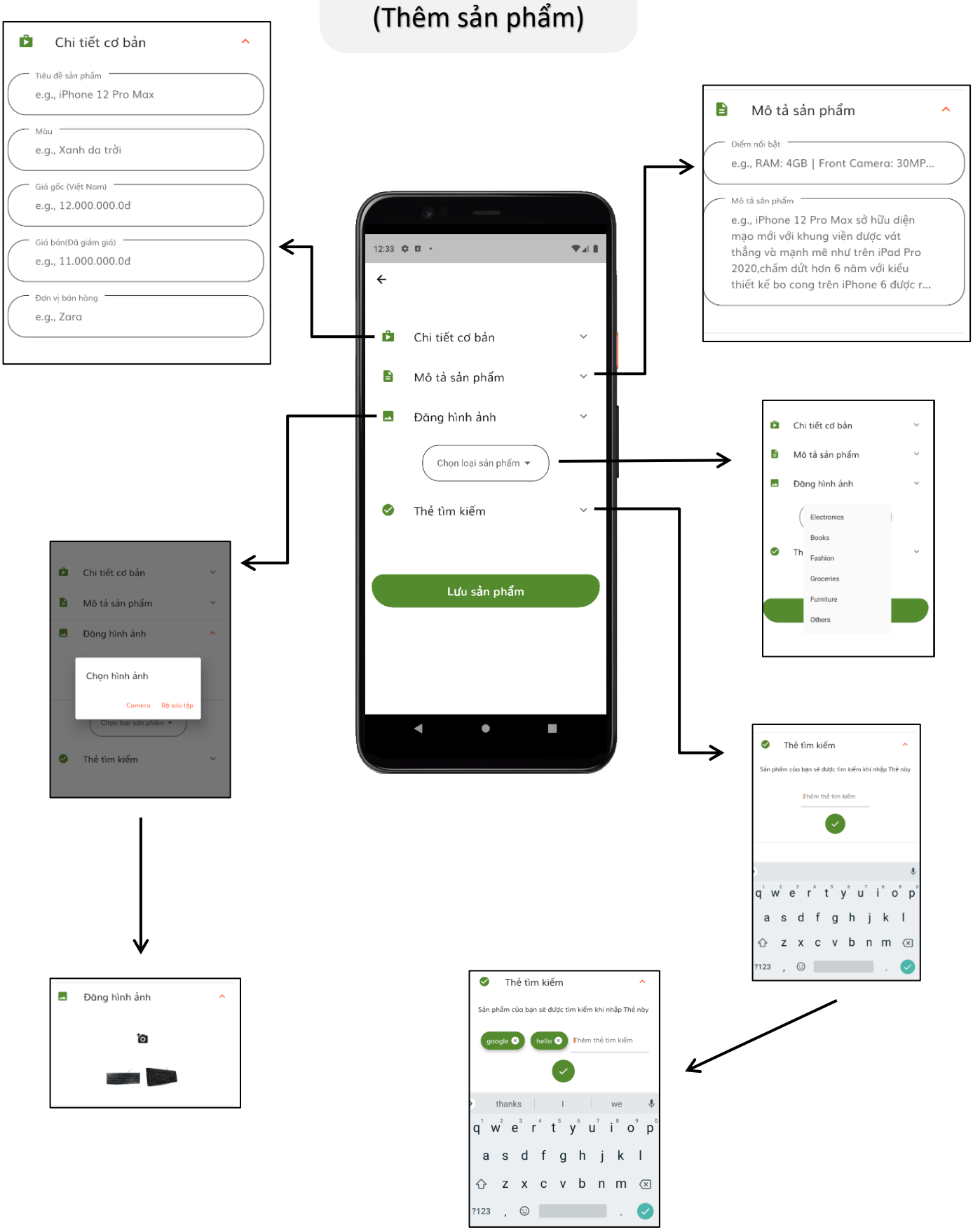
Vuốt qua trái để Sửa

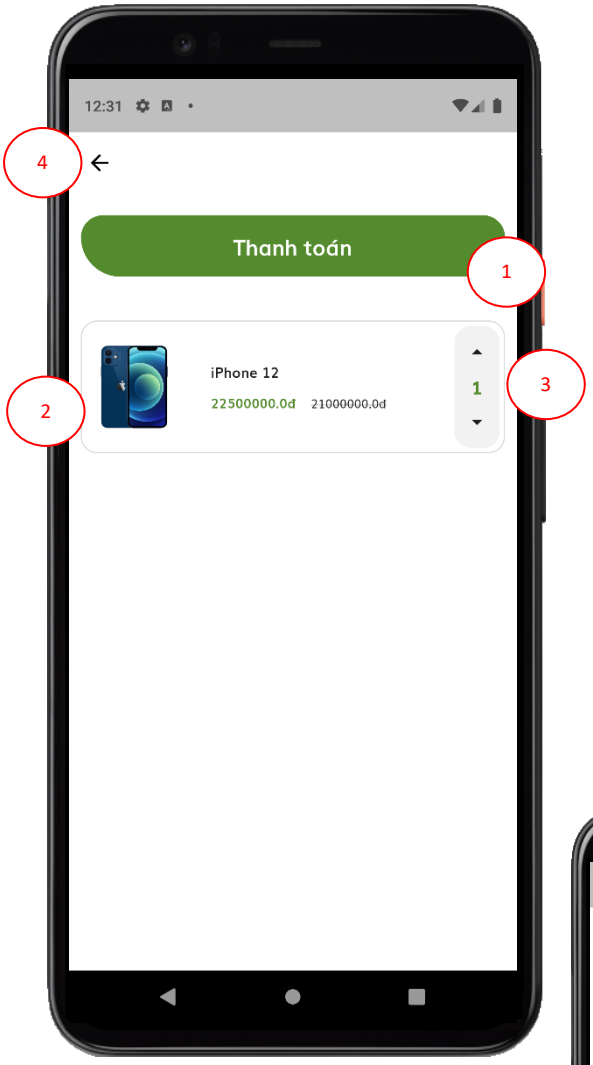


Giao diện nếu không có sản phẩm nào

ADD PRODUCT SCREEN

(Thêm sản phẩm)



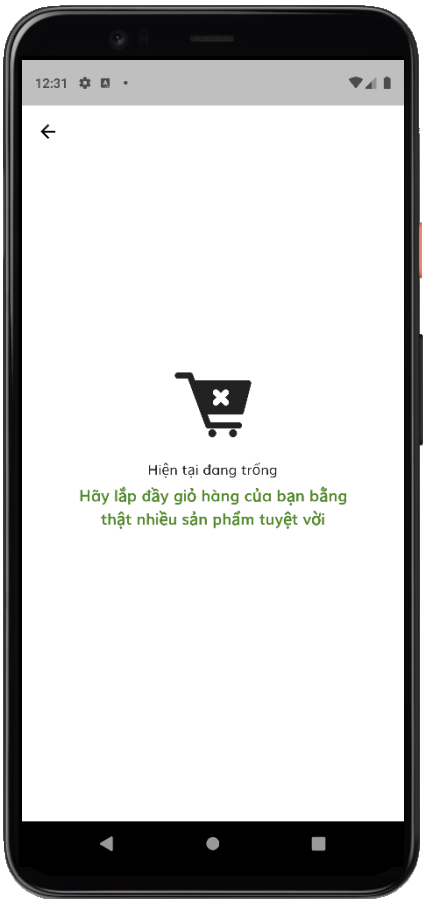


CART SCREEN

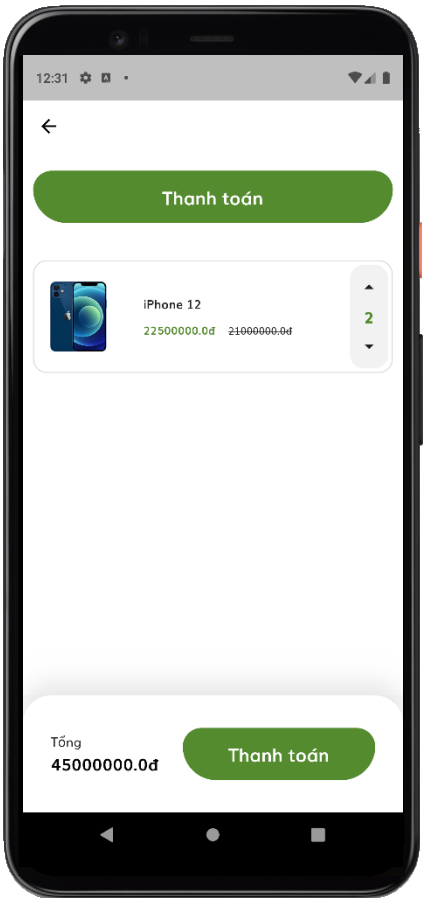
(Giỏ hàng)

Chú thích:

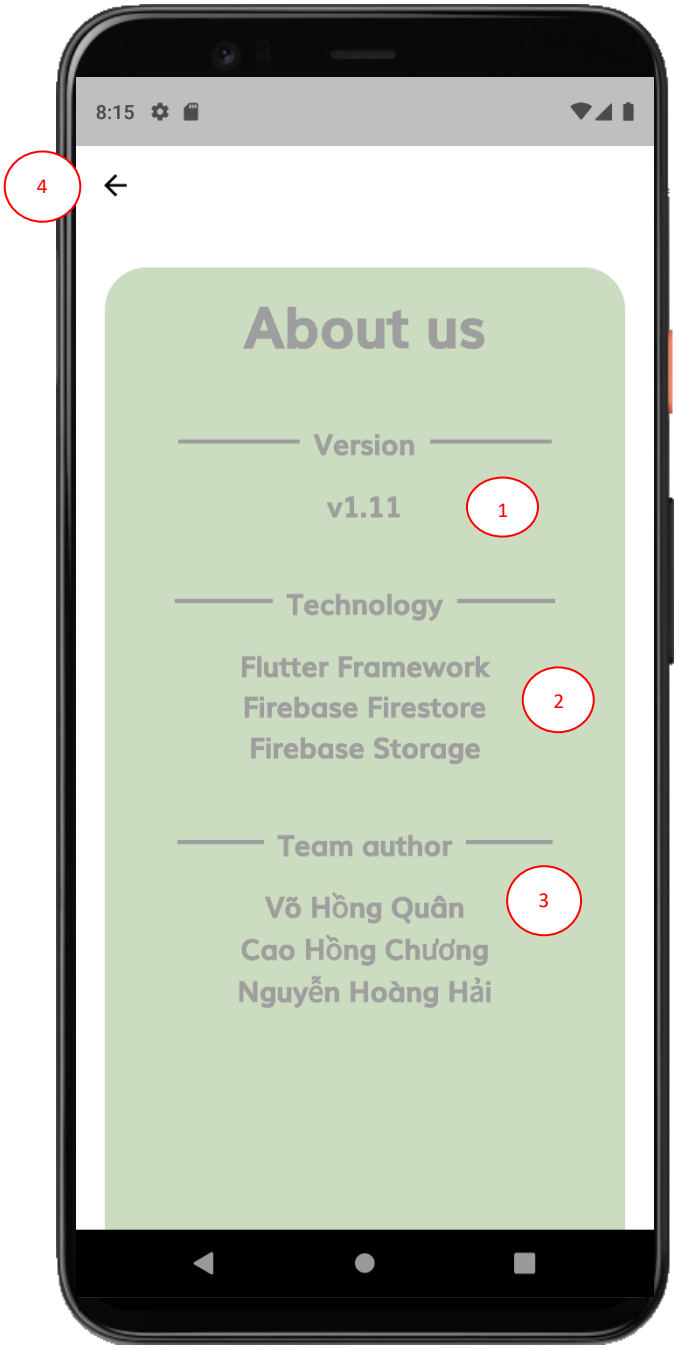
1. Nút thanh toán
2. List sản phẩm
3.Tăng giảm số lượng
4. Nút trở về Màn hình chính



Giao diện nếu không có sản phẩm nào



Giao diện Sau khi nhấn thanh toán



ABOUT US SCREEN

(Thông tin chương trình)

Chú thích:

1. Phiên bản của chương trình
2. Công nghệ sử dụng trong chương trình
3. Nhóm tác giả thực hiện
4. Nút quay trở về màn hình chính

II. SOURCE CODE

1. LOGIN SCREEN SOURCE CODE

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/exceptions/auth/firebase_exception.dart';
import 'package:flutter_online_shop/exceptions/auth/sign_in_exception.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/src/forgot_password/forgot_password_screen.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/circleContainer.dart';
import 'package:flutter_online_shop/widget/default_button.dart';
import 'package:flutter_online_shop/widget/no_acc_text.dart';
import 'package:future_progress_dialog/future_progress_dialog.dart';
import 'package:logger/logger.dart';

class SignInScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    SizeConfig().init(context);
    return Scaffold(
      resizeToAvoidBottomInset: false,
      body: SafeArea(
        child: Container(
          width: double.infinity,
          height: SizeConfig.screenHeight,
          child: Stack(
            children: [
              Positioned(
                top: -getProportionateScreenHeight(
                    SizeConfig.screenHeight * .07),
                left: getProportionateScreenWidth(SizeConfig.screenWidth * .1),
                child: CircleContainer(
                  color: kCircleContainer,
                  sizeRate: .5,
                ),
              ),
              Positioned(
                top:
                    getProportionateScreenHeight(SizeConfig.screenHeight * .12),
                right:
                    getProportionateScreenWidth(SizeConfig.screenWidth * .14),
                child: CircleContainer(
                  color: kCircleContainer,
```

```
    sizeRate: .2,
  ),
),
Positioned(
  bottom:
      getProportionateScreenHeight(SizeConfig.screenHeight * .22),
  left: -getProportionateScreenWidth(SizeConfig.screenWidth * .3),
  child: CircleContainer(
    color: kCircleContainer,
    sizeRate: .8,
  ),
),
Positioned(
  bottom: -getProportionateScreenHeight(
      SizeConfig.screenHeight * .22),
  right:
      -getProportionateScreenWidth(SizeConfig.screenWidth * .6),
  child: CircleContainer(
    color: kCircleContainer,
    sizeRate: 1.2,
  ),
),
Padding(
  padding: EdgeInsets.symmetric(
      horizontal: getProportionateScreenWidth(screenPadding)),
  child: SingleChildScrollView(
    physics: BouncingScrollPhysics(),
    child: Column(
      children: [
        SizedBox(height: SizeConfig.screenHeight * 0.04),
        Text(
          "Globlux".toUpperCase(),
          style: headingStyle,
        ),
        SizedBox(height: SizeConfig.screenHeight * 0.02),
        Text(
          'Mua sắm \t-\t Tin cậy \t-\t Hiện đại',
          style: TextStyle(
            fontWeight: FontWeight.w500,
            color: Colors.grey[700],
          ),
        ),
        SizedBox(height: SizeConfig.screenHeight * 0.06),
        SignInForm(),
        SizedBox(height: SizeConfig.screenHeight * 0.03),
        NoAccountText(),
        SizedBox(height: getProportionateScreenHeight(20)),
      ],
```

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

class SignInForm extends StatefulWidget {
 @override
 _SignInFormState createState() => _SignInFormState();
}

class _SignInFormState extends State<SignInForm> {
 final _formkey = GlobalKey<FormState>();

 final TextEditingController emailFieldController = TextEditingController();
 final TextEditingController passwordFieldController = TextEditingController();

 @override
 void dispose() {
  emailFieldController.dispose();
  passwordFieldController.dispose();
  super.dispose();
 }

 @override
 Widget build(BuildContext context) {
  return Form(
   key: _formkey,
   child: Column(
    children: [
     buildEmailFormField(),
     SizedBox(height: getProportionateScreenHeight(30)),
     Row(
      mainAxisAlignment: MainAxisAlignment.spaceEvenly,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
       Expanded(child: buildPasswordFormField()),
       SizedBox(
        width: getProportionateScreenWidth(10),
       ),
       GestureDetector(
        onTap: () {
         Navigator.push(
```

```
                    context,
                    MaterialPageRoute(
                      builder: (context) => ForgotPasswordScreen(),
                    ));
                },
                child: Text(
                  "Quên \n mật khẩu",
                  textAlign: TextAlign.center,
                  style: TextStyle(
                    color: kPrimaryColor,
                  ),
                ),
              ),
            ],
          ),
          SizedBox(height: getProportionateScreenHeight(30)),
          Padding(
            padding: EdgeInsets.symmetric(
                horizontal: getProportionateScreenWidth(screenPadding * 2)),
            child: DefaultButton(
              text: "Đăng nhập",
              press: signInButtonCallback,
            ),
          ),
        ],
      ),
    );
  }

  Widget buildPasswordFormField() {
    return TextFormField(
      controller: passwordFieldController,
      obscureText: true,
      decoration: InputDecoration(
        enabledBorder: OutlineInputBorder(
          borderRadius: BorderRadius.only(
            topLeft: Radius.circular(25),
            bottomRight: Radius.circular(25),
            topRight: Radius.circular(1),
            bottomLeft: Radius.circular(1),
          ),
        ),
        focusedBorder: OutlineInputBorder(
          borderRadius: BorderRadius.only(
            topLeft: Radius.circular(25),
            bottomRight: Radius.circular(25),
            topRight: Radius.circular(1),
            bottomLeft: Radius.circular(1),
```

```
        ),
      ),
      labelText: "Password",
      hintText: 'Mật khẩu',
      labelStyle: TextStyle(color: Colors.black87),
      floatingLabelBehavior: FloatingLabelBehavior.auto,
      prefixIcon: Icon(
        Icons.lock,
        color: kPrimaryColor,
      ),
    ),
    validator: (value) {
      if (passwordFieldController.text.length < 8 &&
          passwordFieldController.text.isNotEmpty) {
        return kShortPassError;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildEmailFormField() {
  return TextFormField(
    controller: emailFieldController,
    keyboardType: TextInputType.emailAddress,
    decoration: InputDecoration(
        enabledBorder: OutlineInputBorder(
          borderRadius: BorderRadius.only(
            topLeft: Radius.circular(25),
            bottomRight: Radius.circular(25),
            topRight: Radius.circular(1),
            bottomLeft: Radius.circular(1),
          ),
        ),
        focusedBorder: OutlineInputBorder(
          borderRadius: BorderRadius.only(
            topLeft: Radius.circular(25),
            bottomRight: Radius.circular(25),
            topRight: Radius.circular(1),
            bottomLeft: Radius.circular(1),
          ),
        ),
        labelStyle: TextStyle(color: Colors.black87),
        labelText: "Email",
        hintText: 'Tài khoản email',
        floatingLabelBehavior: FloatingLabelBehavior.auto,
        prefixIcon: Icon(
```

```
          Icons.mail,
          color: kPrimaryColor,
        )),
    validator: (value) {
      if (!emailValidatorRegExp.hasMatch(emailFieldController.text) &&
          emailFieldController.text.isNotEmpty) {
        return kInvalidEmailError;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Future<void> signInButtonCallback() async {
  if (_formkey.currentState.validate()) {
    _formkey.currentState.save();
    final AuthentificationService authService = AuthentificationService();
    bool signInStatus = false;
    String snackbarMessage;
    try {
      final signInFuture = authService.signIn(
        email: emailFieldController.text.trim(),
        password: passwordFieldController.text.trim(),
      );
      signInFuture.then((value) => signInStatus = value);
      signInStatus = await showDialog(
        context: context,
        builder: (context) {
          return FutureProgressDialog(
            signInFuture,
            message: Text("Đang đăng nhập vào tài khoản"),
          );
        },
      );
      if (signInStatus == true) {
        snackbarMessage = "Đăng nhập thành công";
      } else {
        throw FirebaseSignInAuthUnknownReasonFailure();
      }
    } on MessagedFirebaseAuthException catch (e) {
      snackbarMessage = e.message;
    } catch (e) {
      snackbarMessage = e.toString();
    } finally {
      Logger().i(snackbarMessage);
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
```

```
            content: Text(snackbarMessage),
          ),
        );
      }
    }
  }
}
```

2. REGISTER SCREEN SOURCE CODE

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/exceptions/auth/firebase_exception.dart';
import 'package:flutter_online_shop/exceptions/auth/sign_up_exception.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/src/sign_in/sign_in_screen.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/circleContainer.dart';
import 'package:flutter_online_shop/widget/default_button.dart';
import 'package:future_progress_dialog/future_progress_dialog.dart';
import 'package:logger/logger.dart';

class SignUpScreen extends StatelessWidget {
  String dk =
      "Việc vi phạm chính sách này có thể dẫn tới một số hành động, bao gồm bất kỳ hoặc tất
cả các hành động sau:\n-Xóa danh mục sản phẩm;\n-Giới hạn quyền sử dụng Tài Khoản;\n-Đình
chỉ và chấm dứt Tài Khoản;\n-Thu hồi tiền/tài sản có được do hành vi gian lận, và các chi
phí có liên quan như chi phí vận chuyển của đơn hàng, phí thanh toán…;\n-Cáo buộc hình
sự;\n-Áp dụng biện pháp dân sự, bao gồm khiếu nại bồi thường thiệt hại và/hoặc áp dụng biện
pháp khẩn cấp tạm thời;\n-Các hành động hoặc biện pháp chế tài khác theo Tiêu Chuẩn Cộng
Đồng, Quy Chế Hoạt Động, hoặc các Chính Sách Globlux.\nNếu bạn phát hiện Người Sử Dụng trên
Trang Globlux của chúng tôi có hành vi vi phạm Điều Khoản Dịch Vụ, vui lòng liên hệ chúng
tôi.";
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      body: SafeArea(
        child: Container(
          height: SizeConfig.screenHeight,
          child: Stack(
            children: [
              Positioned(
                top:
```

```
      getProportionateScreenHeight(SizeConfig.screenHeight * .03),
  left: getProportionateScreenWidth(SizeConfig.screenWidth * .1),
  child: CircleContainer(
    color: kCircleContainer,
    sizeRate: .5,
  ),
),
Positioned(
  top:
      getProportionateScreenHeight(SizeConfig.screenHeight * .01),
  right:
      getProportionateScreenWidth(SizeConfig.screenWidth * .14),
  child: CircleContainer(
    color: kCircleContainer,
    sizeRate: .2,
  ),
),
Positioned(
  top:
      getProportionateScreenHeight(SizeConfig.screenHeight * .22),
  right:
      -getProportionateScreenWidth(SizeConfig.screenWidth * .3),
  child: CircleContainer(
    color: kCircleContainer,
    sizeRate: .6,
  ),
),
Positioned(
  bottom: -getProportionateScreenHeight(
      SizeConfig.screenHeight * .05),
  left:
      -getProportionateScreenWidth(SizeConfig.screenWidth * .38),
  child: CircleContainer(
    color: kCircleContainer,
    sizeRate: 1.2,
  ),
),
SingleChildScrollView(
  physics: BouncingScrollPhysics(),
  child: SizedBox(
    width: double.infinity,
    child: Column(
      children: [
        SizedBox(height: SizeConfig.screenHeight * 0.02),
        Text(
          "Đăng kí".toUpperCase(),
          style: headingStyle,
        ),
```

```
                    Text(
                      "Điền đầy đủ thông tin để tiếp tục \ntạo tài khoản",
                      textAlign: TextAlign.center,
                    ),
                    SizedBox(height: SizeConfig.screenHeight * 0.03),
                    SignUpForm(),
                    SizedBox(height: getProportionateScreenHeight(20)),
                    Text(
                      "Việc xác nhận đăng kí tài khoản đồng nghĩa việc bạn\n chấp nhận
các điều khoản của chúng tôi",
                      textAlign: TextAlign.center,
                    ),
                    SizedBox(height: getProportionateScreenHeight(5)),
                    GestureDetector(
                      onTap: () {
                        showDialog(
                            context: context,
                            builder: (context) {
                              return AlertDialog(
                                actions: [
                                  FlatButton(
                                      onPressed: () {
                                        Navigator.of(context).pop();
                                      },
                                      child: Text('Tôi Đã Hiểu'))
                                ],
                                scrollable: true,
                                title: Text(
                                  "Điều khoản phần mềm Globlux",
                                  textAlign: TextAlign.center,
                                ),
                                content: SingleChildScrollView(
                                  child: Container(
                                    width: SizeConfig.screenWidth,
                                    child: Text(dk),
                                  ),
                                ),
                              );
                            });
                      },
                      child: Text(
                        "Xem thêm về điều khoản >>",
                        textAlign: TextAlign.center,
                        style: TextStyle(
                          color: kPrimaryColor,
                          fontWeight: FontWeight.bold,
                          fontFamily: 'Roboto',
                        ),
```

```
                    ),
                  ),
                  SizedBox(height: getProportionateScreenHeight(20)),
                  Row(
                    mainAxisAlignment: MainAxisAlignment.center,
                    crossAxisAlignment: CrossAxisAlignment.end,
                    children: [
                      SizedBox(width: SizeConfig.screenWidth * 0.03),
                      GestureDetector(
                        onTap: () => Navigator.push(
                            context,
                            MaterialPageRoute(
                              builder: (context) => SignInScreen(),
                            )),
                        child: Text(
                          'Đăng nhập',
                          style: TextStyle(
                              color: kPrimaryColor,
                              fontWeight: FontWeight.bold,
                              fontFamily: 'Roboto',
                              fontSize: getProportionateScreenWidth(20)),
                        ),
                      ),
                    ],
                  )
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}

class SignUpForm extends StatefulWidget {
  @override
  _SignUpFormState createState() => _SignUpFormState();
}

class _SignUpFormState extends State<SignUpForm> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController emailFieldController = TextEditingController();
  final TextEditingController passwordFieldController = TextEditingController();
  final TextEditingController confirmPasswordFieldController =
      TextEditingController();
```

```
  @override
  void dispose() {
    emailFieldController.dispose();
    passwordFieldController.dispose();
    confirmPasswordFieldController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Padding(
        padding: EdgeInsets.symmetric(
            horizontal: getProportionateScreenWidth(screenPadding)),
        child: Column(
          children: [
            buildEmailFormField(),
            SizedBox(height: getProportionateScreenHeight(30)),
            buildPasswordFormField(),
            SizedBox(height: getProportionateScreenHeight(30)),
            buildConfirmPasswordFormField(),
            SizedBox(height: getProportionateScreenHeight(40)),
            Padding(
              padding: EdgeInsets.symmetric(
                  horizontal: getProportionateScreenWidth(screenPadding * 2)),
              child: DefaultButton(
                text: "Đăng Kí Mới",
                press: signUpButtonCallback,
              ),
            ),
          ],
        ),
      ),
    );
  }

//Edit Text ConfirmPassword
  Widget buildConfirmPasswordFormField() {
    return TextFormField(
      controller: confirmPasswordFieldController,
      obscureText: true,
      decoration: InputDecoration(
          labelText: "Confirm Password",
          hintText: 'Xác nhận mật khẩu',
          enabledBorder: OutlineInputBorder(
            borderRadius: BorderRadius.only(
```

```
            topLeft: Radius.circular(25),
            bottomRight: Radius.circular(25),
            topRight: Radius.circular(1),
            bottomLeft: Radius.circular(1),
          ),
        ),
        focusedBorder: OutlineInputBorder(
          borderRadius: BorderRadius.only(
            topLeft: Radius.circular(25),
            bottomRight: Radius.circular(25),
            topRight: Radius.circular(1),
            bottomLeft: Radius.circular(1),
          ),
        ),
        labelStyle: TextStyle(color: Colors.black87),
        floatingLabelBehavior: FloatingLabelBehavior.auto,
        prefixIcon: Icon(
          Icons.lock,
          color: kPrimaryColor,
        )),
      validator: (value) {
        // if (confirmPasswordFieldController.text.isEmpty) {
        //   return kPassNullError;
        // } else
        if (confirmPasswordFieldController.text !=
                passwordFieldController.text &&
            confirmPasswordFieldController.text.isNotEmpty) {
          return kMatchPassError;
        } else if (confirmPasswordFieldController.text.length < 8 &&
            confirmPasswordFieldController.text.isNotEmpty) {
          return kShortPassError;
        }
        return null;
      },
      autovalidateMode: AutovalidateMode.onUserInteraction,
    );
  }

//EditText Email
  Widget buildEmailFormField() {
    return TextFormField(
      controller: emailFieldController,
      keyboardType: TextInputType.emailAddress,
      decoration: InputDecoration(
        hintText: 'Tài khoản email',
        enabledBorder: OutlineInputBorder(
          borderRadius: BorderRadius.only(
            topLeft: Radius.circular(25),
```

```
            bottomRight: Radius.circular(25),
            topRight: Radius.circular(1),
            bottomLeft: Radius.circular(1),
          ),
        ),
        focusedBorder: OutlineInputBorder(
          borderRadius: BorderRadius.only(
            topLeft: Radius.circular(25),
            bottomRight: Radius.circular(25),
            topRight: Radius.circular(1),
            bottomLeft: Radius.circular(1),
          ),
        ),
        labelStyle: TextStyle(color: Colors.black87),
        labelText: "Email",
        floatingLabelBehavior: FloatingLabelBehavior.auto,
        prefixIcon: Icon(
          Icons.mail,
          color: kPrimaryColor,
        ),
      ),
      validator: (value) {
        if (!emailValidatorRegExp.hasMatch(emailFieldController.text) &&
            emailFieldController.text.isNotEmpty) {
          return kInvalidEmailError;
        }
        return null;
      },
      autovalidateMode: AutovalidateMode.onUserInteraction,
    );
  }

//EditText Password
  Widget buildPasswordFormField() {
    return TextFormField(
      controller: passwordFieldController,
      obscureText: true,
      decoration: InputDecoration(
          hintText: 'Mật khẩu',
          enabledBorder: OutlineInputBorder(
            borderRadius: BorderRadius.only(
              topLeft: Radius.circular(25),
              bottomRight: Radius.circular(25),
              topRight: Radius.circular(1),
              bottomLeft: Radius.circular(1),
            ),
          ),
          focusedBorder: OutlineInputBorder(
```

```
            borderRadius: BorderRadius.only(
              topLeft: Radius.circular(25),
              bottomRight: Radius.circular(25),
              topRight: Radius.circular(1),
              bottomLeft: Radius.circular(1),
            ),
          ),
          labelStyle: TextStyle(color: Colors.black87),
          labelText: "Password",
          floatingLabelBehavior: FloatingLabelBehavior.auto,
          prefixIcon: Icon(
            Icons.lock,
            color: kPrimaryColor,
          )),
      validator: (value) {        if (passwordFieldController.text.length < 8 &&
            passwordFieldController.text.isNotEmpty) {
          return kShortPassError;
        }
        return null;
      },
      autovalidateMode: AutovalidateMode.onUserInteraction,
    );
  }

  Future<void> signUpButtonCallback() async {
    if (_formKey.currentState.validate()) {
      final AuthentificationService authService = AuthentificationService();
      bool signUpStatus = false;
      String snackbarMessage;
      try {
        final signUpFuture = authService.signUp(
          email: emailFieldController.text,
          password: passwordFieldController.text,
        );
        signUpFuture.then((value) => signUpStatus = value);
        signUpStatus = await showDialog(
          context: context,
          builder: (context) {
            return FutureProgressDialog(
              signUpFuture,
              message: Text("Hệ thống đang tạo tài khoản"),
            );
          },
        );
        if (signUpStatus == true) {
          snackbarMessage =
              "Tạo tài khoản thành công,bạn cần kích hoạt tài khoản để tiếp tục các chức
năng";
```

```
        } else {
          throw FirebaseSignUpAuthUnknownReasonFailureException();
        }
      } on MessagedFirebaseAuthException catch (e) {
        snackbarMessage = e.message;
      } catch (e) {
        snackbarMessage = e.toString();
      } finally {
        Logger().i(snackbarMessage);
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text(snackbarMessage),
          ),
        );
        if (signUpStatus == true) {
          Navigator.pop(context);
        }
      }
    }
  }
}
```

3. FORGOT PASSWORD SCREEN SOURCE CODE

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/exceptions/auth/actions_exception.dart';
import 'package:flutter_online_shop/exceptions/auth/firebase_exception.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/src/sign_in/sign_in_screen.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/circleContainer.dart';
import 'package:flutter_online_shop/widget/default_button.dart';
import 'package:flutter_online_shop/widget/no_acc_text.dart';
import 'package:future_progress_dialog/future_progress_dialog.dart';
import 'package:logger/logger.dart';

class ForgotPasswordScreen extends StatelessWidget {
  static const String routeName = "/forgot_password";
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      body: SafeArea(
```

```
child: Container(
  height: SizeConfig.screenHeight,
  child: Stack(
    children: [
      Positioned(
        top: getProportionateScreenHeight(SizeConfig.screenHeight * .3),
        right:
            getProportionateScreenWidth(SizeConfig.screenWidth * .05),
        child: CircleContainer(
          color: kCircleContainer,
          sizeRate: .4,
        ),
      ),
      Positioned(
        bottom:
            getProportionateScreenHeight(SizeConfig.screenHeight * .05),
        left: getProportionateScreenWidth(SizeConfig.screenWidth * .0),
        child: CircleContainer(
          color: kCircleContainer,
          sizeRate: .4,
        ),
      ),
      Positioned(
        top:
            getProportionateScreenHeight(SizeConfig.screenHeight * .01),
        left: -getProportionateScreenWidth(SizeConfig.screenWidth * .2),
        child: CircleContainer(
          color: kCircleContainer,
          sizeRate: .8,
        ),
      ),
      Positioned(
        bottom:
            getProportionateScreenHeight(SizeConfig.screenHeight * .1),
        right:
            -getProportionateScreenWidth(SizeConfig.screenWidth * .05),
        child: CircleContainer(
          color: kCircleContainer,
          sizeRate: .75,
        ),
      ),
      SingleChildScrollView(
        physics: BouncingScrollPhysics(),
        child: Padding(
          padding: EdgeInsets.symmetric(
              horizontal: getProportionateScreenWidth(screenPadding)),
          child: SizedBox(
            width: double.infinity,
```

```
                    child: Column(
                      children: [
                        SizedBox(height: SizeConfig.screenHeight * 0.04),
                        Text(
                          "Quên mật khẩu?",
                          textAlign: TextAlign.center,
                          style: headingStyle,
                        ),
                        Text(
                          "Đừng lo lắng,hãy cho chúng tôi biết email của bạn,chúng tôi sẽ
giúp bạn phục hồi tài khoản",
                          textAlign: TextAlign.center,
                        ),
                        SizedBox(height: SizeConfig.screenHeight * 0.1),
                        ForgotPasswordForm(),
                      ],
                    ),
                  ),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}

class ForgotPasswordForm extends StatefulWidget {
  @override
  _ForgotPasswordFormState createState() => _ForgotPasswordFormState();
}

class _ForgotPasswordFormState extends State<ForgotPasswordForm> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController emailFieldController = TextEditingController();
  @override
  void dispose() {
    emailFieldController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Column(
        children: [
```

```
        buildEmailFormField(),
        SizedBox(height: getProportionateScreenHeight(30)),
        SizedBox(height: SizeConfig.screenHeight * 0.1),
        Padding(
          padding: EdgeInsets.symmetric(
              horizontal: getProportionateScreenWidth(screenPadding * 2)),
          child: DefaultButton(
            text: "Gửi email xác minh",
            press: sendVerificationEmailButtonCallback,
          ),
        ),
        SizedBox(height: SizeConfig.screenHeight * 0.03),
        NoAccountText(),
        SizedBox(height: getProportionateScreenHeight(30)),
        backToSignIn(),
      ],
    ),
  );
}

Widget backToSignIn() {
  return Row(
    mainAxisAlignment: MainAxisAlignment.center,
    crossAxisAlignment: CrossAxisAlignment.end,
    children: [
      SizedBox(width: SizeConfig.screenWidth * 0.03),
      GestureDetector(
        onTap: () => Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => SignInScreen(),
            )),
        child: Text(
          'Trở về',
          style: TextStyle(
              color: kPrimaryColor,
              fontWeight: FontWeight.bold,
              fontFamily: 'Roboto',
              fontSize: getProportionateScreenWidth(15)),
        ),
      ),
    ],
  );
}

TextFormField buildEmailFormField() {
  return TextFormField(
    controller: emailFieldController,
```

```
    keyboardType: TextInputType.emailAddress,
    decoration: InputDecoration(
        enabledBorder: OutlineInputBorder(
          borderRadius: BorderRadius.only(
            topLeft: Radius.circular(25),
            bottomRight: Radius.circular(25),
            topRight: Radius.circular(1),
            bottomLeft: Radius.circular(1),
          ),
        ),
        focusedBorder: OutlineInputBorder(
          borderRadius: BorderRadius.only(
            topLeft: Radius.circular(25),
            bottomRight: Radius.circular(25),
            topRight: Radius.circular(1),
            bottomLeft: Radius.circular(1),
          ),
        ),
        labelStyle: TextStyle(color: Colors.black87),
        labelText: "Email",
        hintText: 'Tài khoản email',
        floatingLabelBehavior: FloatingLabelBehavior.auto,
        prefixIcon: Icon(
          Icons.mail,
          color: kPrimaryColor,
        )),
    validator: (value) {
      if (value.isEmpty) {
        return kEmailNullError;
      } else if (!emailValidatorRegExp.hasMatch(value)) {
        return kInvalidEmailError;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Future<void> sendVerificationEmailButtonCallback() async {
  if (_formKey.currentState.validate()) {
    _formKey.currentState.save();
    final String emailInput = emailFieldController.text.trim();
    bool resultStatus;
    String snackbarMessage;
    try {
      final resultFuture =
          AuthentificationService().resetPasswordForEmail(emailInput);
      resultFuture.then((value) => resultStatus = value);
```

```
        resultStatus = await showDialog(
          context: context,
          builder: (context) {
            return FutureProgressDialog(
              resultFuture,
              message: Text("Đang tiến hành gửi email xác thực"),
            );
          },
        );
        if (resultStatus == true) {
          snackbarMessage = "Đã gửi đường dẫn khôi phục mật khẩu";
        } else {
          throw FirebaseCredentialActionAuthUnknownReasonFailureException(
              message: "Thao tác thất bại,vui lòng thử lại");
        }
      } on MessagedFirebaseAuthException catch (e) {
        snackbarMessage = e.message;
      } catch (e) {
        snackbarMessage = e.toString();
      } finally {
        Logger().i(snackbarMessage);
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text(snackbarMessage),
          ),
        );
      }
    }
  }
}
```

4. HOME SCREEN SOURCE CODE

Home Screen chứa các widget con gồm :

1. body.dart (Tổng hợp tất cả widget con)
2. categories_card.dart(Từng thẻ danh mục sản phẩm)
3. categories.dart(Danh mục sản phẩm)
4. drawer.dart(Menu công cụ)
5. header.dart(Gồm : Phần tìm kiếm và giỏ hàng)
6. product_type_box.dart(Card item của danh mục)

7. products_sections.dart(List tất cả sản phẩm)
8. section_tile.dart(Tiêu đề của list sản phẩm)
9. slideRatingItem.dart(Các item sản phẩm theo đánh giá)
10. slideRatingList.dart(List Sản phẩm theo đánh giá)
11. title_container.dart(Widget Tiêu đề)

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/src/home/widget/body.dart';
import 'package:flutter_online_shop/src/home/widget/drawer.dart';
import 'package:flutter_online_shop/utils/size_config.dart';

class HomeScreen extends StatelessWidget {
  static const String routeName = "/home";
  @override
  Widget build(BuildContext context) {
    SizeConfig().init(context);
    return Scaffold(
      body: GestureDetector(
        onTap: () => FocusManager.instance.primaryFocus?.unfocus(),
        child: Stack(
          children: [
            HomeBody(),
          ],
        ),
      ),
      drawer: HomeScreenDrawer(),
    );
  }
}
```

HOME SCREEN WIDGET SOURCE CODE

4.1. BODY.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/services/steams/all_product_stream.dart';
import 'package:flutter_online_shop/services/steams/favourite_product_stream.dart';
import 'package:flutter_online_shop/src/cart/cart_screen.dart';
import 'package:flutter_online_shop/src/category/category_screen.dart';
import 'package:flutter_online_shop/src/home/widget/header.dart';
import 'package:flutter_online_shop/src/home/widget/product_type_box.dart';
```

```
import 'package:flutter_online_shop/src/home/widget/products_sections.dart';
import 'package:flutter_online_shop/src/home/widget/slideRatingList.dart';
import 'package:flutter_online_shop/src/product_details/product_details_screen.dart';
import 'package:flutter_online_shop/src/search/search_screen.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/utils/utils.dart';
import 'package:future_progress_dialog/future_progress_dialog.dart';
import 'package:logger/logger.dart';

const String ICON_KEY = "icon";
const String TITLE_KEY = "title";
const String PRODUCT_TYPE_KEY = "product_type";

class HomeBody extends StatefulWidget {
  @override
  _HomeBodyState createState() => _HomeBodyState();
}

class _HomeBodyState extends State<HomeBody> {
  final productCategories = <Map>[
    <String, dynamic>{
      ICON_KEY: "assets/icons/Electronics.svg",
      TITLE_KEY: "Đồ điện tử",
      PRODUCT_TYPE_KEY: ProductType.Electronics,
    },
    <String, dynamic>{
      ICON_KEY: "assets/icons/Books.svg",
      TITLE_KEY: "Sách",
      PRODUCT_TYPE_KEY: ProductType.Books,
    },
    <String, dynamic>{
      ICON_KEY: "assets/icons/Fashion.svg",
      TITLE_KEY: "Thời trang",
      PRODUCT_TYPE_KEY: ProductType.Fashion,
    },
    <String, dynamic>{
      ICON_KEY: "assets/icons/Groceries.svg",
      TITLE_KEY: "Thực phẩm",
      PRODUCT_TYPE_KEY: ProductType.Groceries,
    },
    <String, dynamic>{
      ICON_KEY: "assets/icons/Art.svg",
      TITLE_KEY: "Đồ trang trí",
      PRODUCT_TYPE_KEY: ProductType.Furniture,
    },
    <String, dynamic>{
      ICON_KEY: "assets/icons/Others.svg",
```

```
    TITLE_KEY: "Khác",
    PRODUCT_TYPE_KEY: ProductType.Others,
  },
];

bool isType = false;

final FavouriteProductsStream favouriteProductsStream =
    FavouriteProductsStream();
final AllProductsStream allProductsStream = AllProductsStream();
final AllProductsStream allRatingProductsStream = AllProductsStream();

@override
void initState() {
  super.initState();
  //favouriteProductsStream.init();
  allProductsStream.init();
  allRatingProductsStream.init();
}

@override
void dispose() {
  //favouriteProductsStream.dispose();
  allProductsStream.dispose();
  allRatingProductsStream.init();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Stack(
    children: [
      SafeArea(
        child: RefreshIndicator(
          onRefresh: refreshPage,
          child: SingleChildScrollView(
            physics: AlwaysScrollableScrollPhysics(),
            child: Padding(
              padding: EdgeInsets.symmetric(
                  horizontal: getProportionateScreenWidth(screenPadding)),
              child: Column(
                mainAxisSize: MainAxisSize.max,
                mainAxisAlignment: MainAxisAlignment.spaceBetween,
                children: [
                  SizedBox(height: getProportionateScreenHeight(15)),
                  HomeHeader(
                    onSearchSubmitted: (value) async {
                      final query = value.toString();
```

```
                if (query.length <= 0) return;
                List<String> searchedProductsId;
                try {
                  searchedProductsId = await ProductDatabaseHelper()
                      .searchInProducts(query.toLowerCase());
                  if (searchedProductsId != null) {
                    await Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => SearchResultScreen(
                          searchQuery: query,
                          searchResultProductsId: searchedProductsId,
                          searchIn: "Tất cả sản phẩm",
                        ),
                      ),
                    );
                    await refreshPage();
                  } else {
                    throw "Thao tác thất bại,vui lòng thử lại";
                  }
                } catch (e) {
                  final error = e.toString();
                  Logger().e(error);
                  ScaffoldMessenger.of(context).showSnackBar(
                    SnackBar(
                      content: Text("$error"),
                    ),
                  );
                }
              },
              onCartButtonPressed: () async {
                bool allowed =
                    AuthentificationService().currentUserVerified;
                if (!allowed) {
                  final reverify = await showConfirmationDialog(context,
                      "Bạn chưa xác minh địa chỉ email của mình. Hành động này chỉ
được phép đối với những người dùng đã xác minh.",
                      positiveResponse: "Gửi email xác minh",
                      negativeResponse: "Trở về");
                  if (reverify) {
                    final future = AuthentificationService()
                        .sendVerificationEmailToCurrentUser();
                    await showDialog(
                      context: context,
                      builder: (context) {
                        return FutureProgressDialog(
                          future,
                          message: Text(
```

```
                                "Đang tiến hành gửi xác minh.\nVui lòng đợi trong
giây lát"),
                              );
                            },
                          );
                        }
                        return;
                      }
                      await Navigator.push(
                        context,
                        MaterialPageRoute(
                          builder: (context) => CartScreen(),
                        ),
                      );
                      await refreshPage();
                    },
                  ),
                  SizedBox(height: getProportionateScreenHeight(10)),
                  Padding(
                    padding: EdgeInsets.only(left: 5),
                    child: Row(
                      mainAxisAlignment: MainAxisAlignment.start,
                      children: [
                        Text(
                          "Danh mục",
                          style: TextStyle(
                              fontSize: getProportionateScreenHeight(20),
                              color: kPrimaryColor,
                              fontFamily: 'Roboto'),
                        ),
                      ],
                    ),
                  ),
                  SizedBox(height: getProportionateScreenHeight(10)),
                  SizedBox(
                    height: SizeConfig.screenHeight * 0.1,
                    child: Padding(
                      padding: const EdgeInsets.symmetric(vertical: 4),
                      child: ListView(
                        scrollDirection: Axis.horizontal,
                        physics: BouncingScrollPhysics(),
                        children: [
                          ...List.generate(
                            productCategories.length,
                            (index) {
                              return ProductTypeBox(
                                icon: productCategories[index][ICON_KEY],
                                title: productCategories[index][TITLE_KEY],
```

```
                onPress: () {
                  Navigator.push(
                    context,
                    MaterialPageRoute(
                      builder: (context) =>
                          CategoryProductsScreen(
                        productType: productCategories[index]
                            [PRODUCT_TYPE_KEY],
                      ),
                    ),
                  );
                },
              );
            },
          ),
        ],
      ),
    ),
  ),
  SizedBox(height: getProportionateScreenHeight(10)),
  SizedBox(
    height: SizeConfig.screenHeight * 0.4,
    child: SlideRatingList(
      sectionTitle: "Đánh giá về sản phẩm",
      productsStreamController: allRatingProductsStream,
      emptyListMessage: "Hiện không có sản phẩm nào",
      onProductCardTapped: onProductCardTapped,
    ),
  ),
  SizedBox(height: getProportionateScreenHeight(5)),
  Row(
    mainAxisAlignment: MainAxisAlignment.end,
    children: [
      Container(
        margin: EdgeInsets.only(
            top: getProportionateScreenHeight(5)),
        width: SizeConfig.screenWidth * .3,
        height: SizeConfig.screenHeight * .05,
        decoration: BoxDecoration(
            color: kPrimaryColor.withOpacity(.2),
            borderRadius: BorderRadius.only(
              topLeft: Radius.circular(20),
              topRight: Radius.circular(20),
            )),
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            IconButton(
```

```
                                icon: Icon(Icons.grid_view),
                                onPressed: () {
                                  setState(() {
                                    isType = false;
                                  });
                                }),
                            IconButton(
                                icon: Icon(Icons.list),
                                onPressed: () {
                                  setState(() {
                                    isType = true;
                                  });
                                }),
                          ],
                        ),
                      ),
                    ],
                  ),
                  SizedBox(
                    height: SizeConfig.screenHeight * 0.8,
                    child: ProductsSection(
                      type: isType,
                      sectionTitle: "Tất cả sản phẩm",
                      productsStreamController: allProductsStream,
                      emptyListMessage: "Hiện tại chưa có sản phẩm nào cả",
                      onProductCardTapped: onProductCardTapped,
                    ),
                  ),
                  SizedBox(height: getProportionateScreenHeight(80)),
                ],
              ),
            ),
          ),
        ),
      ),
    ],
  );
}

Future<void> refreshPage() {
  //favouriteProductsStream.reload();
  allProductsStream.reload();
  return Future<void>.value();
}

void onProductCardTapped(String productId) {
  Navigator.push(
    context,
```

```
      MaterialPageRoute(
        builder: (context) => ProductDetailsScreen(productId: productId),
      ),
    ).then((_) async {
      await refreshPage();
    });
  }
}
```

4.2. CATEGORIES_CARD.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_svg/flutter_svg.dart';

class CategoryCard extends StatelessWidget {
  final String icon;
  final String text;
  final GestureTapCallback press;
  const CategoryCard({
    Key key,
    @required this.icon,
    @required this.text,
    @required this.press,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: press,
      child: SizedBox(
        width: getProportionateScreenWidth(55),
        child: Column(
          children: [
            AspectRatio(
              aspectRatio: 1,
              child: Container(
                padding: EdgeInsets.all(getProportionateScreenWidth(15)),
                decoration: BoxDecoration(
                  color: Color(0xFFFFECDF),
                  borderRadius: BorderRadius.circular(10),
                ),
                child: SvgPicture.asset(icon),
              ),
            ),
```

```
            const SizedBox(height: 5),
            Text(
              text,
              textAlign: TextAlign.center,
            ),
          ],
        ),
      ),
    );
  }
}
```

4.3. CATEGORIES.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/src/home/widget/categories_card.dart';

class Categories extends StatelessWidget {
  final List<Map<String, dynamic>> categories = [
    {"icon": "assets/icons/Flash Icon.svg", "text": "Flash Deal"},
    {"icon": "assets/icons/Bill Icon.svg", "text": "Bill"},
    {"icon": "assets/icons/Game Icon.svg", "text": "Game"},
    {"icon": "assets/icons/Gift Icon.svg", "text": "Daily Gift"},
    {"icon": "assets/icons/Discover.svg", "text": "More"},
  ];
  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      crossAxisAlignment: CrossAxisAlignment.start,
      children: List.generate(
        categories.length,
        (index) => CategoryCard(
          icon: categories[index]["icon"],
          text: categories[index]["text"],
          press: () {},
        ),
      ),
    );
  }
}
```

4.4. DRAWER.DART

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/src/about/about_screen.dart';
```

```
import 'package:flutter_online_shop/src/change_infomation/change_email.dart';
import 'package:flutter_online_shop/src/change_infomation/change_name.dart';
import 'package:flutter_online_shop/src/change_infomation/change_password.dart';
import 'package:flutter_online_shop/src/edit_product/edit_product_screen.dart';
import 'package:flutter_online_shop/src/manager_address/manager_address_screen.dart';
import 'package:flutter_online_shop/src/orders/my_orders_screen.dart';
import 'package:flutter_online_shop/src/product/my_product_screen.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/utils/utils.dart';
import 'package:future_progress_dialog/future_progress_dialog.dart';

class HomeScreenDrawer extends StatelessWidget {
  const HomeScreenDrawer({
    Key key,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: ListView(
        physics: BouncingScrollPhysics(),
        children: [
          StreamBuilder<User>(
              stream: AuthentificationService().userChanges,
              builder: (context, snapshot) {
                if (snapshot.hasData) {
                  final user = snapshot.data;
                  return buildUserAccountsHeader(user);
                } else if (snapshot.connectionState ==
                    ConnectionState.waiting) {
                  return Center(
                    child: CircularProgressIndicator(),
                  );
                } else {
                  return Center(
                    child: Icon(Icons.error),
                  );
                }
              }),
          buildEditAccountExpansionTile(context),
          ListTile(
            leading: Icon(
              Icons.edit_location,
              color: kPrimaryColor,
            ),
            title: Text(
              "Địa chỉ",
```

```
            style: TextStyle(fontSize: 16, color: Colors.black),
          ),
          onTap: () async {
            bool allowed = AuthentificationService().currentUserVerified;
            if (!allowed) {
              final reverify = await showConfirmationDialog(context,
                  "Bạn chưa xác minh địa chỉ email của mình. Hành động này chỉ được phép
đối với những người dùng đã xác minh.",
                  positiveResponse: "Gửi email xác minh",
                  negativeResponse: "Trở về");
              if (reverify) {
                final future = AuthentificationService()
                    .sendVerificationEmailToCurrentUser();
                await showDialog(
                  context: context,
                  builder: (context) {
                    return FutureProgressDialog(
                      future,
                      message: Text(
                          "Đang tiến hành gửi xác minh.\nVui lòng đợi trong giây lát"),
                    );
                  },
                );
              }
              return;
            }
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => ManageAddressesScreen(),
              ),
            );
          },
        ),
        ListTile(
          leading: Icon(
            Icons.grid_view,
            color: kPrimaryColor,
          ),
          title: Text(
            "Đơn đặt hàng",
            style: TextStyle(fontSize: 16, color: Colors.black),
          ),
          onTap: () async {
            bool allowed = AuthentificationService().currentUserVerified;
            if (!allowed) {
              final reverify = await showConfirmationDialog(context,
```

```
                "Bạn chưa xác minh địa chỉ email của mình. Hành động này chỉ được phép
đối với những người dùng đã xác minh.",
                positiveResponse: "Gửi email xác minh",
                negativeResponse: "Trở về");
            if (reverify) {
              final future = AuthentificationService()
                  .sendVerificationEmailToCurrentUser();
              await showDialog(
                context: context,
                builder: (context) {
                  return FutureProgressDialog(
                    future,
                    message: Text(
                        "Đang tiến hành gửi xác minh.\nVui lòng đợi trong giây lát"),
                  );
                },
              );
            }
            return;
          }
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => MyOrdersScreen(),
            ),
          );
        },
      ),
      buildSellerExpansionTile(context),
      ListTile(
        leading: Icon(
          Icons.info,
          color: kPrimaryColor,
        ),
        title: Text(
          "About Us",
          style: TextStyle(fontSize: 16, color: Colors.black),
        ),
        onTap: () async {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => AboutDeveloperScreen(),
            ),
          );
        },
      ),
      ListTile(
```

```
          leading: Icon(
            Icons.logout,
            color: kPrimaryColor,
          ),
          title: Text(
            "Đăng xuất",
            style: TextStyle(fontSize: 16, color: Colors.black),
          ),
          onTap: () async {
            final confirmation = await showConfirmationDialog(
                context, "Bạn có chắc muốn đăng xuất không?");
            if (confirmation) AuthentificationService().signOut();
          },
        ),
      ],
    ),
  );
}

UserAccountsDrawerHeader buildUserAccountsHeader(User user) {
  return UserAccountsDrawerHeader(
    margin: EdgeInsets.zero,
    decoration: BoxDecoration(
      color: kPrimaryColor.withOpacity(0.5),
    ),
    accountEmail: Text(
      user.email ?? "No Email",
      style: TextStyle(
        fontSize: 15,
        color: Colors.black,
      ),
    ),
    accountName: Text(
      user.displayName ?? "Mr.No-Name",
      style: TextStyle(
        fontSize: 18,
        fontWeight: FontWeight.w800,
        color: Colors.black,
      ),
    ),
  );
}

ExpansionTile buildEditAccountExpansionTile(BuildContext context) {
  return ExpansionTile(
    leading: Icon(
      Icons.person,
      color: kPrimaryColor,
```

```
),
title: Text(
  "Chỉnh sửa tài khoản",
  style: TextStyle(fontSize: 16, color: Colors.black),
),
children: [
  Padding(
    padding: EdgeInsets.only(
        left: getProportionateScreenWidth(screenPadding * 1.5)),
    child: ListTile(
      title: Row(
        children: [
          Icon(
            Icons.title,
            color: kPrimaryColor,
          ),
          SizedBox(
            width: SizeConfig.screenWidth * .05,
          ),
          Text(
            "Thay đổi Tên",
            style: TextStyle(
              color: Colors.black,
              fontSize: 15,
            ),
          ),
        ],
      ),
      onTap: () {
        Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => ChangeDisplayNameScreen(),
            ));
      },
    ),
  ),
  Padding(
    padding: EdgeInsets.only(
        left: getProportionateScreenWidth(screenPadding * 1.6)),
    child: ListTile(
      title: Row(
        children: [
          Icon(
            Icons.mail_outline_sharp,
            color: kPrimaryColor,
          ),
          SizedBox(
```

```
            width: SizeConfig.screenWidth * .05,
          ),
          Text(
            "Thay đổi địa chỉ Email",
            style: TextStyle(
              color: Colors.black,
              fontSize: 15,
            ),
          ),
        ],
      ),
      onTap: () {
        Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => ChangeEmailScreen(),
            ));
      },
    ),
  ),
  Padding(
    padding: EdgeInsets.only(
        left: getProportionateScreenWidth(screenPadding * 1.6)),
    child: ListTile(
      title: Row(
        children: [
          Icon(
            Icons.lock_open,
            color: kPrimaryColor,
          ),
          SizedBox(
            width: SizeConfig.screenWidth * .05,
          ),
          Text(
            "Thay đổi mật khẩu",
            style: TextStyle(
              color: Colors.black,
              fontSize: 15,
            ),
          ),
        ],
      ),
      onTap: () {
        Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => ChangePasswordScreen(),
            ));
```

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

  Widget buildSellerExpansionTile(BuildContext context) {
    return ExpansionTile(
      leading: Icon(
        Icons.shop_two,
        color: kPrimaryColor,
      ),
      title: Text(
        "Chế độ bán hàng",
        style: TextStyle(fontSize: 16, color: Colors.black),
      ),
      children: [
        Padding(
          padding: EdgeInsets.only(
              left: getProportionateScreenWidth(screenPadding * 1.6)),
          child: ListTile(
            title: Row(
              children: [
                Icon(
                  Icons.add_business,
                  color: kPrimaryColor,
                ),
                SizedBox(
                  width: SizeConfig.screenWidth * .05,
                ),
                Text(
                  "Thêm sản phẩm",
                  style: TextStyle(
                    color: Colors.black,
                    fontSize: 15,
                  ),
                ),
              ],
            ),
            onTap: () async {
              bool allowed = AuthentificationService().currentUserVerified;
              if (!allowed) {
                final reverify = await showConfirmationDialog(context,
                    "Bạn chưa xác minh địa chỉ email của mình. Hành động này chỉ được phép
đối với những người dùng đã xác minh.",
                    positiveResponse: "Gửi email xác minh",
                    negativeResponse: "Trở về");
```

```
        if (reverify) {
          final future = AuthentificationService()
              .sendVerificationEmailToCurrentUser();
          await showDialog(
            context: context,
            builder: (context) {
              return FutureProgressDialog(
                future,
                message: Text(
                    "Đang tiến hành gửi xác minh.\nVui lòng đợi trong giây lát"),
              );
            },
          );
        }
        return;
      }
      Navigator.push(context,
          MaterialPageRoute(builder: (context) => EditProductScreen()));
    },
  ),
),
Padding(
  padding: EdgeInsets.only(
      left: getProportionateScreenWidth(screenPadding * 1.6)),
  child: ListTile(
    title: Row(
      children: [
        Icon(
          Icons.mode_edit,
          color: kPrimaryColor,
        ),
        SizedBox(
          width: SizeConfig.screenWidth * .05,
        ),
        Text(
          "Quản lý sản phẩm",
          style: TextStyle(
            color: Colors.black,
            fontSize: 15,
          ),
        ),
      ],
    ),
    onTap: () async {
      bool allowed = AuthentificationService().currentUserVerified;
      if (!allowed) {
        final reverify = await showConfirmationDialog(context,
```

```
                    "Bạn chưa xác minh địa chỉ email của mình. Hành động này chỉ được phép
đối với những người dùng đã xác minh.",
                    positiveResponse: "Gửi email xác minh",
                    negativeResponse: "Trở về");
                if (reverify) {
                  final future = AuthentificationService()
                      .sendVerificationEmailToCurrentUser();
                  await showDialog(
                    context: context,
                    builder: (context) {
                      return FutureProgressDialog(
                        future,
                        message: Text(
                            "Đang tiến hành gửi xác minh.\nVui lòng đợi trong giây lát"),
                      );
                    },
                  );
                }
                return;
              }
              Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => MyProductsScreen(),
                ),
              );
            },
          ),
        ),
      ],
    );
  }
}
```

4.5. HEADER.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/widget/icon_button_counter.dart';
import 'package:flutter_online_shop/widget/round_icon_button.dart';
import 'package:flutter_online_shop/widget/search_text_field.dart';

class HomeHeader extends StatelessWidget {
  final Function onSearchSubmitted;
  final Function onCartButtonPressed;
  const HomeHeader({
```

```
    Key key,
    @required this.onSearchSubmitted,
    @required this.onCartButtonPressed,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: [
        RoundedIconButton(
            iconData: Icons.menu,
            press: () {
              Scaffold.of(context).openDrawer();
            }),
        Expanded(
          child: SearchField(
            onSubmit: onSearchSubmitted,
          ),
        ),
        SizedBox(width: 10),
        Container(
          decoration: BoxDecoration(
              color: kPrimaryColor.withOpacity(.4), shape: BoxShape.circle),
          child: IconButtonWithCounter(
            svgSrc: "assets/icons/Cart Icon.svg",
            numOfItems: 0,
            press: onCartButtonPressed,
          ),
        ),
      ],
    );
  }
}
```

4.6. PRODUCT_TYPE_BOX.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_svg/flutter_svg.dart';

class ProductTypeBox extends StatelessWidget {
  final String icon;
  final String title;
  final VoidCallback onPress;
```

```
const ProductTypeBox({
  Key key,
  @required this.icon,
  @required this.title,
  @required this.onPress,
}) : super(key: key);

@override
Widget build(BuildContext context) {
  return InkWell(
    onTap: onPress,
    child: AspectRatio(
      aspectRatio: 1.05,
      child: Container(
        margin: EdgeInsets.symmetric(
          horizontal: 4,
        ),
        padding: EdgeInsets.all(8),
        decoration: BoxDecoration(
          color: kPrimaryColor.withOpacity(0.09),
          borderRadius: BorderRadius.circular(15),
          border: Border.all(
            color: kPrimaryColor.withOpacity(0.3),
          ),
        ),
        child: Column(
          mainAxisAlignment: MainAxisAlignment.spaceBetween,
          children: [
            Expanded(
              child: Padding(
                padding: const EdgeInsets.all(6.0),
                child: AspectRatio(
                  aspectRatio: 1,
                  child: SvgPicture.asset(
                    icon,
                    color: Colors.green[900].withOpacity(.8),
                  ),
                ),
              ),
            ),
            SizedBox(height: 2),
            Text(
              title,
              style: TextStyle(
                color: Colors.green[900],
                fontSize: getProportionateScreenHeight(8),
                fontWeight: FontWeight.w900,
              ),
```

```
                maxLines: 1,
                overflow: TextOverflow.ellipsis,
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

4.7. PRODUCTS_SECTIONS.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/services/steams/data_stream.dart';
import 'package:flutter_online_shop/src/home/widget/section_tile.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/nothing_show.dart';
import 'package:flutter_online_shop/widget/product_card.dart';
import 'package:flutter_online_shop/widget/product_card_listview.dart';
import 'package:logger/logger.dart';

class ProductsSection extends StatelessWidget {
  final String sectionTitle;
  final DataStream productsStreamController;
  final String emptyListMessage;
  final Function onProductCardTapped;
  final bool type;

  const ProductsSection({
    Key key,
    @required this.sectionTitle,
    @required this.productsStreamController,
    this.emptyListMessage = "Hiện tại không có sản phẩm",
    @required this.onProductCardTapped,
    this.type = false,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.symmetric(
        horizontal: 14,
        vertical: 16,
```

```
    ),
    decoration: BoxDecoration(
      color: kPrimaryColor.withOpacity(.2),
      borderRadius: BorderRadius.only(
        topLeft: Radius.circular(15),
        bottomLeft: Radius.circular(15),
        bottomRight: Radius.circular(15),
      ),
    ),
    child: Column(
      children: [
        SectionTile(
          title: sectionTitle,
          press: () {},
        ),
        SizedBox(height: getProportionateScreenHeight(15)),
        Expanded(
          child: buildProductsList(),
        ),
      ],
    ),
  );
}

Widget buildProductsList() {
  return StreamBuilder<List<String>>(
    stream: productsStreamController.stream,
    builder: (context, snapshot) {
      if (snapshot.hasData) {
        if (snapshot.data.length == 0) {
          return Center(
            child: NothingToShowContainer(
              secondaryMessage: emptyListMessage,
            ),
          );
        }
        return !type
            ? buildProductGrid(snapshot.data)
            : buildProductList(snapshot.data);
      } else if (snapshot.connectionState == ConnectionState.waiting) {
        return Center(
          child: CircularProgressIndicator(),
        );
      } else if (snapshot.hasError) {
        final error = snapshot.error;
        Logger().w(error.toString());
      }
      return Center(
```

```
        child: NothingToShowContainer(
          iconPath: "assets/icons/network_error.svg",
          primaryMessage: "Something went wrong",
          secondaryMessage: "Unable to connect to Database",
        ),
      );
    },
  );
}

Widget buildProductGrid(List<String> productsId) {
  return GridView.builder(
    shrinkWrap: true,
    physics: BouncingScrollPhysics(),
    gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
      crossAxisCount: 2,
      childAspectRatio: 0.7,
      crossAxisSpacing: 4,
      mainAxisSpacing: 4,
    ),
    itemCount: productsId.length,
    itemBuilder: (context, index) {
      return ProductCard(
        productId: productsId[index],
        press: () {
          onProductCardTapped.call(productsId[index]);
        },
      );
    },
  );
}

Widget buildProductList(List<String> productsId) {
  return Padding(
    padding: EdgeInsets.only(right: getProportionateScreenWidth(50)),
    child: ListView.builder(
      shrinkWrap: true,
      itemCount: productsId.length,
      itemBuilder: (context, index) {
        return ProductCardListView(
          productId: productsId[index],
          press: () {
            onProductCardTapped.call(productsId[index]);
          },
        );
      },
    ),
  );
```

```
  }
}
```

4.8. SECTION_TILE.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/size_config.dart';

class SectionTile extends StatelessWidget {
  final String title;
  final GestureTapCallback press;
  const SectionTile({
    Key key,
    @required this.title,
    @required this.press,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      width: double.infinity,
      child: Text(
        title,
        style: TextStyle(
          color: Colors.black,
          fontFamily: 'Roboto',
          fontSize: getProportionateScreenWidth(21),
          fontWeight: FontWeight.bold,
        ),
      ),
    );
  }
```

4.9. SLIDERATINGLIST.DART

```
import 'package:carousel_slider/carousel_slider.dart';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/services/steams/data_stream.dart';
import 'package:flutter_online_shop/src/home/widget/section_tile.dart';
import 'package:flutter_online_shop/src/home/widget/slideRatingItem.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/nothing_show.dart';
import 'package:logger/logger.dart';
```

```
class SlideRatingList extends StatelessWidget {
  final String sectionTitle;
  final DataStream productsStreamController;
  final String emptyListMessage;
  final Function onProductCardTapped;

  const SlideRatingList(
      {Key key,
      this.sectionTitle,
      this.productsStreamController,
      this.emptyListMessage,
      this.onProductCardTapped})
      : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.symmetric(
        horizontal: 14,
        vertical: 16,
      ),
      decoration: BoxDecoration(
        color: kPrimaryColor.withOpacity(.2),
        borderRadius: BorderRadius.only(
          topLeft: Radius.circular(15),
          bottomLeft: Radius.circular(15),
          bottomRight: Radius.circular(15),
        ),
      ),
      child: Column(
        children: [
          SectionTile(
            title: sectionTitle,
            press: () {},
          ),
          SizedBox(height: getProportionateScreenHeight(15)),
          Expanded(
            child: buildProductsList(),
          ),
        ],
      ),
    );
  }

  Widget buildProductsList() {
    return StreamBuilder<List<String>>(
      stream: productsStreamController.stream,
```

```
    builder: (context, snapshot) {
      if (snapshot.hasData) {
        if (snapshot.data.length == 0) {
          return Center(
            child: NothingToShowContainer(
              secondaryMessage: emptyListMessage,
            ),
          );
        }
        return buildProductSlider(snapshot.data);
      } else if (snapshot.connectionState == ConnectionState.waiting) {
        return Center(
          child: CircularProgressIndicator(),
        );
      } else if (snapshot.hasError) {
        final error = snapshot.error;
        Logger().w(error.toString());
      }
      return Center(
        child: NothingToShowContainer(
          iconPath: "assets/icons/network_error.svg",
          primaryMessage: "Something went wrong",
          secondaryMessage: "Unable to connect to Database",
        ),
      );
    },
  );
}

Widget buildProductSlider(List<String> productsId) {
  return CarouselSlider.builder(
    itemCount: productsId.length,
    itemBuilder: (BuildContext context, int index, int i) {
      return Padding(
        padding: const EdgeInsets.symmetric(vertical: 10),
        child: SlideRatingItem(
          productId: productsId[index],
          press: () {
            onProductCardTapped.call(productsId[index]);
          },
        ),
      );
    },
    options: CarouselOptions(
      height: SizeConfig.screenHeight * .3,
      autoPlay: true,
      autoPlayInterval: Duration(seconds: 7),
      enlargeCenterPage: true,
```

```
        autoPlayCurve: Curves.fastOutSlowIn,
        autoPlayAnimationDuration: Duration(milliseconds: 800),
        enableInfiniteScroll: false,
      ),
    );
  }
}
```

4.10. SLIDERATINGITEM.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:logger/logger.dart';

class SlideRatingItem extends StatelessWidget {
  final String productId;
  final GestureTapCallback press;

  const SlideRatingItem({Key key, this.productId, this.press})
      : super(key: key);

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: press,
      child: Container(
        decoration: BoxDecoration(
          color: Colors.white,
          border: Border.all(color: kTextColor.withOpacity(0.15)),
          borderRadius: BorderRadius.all(
            Radius.circular(16),
          ),
        ),
        child: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 15, vertical: 10),
          child: FutureBuilder<Product>(
            future: ProductDatabaseHelper().getProductWithID(productId),
            builder: (context, snapshot) {
              if (snapshot.hasData) {
                final Product product = snapshot.data;
                return buildProductRatingCardItems(product);
              } else if (snapshot.connectionState == ConnectionState.waiting) {
                return Center(
                  child: Center(child: CircularProgressIndicator()),
```

```
            );
          } else if (snapshot.hasError) {
            final error = snapshot.error.toString();
            Logger().e(error);
          }
          return Center(
            child: Icon(
              Icons.error,
              color: kTextColor,
              size: 60,
            ),
          );
        },
      ),
    ),
  ),
);
}

Column buildProductRatingCardItems(Product product) {
  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Flexible(
        flex: 2,
        child: Row(
          children: [
            Padding(
              padding: const EdgeInsets.all(8.0),
              child: ClipRRect(
                borderRadius: BorderRadius.circular(10),
                child: Image.network(
                  product.images[0],
                  fit: BoxFit.fill,
                  width: SizeConfig.screenWidth * .3,
                ),
              ),
            ),
            SizedBox(
              width: getProportionateScreenWidth(20),
            ),
            Container(
              width: SizeConfig.screenWidth * .18,
              height: SizeConfig.screenWidth * .18,
              child: ClipOval(
                child: Container(
                  color: Colors.amber,
                  child: Center(
```

```
                    child: Text(
                      '★\t${product.rating}',
                      style: TextStyle(
                          color: Colors.white,
                          fontSize: getProportionateScreenHeight(20),
                          fontWeight: FontWeight.w700),
                    ),
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
      SizedBox(height: getProportionateScreenHeight(10)),
      Flexible(
        flex: 1,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Flexible(
              flex: 1,
              child: Text(
                "${product.title}\n",
                style: TextStyle(
                  color: kTextColor,
                  fontSize: 22,
                  fontWeight: FontWeight.bold,
                ),
                maxLines: 2,
                overflow: TextOverflow.ellipsis,
                textAlign: TextAlign.center,
              ),
            ),
          ],
        ),
      ),
    ],
  );
 }
}
```

4.11. TITLE_CONTAINER.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/constant.dart';
```

```
import 'package:flutter_online_shop/utils/size_config.dart';

class TitleContainer extends StatelessWidget {
  final String title;
  TitleContainer({this.title});

  @override
  Widget build(BuildContext context) {
    return Container(
      width: SizeConfig.screenWidth,
      height: SizeConfig.screenHeight * .08,
      decoration: BoxDecoration(
          color: kPrimaryColor,
          borderRadius: BorderRadius.only(
              topLeft: Radius.circular(30), topRight: Radius.circular(30))),
      child: Center(
        child: Text(
          title.toUpperCase(),
          style: titleStyle,
        ),
      ),
    );
  }
}
```

5. SEACH SCREEN SOURCE CODE

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/src/product_details/product_details_screen.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/nothing_show.dart';
import 'package:flutter_online_shop/widget/product_card.dart';

class SearchResultScreen extends StatelessWidget {
  final String searchQuery;
  final String searchIn;
  final List<String> searchResultProductsId;

  const SearchResultScreen({
    Key key,
    @required this.searchQuery,
    @required this.searchResultProductsId,
    @required this.searchIn,
  }) : super(key: key);
  @override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: SearchBody(
        searchQuery: searchQuery,
        searchResultProductsId: searchResultProductsId,
        searchIn: searchIn,
      ),
    );
  }
}

class SearchBody extends StatelessWidget {
  final String searchQuery;
  final List<String> searchResultProductsId;
  final String searchIn;

  const SearchBody({
    Key key,
    @required this.searchQuery,
    @required this.searchResultProductsId,
    @required this.searchIn,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: SingleChildScrollView(
        child: Padding(
          padding: EdgeInsets.symmetric(
              horizontal: getProportionateScreenWidth(screenPadding)),
          child: SizedBox(
            width: double.infinity,
            child: Column(
              children: [
                SizedBox(height: getProportionateScreenHeight(10)),
                Text(
                  "Kết quả tìm kiếm",
                  style: headingStyle,
                ),
                Text.rich(
                  TextSpan(
                    text: "$searchQuery",
                    style: TextStyle(
                      fontWeight: FontWeight.bold,
                      fontStyle: FontStyle.italic,
                    ),
                    children: [
```

```
                    TextSpan(
                      text: " ở ",
                      style: TextStyle(
                        fontWeight: FontWeight.normal,
                        fontStyle: FontStyle.normal,
                      ),
                    ),
                    TextSpan(
                      text: "$searchIn",
                      style: TextStyle(
                        fontWeight: FontWeight.w900,
                        fontStyle: FontStyle.italic,
                      ),
                    ),
                  ],
                ),
              ),
              SizedBox(height: getProportionateScreenHeight(30)),
              SizedBox(
                height: SizeConfig.screenHeight * 0.75,
                child: buildProductsGrid(),
              ),
              SizedBox(height: getProportionateScreenHeight(30)),
            ],
          ),
        ),
      ),
    ),
  );
}

Widget buildProductsGrid() {
  return Container(
    padding: EdgeInsets.symmetric(
      horizontal: 8,
      vertical: 16,
    ),
    decoration: BoxDecoration(
      color: kPrimaryColor.withOpacity(.2),
      borderRadius: BorderRadius.circular(15),
    ),
    child: Builder(
      builder: (context) {
        if (searchResultProductsId.length > 0) {
          return GridView.builder(
            shrinkWrap: true,
            physics: BouncingScrollPhysics(),
            gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
```

```
                crossAxisCount: 2,
                childAspectRatio: 0.7,
                crossAxisSpacing: 4,
                mainAxisSpacing: 4,
              ),
              itemCount: searchResultProductsId.length,
              itemBuilder: (context, index) {
                return ProductCard(
                  productId: searchResultProductsId[index],
                  press: () {
                    Navigator.push(
                      context,
                      MaterialPageRoute(
                        builder: (context) => ProductDetailsScreen(
                          productId: searchResultProductsId[index],
                        ),
                      ),
                    );
                  },
                );
              },
            );
          }
          return Center(
            child: NothingToShowContainer(
              iconPath: "assets/icons/search_no_found.svg",
              secondaryMessage: "Tìm thấy 0 sản phẩm",
              primaryMessage: "Hãy thử một từ khóa tìm kiếm khác",
            ),
          );
        },
      ),
    );
  }
}
```

6. CATEGORY SCREEN SOURCE CODE

```
import 'package:enum_to_string/enum_to_string.dart';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/services/steams/category_product_stream.dart';
import 'package:flutter_online_shop/src/product_details/product_details_screen.dart';
```

```
import 'package:flutter_online_shop/src/search/search_screen.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/nothing_show.dart';
import 'package:flutter_online_shop/widget/product_card.dart';
import 'package:flutter_online_shop/widget/round_icon_button.dart';
import 'package:flutter_online_shop/widget/search_text_field.dart';
import 'package:logger/logger.dart';

class CategoryProductsScreen extends StatelessWidget {
  final ProductType productType;

  const CategoryProductsScreen({
    Key key,
    @required this.productType,
  }) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      body: CategoryBody(
        productType: productType,
      ),
    );
  }
}

class CategoryBody extends StatefulWidget {
  final ProductType productType;

  CategoryBody({
    Key key,
    @required this.productType,
  }) : super(key: key);

  @override
  _CategoryBodyState createState() => _CategoryBodyState(
      categoryProductsStream: CategoryProductsStream(productType));
}

class _CategoryBodyState extends State<CategoryBody> {
  final CategoryProductsStream categoryProductsStream;

  _CategoryBodyState({@required this.categoryProductsStream});

  @override
  void initState() {
    super.initState();
    categoryProductsStream.init();
```

```
  }

  @override
  void dispose() {
    super.dispose();
    categoryProductsStream.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: RefreshIndicator(
        onRefresh: refreshPage,
        child: SingleChildScrollView(
          physics: AlwaysScrollableScrollPhysics(),
          child: Padding(
            padding: EdgeInsets.symmetric(
                horizontal: getProportionateScreenWidth(screenPadding)),
            child: SizedBox(
              width: double.infinity,
              child: Column(
                children: [
                  SizedBox(height: getProportionateScreenHeight(20)),
                  buildHeadBar(),
                  SizedBox(height: getProportionateScreenHeight(20)),
                  SizedBox(
                    height: SizeConfig.screenHeight * 0.13,
                    child: buildCategoryBanner(),
                  ),
                  SizedBox(height: getProportionateScreenHeight(20)),
                  SizedBox(
                    height: SizeConfig.screenHeight * 0.68,
                    child: StreamBuilder<List<String>>(
                      stream: categoryProductsStream.stream,
                      builder: (context, snapshot) {
                        if (snapshot.hasData) {
                          List<String> productsId = snapshot.data;
                          if (productsId.length == 0) {
                            return Center(
                              child: NothingToShowContainer(
                                primaryMessage:
                                    "${EnumToString.convertToString(widget.productType)}
hiện tại đang trống",
                              ),
                            );
                          }

                          return buildProductsGrid(productsId);
```

```
                    } else if (snapshot.connectionState ==
                        ConnectionState.waiting) {
                      return Center(
                        child: CircularProgressIndicator(),
                      );
                    } else if (snapshot.hasError) {
                      final error = snapshot.error;
                      Logger().w(error.toString());
                    }
                    return Center(
                      child: NothingToShowContainer(
                        iconPath: "assets/icons/network_error.svg",
                        primaryMessage: "Something went wrong",
                        secondaryMessage: "Unable to connect to Database",
                      ),
                    );
                  },
                ),
              ),
              SizedBox(height: getProportionateScreenHeight(20)),
            ],
          ),
        ),
      ),
    ),
  ),
  );
}

Widget buildHeadBar() {
  return Row(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: [
      RoundedIconButton(
        iconData: Icons.arrow_back_ios,
        press: () {
          Navigator.pop(context);
        },
      ),
      SizedBox(width: 5),
      Expanded(
        child: SearchField(
          onSubmit: (value) async {
            final query = value.toString();
            if (query.length <= 0) return;
            List<String> searchedProductsId;
            try {
              searchedProductsId = await ProductDatabaseHelper()
```

```
                .searchInProducts(query.toLowerCase(),
                    productType: widget.productType);
            if (searchedProductsId != null) {
              await Navigator.push(
                context,
                MaterialPageRoute(
                  builder: (context) => SearchResultScreen(
                    searchQuery: query,
                    searchResultProductsId: searchedProductsId,
                    searchIn:
                        EnumToString.convertToString(widget.productType),
                  ),
                ),
              );
              await refreshPage();
            } else {
              throw "Thao tác thất bại,vui lòng thử lại";
            }
          } catch (e) {
            final error = e.toString();
            Logger().e(error);
            ScaffoldMessenger.of(context).showSnackBar(
              SnackBar(
                content: Text("$error"),
              ),
            );
          }
        },
      ),
    ),
  ],
);
}

Future<void> refreshPage() {
  categoryProductsStream.reload();
  return Future<void>.value();
}

Widget buildCategoryBanner() {
  return Stack(
    children: [
      Container(
        decoration: BoxDecoration(
          image: DecorationImage(
            image: AssetImage(bannerFromProductType()),
            fit: BoxFit.fill,
            colorFilter: ColorFilter.mode(
```

```
              kPrimaryColor.withOpacity(.1),
              BlendMode.hue,
            ),
          ),
          borderRadius: BorderRadius.circular(30),
        ),
      ),
      Align(
        alignment: Alignment.centerLeft,
        child: Padding(
          padding: const EdgeInsets.only(left: 16),
          child: Text(
            EnumToString.convertToString(widget.productType),
            style: TextStyle(
              color: Colors.white,
              fontWeight: FontWeight.w900,
              fontSize: 24,
            ),
          ),
        ),
      ),
    ],
  );
}

Widget buildProductsGrid(List<String> productsId) {
  return Container(
    padding: EdgeInsets.symmetric(
      vertical: 16,
      horizontal: 8,
    ),
    decoration: BoxDecoration(
      color: kPrimaryColor.withOpacity(.3),
      borderRadius: BorderRadius.circular(15),
    ),
    child: GridView.builder(
      physics: BouncingScrollPhysics(),
      itemCount: productsId.length,
      shrinkWrap: true,
      itemBuilder: (context, index) {
        return ProductCard(
          productId: productsId[index],
          press: () {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => ProductDetailsScreen(
                  productId: productsId[index],
```

```
                ),
              ),
            ).then(
              (_) async {
                await refreshPage();
              },
            );
          },
        );
      },
      gridDelegate: SliverGridDelegateWithFixedCrossAxisCount(
        crossAxisCount: 2,
        childAspectRatio: 0.75,
        crossAxisSpacing: 2,
        mainAxisSpacing: 8,
      ),
      padding: EdgeInsets.symmetric(
        horizontal: 4,
        vertical: 12,
      ),
    ),
  );
}

String bannerFromProductType() {
  switch (widget.productType) {
    case ProductType.Electronics:
      return "assets/images/electronics_banner.jpg";
    case ProductType.Books:
      return "assets/images/books_banner.jpg";
    case ProductType.Fashion:
      return "assets/images/fashions_banner.jpg";
    case ProductType.Groceries:
      return "assets/images/groceries_banner.jpg";
    case ProductType.Furniture:
      return "assets/images/arts_banner.jpg";
    case ProductType.Others:
      return "assets/images/others_banner.jpg";
    default:
      return "assets/images/others_banner.jpg";
  }
}
}
```

7. CHANGE INFORMATION SCREEN SOURCE CODE

Change Information Screen chứa các widget con gồm :

1. Change_name.dart
2. Change_email.dart
3. Change_password.dart

7.1. CHANGE_NAME.DART

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/src/home/widget/title_container.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/change_infor_button.dart';
import 'package:future_progress_dialog/future_progress_dialog.dart';

class ChangeDisplayNameScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      appBar: AppBar(),
      body: Container(
        height: SizeConfig.screenHeight,
        child: Stack(
          children: [

            SingleChildScrollView(
              physics: BouncingScrollPhysics(),
              child: Padding(
                padding: EdgeInsets.symmetric(
                    horizontal: getProportionateScreenWidth(0)),
                child: Column(
                  children: [
                    SizedBox(height: SizeConfig.screenHeight * 0.05),
                    ChangeDisplayNameForm(),
                  ],
                ),
              ),
            ),
```

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

class ChangeDisplayNameForm extends StatefulWidget {
  const ChangeDisplayNameForm({
    Key key,
  }) : super(key: key);

  @override
  _ChangeDisplayNameFormState createState() => _ChangeDisplayNameFormState();
}

class _ChangeDisplayNameFormState extends State<ChangeDisplayNameForm> {
  final _formKey = GlobalKey<FormState>();

  final TextEditingController newDisplayNameController =
      TextEditingController();

  final TextEditingController currentDisplayNameController =
      TextEditingController();

  @override
  void dispose() {
    newDisplayNameController.dispose();
    currentDisplayNameController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    final form = Form(
      key: _formKey,
      child: Padding(
        padding: EdgeInsets.symmetric(
            horizontal: getProportionateScreenWidth(screenPadding * 2)),
        child: Column(
          children: [
            TitleContainer(
              title: 'Đổi tên',
            ),
            Stack(
              children: [
                Container(
                  padding: EdgeInsets.only(
```

```
                left: getProportionateScreenWidth(screenPadding * 2),
                top: getProportionateScreenHeight(screenPadding),
                bottom: getProportionateScreenHeight(screenPadding)),
            decoration: BoxDecoration(
              color: Colors.grey[200].withOpacity(.9),
              border: Border.all(color: kPrimaryColor, width: 2),
            ),
            child: Column(
              children: [
                SizedBox(height: SizeConfig.screenHeight * 0.01),
                buildCurrentDisplayNameField(),
                SizedBox(height: SizeConfig.screenHeight * 0.05),
                buildNewDisplayNameField(),
                SizedBox(height: SizeConfig.screenHeight * 0.01),
              ],
            ),
          ),
          Positioned(
            right: 0,
            child: Container(
              width: 10,
              height: SizeConfig.screenHeight * .34,
              color: kPrimaryColor,
            ),
          ),
        ],
      ),
      ChangeInfoButton(
        text: "Cập nhật Tên".toUpperCase(),
        icon: Icons.check,
        press: () {
          final uploadFuture = changeDisplayNameButtonCallback();
          showDialog(
            context: context,
            builder: (context) {
              return FutureProgressDialog(
                uploadFuture,
                message: Text("Hệ thống đang cập nhật tên người dùng"),
              );
            },
          );
          ScaffoldMessenger.of(context).showSnackBar(SnackBar(
              content: Text("Cập nhật tên người dùng thành công")));
        },
      ),
    ],
  ),
),
```

```
  );

  return form;
}

Widget buildNewDisplayNameField() {
  return TextFormField(
    controller: newDisplayNameController,
    keyboardType: TextInputType.name,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập tên bạn muốn thay đổi",
      labelText: "Tên mới",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (newDisplayNameController.text.isEmpty) {
        return "Vui lòng nhập tên để thực hiện cập nhật";
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildCurrentDisplayNameField() {
  return StreamBuilder<User>(
    stream: AuthentificationService().userChanges,
    builder: (context, snapshot) {
      String displayName;
      if (snapshot.hasData && snapshot.data != null)
        displayName = snapshot.data.displayName;
      final textField = TextFormField(
        controller: currentDisplayNameController,
        decoration: InputDecoration(
          enabledBorder: outlineTextField,
          focusedBorder: outlineTextField,
          labelText: "Tên hiện tại",
          hintText: 'Không có tên nào tồn tại',
          labelStyle: TextStyle(color: Colors.black87),
          floatingLabelBehavior: FloatingLabelBehavior.always,
        ),
        readOnly: true,
      );
      if (displayName != null)
        currentDisplayNameController.text = displayName;
      return textField;
```

```
      },
    );
  }

  Future<void> changeDisplayNameButtonCallback() async {
    if (_formKey.currentState.validate()) {
      _formKey.currentState.save();
      await AuthentificationService()
          .updateCurrentUserDisplayName(newDisplayNameController.text);
      print("Display Name updated to ${newDisplayNameController.text} ...");
    }
  }
}
```

7.2. CHANGE_EMAIL.DART

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/exceptions/auth/actions_exception.dart';
import 'package:flutter_online_shop/exceptions/auth/firebase_exception.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/src/home/widget/title_container.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/change_infor_button.dart';
import 'package:flutter_online_shop/widget/custom_stuffix_icon.dart';
import 'package:flutter_online_shop/widget/default_button.dart';
import 'package:future_progress_dialog/future_progress_dialog.dart';
import 'package:logger/logger.dart';

class ChangeEmailScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      appBar: AppBar(),
      body: SafeArea(
        child: Container(
          height: SizeConfig.screenHeight,
          child: Stack(
            children: [

              SingleChildScrollView(
                physics: BouncingScrollPhysics(),
                child: Padding(
                  padding: EdgeInsets.symmetric(
```

```
                    horizontal: getProportionateScreenWidth(0)),
                child: SizedBox(
                  width: double.infinity,
                  child: Column(
                    children: [
                      SizedBox(height: SizeConfig.screenHeight * 0.05),
                      ChangeEmailForm(),
                    ],
                  ),
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}
}

class ChangeEmailForm extends StatefulWidget {
  @override
  _ChangeEmailFormState createState() => _ChangeEmailFormState();
}

class _ChangeEmailFormState extends State<ChangeEmailForm> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController currentEmailController = TextEditingController();
  final TextEditingController newEmailController = TextEditingController();
  final TextEditingController passwordController = TextEditingController();

  @override
  void dispose() {
    currentEmailController.dispose();
    newEmailController.dispose();
    passwordController.dispose();
    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    final form = Form(
      key: _formKey,
      child: Padding(
        padding: EdgeInsets.symmetric(
            horizontal: getProportionateScreenWidth(screenPadding * 2)),
        child: Column(
          children: [
```

```
TitleContainer(
  title: "Thay đổi Địa chỉ Email",
),
Stack(
  children: [
    Container(
      padding: EdgeInsets.only(
          left: getProportionateScreenWidth(screenPadding * 2),
          top: getProportionateScreenHeight(screenPadding),
          bottom: getProportionateScreenHeight(screenPadding)),
      decoration: BoxDecoration(
        color: Colors.grey[200].withOpacity(.9),
        border: Border.all(color: kPrimaryColor, width: 2),
      ),
      child: Column(
        children: [
          buildCurrentEmailFormField(),
          SizedBox(height: getProportionateScreenHeight(20)),
          buildNewEmailFormField(),
          SizedBox(height: getProportionateScreenHeight(20)),
          buildPasswordFormField(),
        ],
      ),
    ),
    Positioned(
      right: 0,
      child: Container(
        width: 10,
        height: SizeConfig.screenHeight * .34,
        color: kPrimaryColor,
      ),
    ),
  ],
),
ChangeInfoButton(
  text: "Cập nhật Email".toUpperCase(),
  icon: Icons.check,
  press: () {
    final updateFuture = changeEmailButtonCallback();
    showDialog(
      context: context,
      builder: (context) {
        return FutureProgressDialog(
          updateFuture,
          message: Text("Đang cập nhật Email"),
        );
      },
    );
```

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

  return form;
}

Widget buildPasswordFormField() {
  return TextFormField(
    controller: passwordController,
    obscureText: true,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập Mật khẩu của bạn",
      labelText: "Password",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (passwordController.text.isEmpty) {
        return "Mật khẩu không được bỏ trống";
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildCurrentEmailFormField() {
  return StreamBuilder<User>(
    stream: AuthentificationService().userChanges,
    builder: (context, snapshot) {
      String currentEmail;
      if (snapshot.hasData && snapshot.data != null)
        currentEmail = snapshot.data.email;
      final textField = TextFormField(
        controller: currentEmailController,
        decoration: InputDecoration(
          enabledBorder: outlineTextField,
          focusedBorder: outlineTextField,
          hintText: "Email hiện tại",
          labelText: "Email hiện tại",
          floatingLabelBehavior: FloatingLabelBehavior.always,
        ),
        readOnly: true,
```

```
      );
      if (currentEmail != null) currentEmailController.text = currentEmail;
      return textField;
    },
  );
}

Widget buildNewEmailFormField() {
  return TextFormField(
    controller: newEmailController,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập Email muốn thay đổi",
      labelText: "Email mới",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (newEmailController.text.isEmpty) {
        return kEmailNullError;
      } else if (!emailValidatorRegExp.hasMatch(newEmailController.text)) {
        return kInvalidEmailError;
      } else if (newEmailController.text == currentEmailController.text) {
        return "Email đã được liên kết";
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Future<void> changeEmailButtonCallback() async {
  if (_formKey.currentState.validate()) {
    _formKey.currentState.save();
    final AuthentificationService authService = AuthentificationService();
    bool passwordValidation =
        await authService.verifyCurrentUserPassword(passwordController.text);
    if (passwordValidation) {
      bool updationStatus = false;
      String snackbarMessage;
      try {
        updationStatus = await authService.changeEmailForCurrentUser(
            newEmail: newEmailController.text);
        if (updationStatus == true) {
          snackbarMessage =
              "Gửi email xác minh. Vui lòng xác minh email mới của bạn";
        } else {
          throw FirebaseCredentialActionAuthUnknownReasonFailureException(
```

```
                message: "Thao tác thất bại,vui lòng thử lại");
            }
          } on MessagedFirebaseAuthException catch (e) {
            snackbarMessage = e.message;
          } catch (e) {
            snackbarMessage = e.toString();
          } finally {
            Logger().i(snackbarMessage);
            ScaffoldMessenger.of(context).showSnackBar(
              SnackBar(
                content: Text(snackbarMessage),
              ),
            );
          }
        }
      }
    }
  }
```

7.3. CHANGE_PASSWORD.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/exceptions/auth/actions_exception.dart';
import 'package:flutter_online_shop/exceptions/auth/firebase_exception.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/src/home/widget/title_container.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/change_infor_button.dart';
import 'package:future_progress_dialog/future_progress_dialog.dart';
import 'package:logger/logger.dart';

class ChangePasswordScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      resizeToAvoidBottomInset: false,
      appBar: AppBar(),
      body: SafeArea(
        child: Container(
          height: SizeConfig.screenHeight,
          child: Stack(
            children: [
              SingleChildScrollView(
                physics: BouncingScrollPhysics(),
                child: SizedBox(
                  width: double.infinity,
```

```
                child: Column(
                  children: [
                    SizedBox(height: SizeConfig.screenHeight * 0.05),
                    ChangePasswordForm(),
                  ],
                ),
              ),
            ),
          ],
        ),
      ),
    ),
  );
}
}

class ChangePasswordForm extends StatefulWidget {
  @override
  _ChangePasswordFormState createState() => _ChangePasswordFormState();
}

class _ChangePasswordFormState extends State<ChangePasswordForm> {
  final _formKey = GlobalKey<FormState>();
  final TextEditingController currentPasswordController =
      TextEditingController();
  final TextEditingController newPasswordController = TextEditingController();
  final TextEditingController confirmNewPasswordController =
      TextEditingController();
  bool isVisible = false;

  @override
  void dispose() {
    currentPasswordController.dispose();
    newPasswordController.dispose();
    confirmNewPasswordController.dispose();

    super.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Form(
      key: _formKey,
      child: Padding(
        padding: EdgeInsets.symmetric(
            horizontal: getProportionateScreenWidth(screenPadding * 2)),
        child: Column(
          children: [
```

```
TitleContainer(
  title: 'Thay Đổi mật khẩu',
),
Stack(
  children: [
    Container(
      padding: EdgeInsets.only(
          left: getProportionateScreenWidth(screenPadding * 2),
          top: getProportionateScreenHeight(screenPadding),
          bottom: getProportionateScreenHeight(screenPadding)),
      decoration: BoxDecoration(
        border: Border.all(color: kPrimaryColor, width: 2),
        color: Colors.grey[200].withOpacity(.9),
      ),
      child: Column(
        children: [
          buildCurrentPasswordFormField(),
          SizedBox(height: getProportionateScreenHeight(30)),
          buildNewPasswordFormField(),
          SizedBox(height: getProportionateScreenHeight(30)),
          buildConfirmNewPasswordFormField(),
        ],
      ),
    ),
    Positioned(
      right: 0,
      child: Container(
        width: 10,
        height: SizeConfig.screenHeight * .37,
        color: kPrimaryColor,
      ),
    ),
  ],
),
ChangeInfoButton(
  text: "Đổi mật khẩu".toUpperCase(),
  icon: Icons.check,
  press: () {
    final updateFuture = changePasswordButtonCallback();
    showDialog(
      context: context,
      builder: (context) {
        return FutureProgressDialog(
          updateFuture,
          message: Text("Đang cập nhật Mật khẩu"),
        );
      },
    );
```

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

Widget buildConfirmNewPasswordFormField() {
  return TextFormField(
    controller: confirmNewPasswordController,
    obscureText: true,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Xác nhận mật khẩu",
      labelText: "Xác nhận mật khẩu",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (confirmNewPasswordController.text != newPasswordController.text) {
        return "Mật khẩu không trùng khớp";
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildCurrentPasswordFormField() {
  return TextFormField(
    controller: currentPasswordController,
    obscureText: true,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập mật khẩu hiện tại",
      labelText: "Mật khẩu hiện tại",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      return null;
    },
    autovalidateMode: AutovalidateMode.disabled,
  );
}

Widget buildNewPasswordFormField() {
```

```
  return TextFormField(
    controller: newPasswordController,
    obscureText: true,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập mật khẩu mới",
      labelText: "Mật khẩu mới",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (newPasswordController.text.isEmpty) {
        return "Mật khẩu không thể để trống";
      } else if (newPasswordController.text.length < 8) {
        return "Mật khẩu quá ngắn";
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Future<void> changePasswordButtonCallback() async {
  if (_formKey.currentState.validate()) {
    _formKey.currentState.save();
    final AuthentificationService authService = AuthentificationService();
    bool currentPasswordValidation = await authService
        .verifyCurrentUserPassword(currentPasswordController.text);
    if (currentPasswordValidation == false) {
      print("Current password provided is wrong");
    } else {
      bool updationStatus = false;
      String snackbarMessage;
      try {
        updationStatus = await authService.changePasswordForCurrentUser(
            newPassword: newPasswordController.text);
        if (updationStatus == true) {
          snackbarMessage = "Cập nhật mật khẩu thành công";
        } else {
          throw FirebaseCredentialActionAuthUnknownReasonFailureException(
              message: "Thay đổi mật khẩu thất bại,vui lòng thử lại sau");
        }
      } on MessagedFirebaseAuthException catch (e) {
        snackbarMessage = e.message;
      } catch (e) {
        snackbarMessage = e.toString();
      } finally {
        Logger().i(snackbarMessage);
```

```
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
              content: Text(snackbarMessage),
            ),
          );
        }
      }
    }
  }
}
```

8. EDIT ADDRESS SCREEN.DART

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/address_model.dart';
import 'package:flutter_online_shop/services/db/user_db_helper.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/default_button.dart';
import 'package:logger/logger.dart';
import 'package:string_validator/string_validator.dart';

class EditAddressScreen extends StatelessWidget {
  final String addressIdToEdit;

  const EditAddressScreen({Key key, this.addressIdToEdit}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: SafeArea(
        child: SingleChildScrollView(
          physics: BouncingScrollPhysics(),
          child: Padding(
            padding: EdgeInsets.symmetric(
                horizontal: getProportionateScreenWidth(screenPadding)),
            child: SizedBox(
              width: double.infinity,
              child: Column(
                children: [
                  SizedBox(height: getProportionateScreenHeight(30)),
                  addressIdToEdit == null
                      ? AddressDetailsForm(
                          addressToEdit: null,
                        )
```

```
                    : FutureBuilder<Address>(
                        future: UserDatabaseHelper()
                            .getAddressFromId(addressIdToEdit),
                        builder: (context, snapshot) {
                          if (snapshot.hasData) {
                            final address = snapshot.data;
                            return AddressDetailsForm(addressToEdit: address);
                          } else if (snapshot.connectionState ==
                              ConnectionState.waiting) {
                            return Center(
                              child: CircularProgressIndicator(),
                            );
                          }
                          return AddressDetailsForm(
                            addressToEdit: null,
                          );
                        },
                      ),
                  SizedBox(height: getProportionateScreenHeight(40)),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}

class AddressDetailsForm extends StatefulWidget {
  final Address addressToEdit;
  AddressDetailsForm({
    Key key,
    this.addressToEdit,
  }) : super(key: key);

  @override
  _AddressDetailsFormState createState() => _AddressDetailsFormState();
}

class _AddressDetailsFormState extends State<AddressDetailsForm> {
  final _formKey = GlobalKey<FormState>();

  final TextEditingController titleFieldController = TextEditingController();

  final TextEditingController receiverFieldController = TextEditingController();

  final TextEditingController addressLine1FieldController =
```

```
    TextEditingController();

final TextEditingController addressLine2FieldController =
    TextEditingController();

final TextEditingController cityFieldController = TextEditingController();

final TextEditingController districtFieldController = TextEditingController();

final TextEditingController stateFieldController = TextEditingController();

final TextEditingController landmarkFieldController = TextEditingController();

final TextEditingController pincodeFieldController = TextEditingController();

final TextEditingController phoneFieldController = TextEditingController();

@override
void dispose() {
  titleFieldController.dispose();
  receiverFieldController.dispose();
  addressLine1FieldController.dispose();
  addressLine2FieldController.dispose();
  cityFieldController.dispose();
  stateFieldController.dispose();
  districtFieldController.dispose();
  landmarkFieldController.dispose();
  pincodeFieldController.dispose();
  phoneFieldController.dispose();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  final form = Form(
    key: _formKey,
    child: Stack(
      children: [
        Column(
          children: [
            SizedBox(height: getProportionateScreenHeight(20)),
            buildTitleField(),
            SizedBox(height: getProportionateScreenHeight(30)),
            buildReceiverField(),
            SizedBox(height: getProportionateScreenHeight(30)),
            buildAddressLine1Field(),
            SizedBox(height: getProportionateScreenHeight(30)),
            buildAddressLine2Field(),
```

```
            SizedBox(height: getProportionateScreenHeight(30)),
            buildCityField(),
            SizedBox(height: getProportionateScreenHeight(30)),
            buildDistrictField(),
            SizedBox(height: getProportionateScreenHeight(30)),
            buildStateField(),
            SizedBox(height: getProportionateScreenHeight(30)),
            buildLandmarkField(),
            SizedBox(height: getProportionateScreenHeight(30)),
            buildPincodeField(),
            SizedBox(height: getProportionateScreenHeight(30)),
            buildPhoneField(),
            SizedBox(height: getProportionateScreenHeight(30)),
            DefaultButton(
              text: "Lưu địa chỉ",
              press: widget.addressToEdit == null
                  ? saveNewAddressButtonCallback
                  : saveEditedAddressButtonCallback,
            ),
          ],
        ),
        Positioned(
          right: 0,
          top: getProportionateScreenHeight(20),
          child: Container(
            width: 10,
            height: SizeConfig.screenHeight * 1.145,
            color: kPrimaryColor,
          ),
        ),
      ],
    ),
  );
  if (widget.addressToEdit != null) {
    titleFieldController.text = widget.addressToEdit.title;
    receiverFieldController.text = widget.addressToEdit.receiver;
    addressLine1FieldController.text = widget.addressToEdit.addresLine1;
    addressLine2FieldController.text = widget.addressToEdit.addressLine2;
    cityFieldController.text = widget.addressToEdit.city;
    districtFieldController.text = widget.addressToEdit.district;
    stateFieldController.text = widget.addressToEdit.state;
    landmarkFieldController.text = widget.addressToEdit.landmark;
    pincodeFieldController.text = widget.addressToEdit.pincode;
    phoneFieldController.text = widget.addressToEdit.phone;
  }
  return form;
}
```

```
Widget buildTitleField() {
  return TextFormField(
    controller: titleFieldController,
    keyboardType: TextInputType.name,
    maxLength: 8,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập tiêu đề",
      labelText: "Tiêu đề",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (titleFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildReceiverField() {
  return TextFormField(
    controller: receiverFieldController,
    keyboardType: TextInputType.name,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập Họ & Tên người nhận",
      labelText: "Họ & Tên",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (receiverFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildAddressLine1Field() {
  return TextFormField(
    controller: addressLine1FieldController,
    keyboardType: TextInputType.streetAddress,
    decoration: InputDecoration(
```

```
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập địa chỉ thường trú",
      labelText: "Địa chỉ thường trú",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (addressLine1FieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildAddressLine2Field() {
  return TextFormField(
    controller: addressLine2FieldController,
    keyboardType: TextInputType.streetAddress,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập địa chỉ dự phòng",
      labelText: "Địa chỉ dự phòng",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (addressLine2FieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildCityField() {
  return TextFormField(
    controller: cityFieldController,
    keyboardType: TextInputType.name,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập Tỉnh\\Thành phố",
      labelText: "Tỉnh\\Thành Phố",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
```

```
    validator: (value) {
      if (cityFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildDistrictField() {
  return TextFormField(
    controller: districtFieldController,
    keyboardType: TextInputType.name,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập Quận\\Huyện",
      labelText: "Quận\\Huyện",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (districtFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildStateField() {
  return TextFormField(
    controller: stateFieldController,
    keyboardType: TextInputType.name,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập Phường\\Xã",
      labelText: "Phường\\Xã",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (stateFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
```

```
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildPincodeField() {
  return TextFormField(
    controller: pincodeFieldController,
    keyboardType: TextInputType.number,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập mã PIN",
      labelText: "PINCODE",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (pincodeFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      } else if (!isNumeric(pincodeFieldController.text)) {
        return "Yêu cầu chỉ nhập kí tự";
      } else if (pincodeFieldController.text.length != 6) {
        return "Mã PIN yêu cầu đủ 6 kí tự";
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildLandmarkField() {
  return TextFormField(
    controller: landmarkFieldController,
    keyboardType: TextInputType.name,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập số nhà",
      labelText: "Số nhà",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (landmarkFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
```

```
}

Widget buildPhoneField() {
  return TextFormField(
    controller: phoneFieldController,
    keyboardType: TextInputType.phone,
    decoration: InputDecoration(
      enabledBorder: outlineTextField,
      focusedBorder: outlineTextField,
      hintText: "Nhập số điện thoại",
      labelText: "Số điện thoại",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (value) {
      if (phoneFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      } else if (phoneFieldController.text.length != 10) {
        return "Tối đa 10 kí tự";
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Future<void> saveNewAddressButtonCallback() async {
  if (_formKey.currentState.validate()) {
    _formKey.currentState.save();
    final Address newAddress = generateAddressObject();
    bool status = false;
    String snackbarMessage;
    try {
      status =
          await UserDatabaseHelper().addAddressForCurrentUser(newAddress);
      if (status == true) {
        snackbarMessage = "Lưu địa chỉ thành công";
      } else {
        throw "Thao tác thất bại,vui lòng thử lại";
      }
    } on FirebaseException catch (e) {
      Logger().w("Firebase Exception: $e");
      snackbarMessage = "Thao tác thất bại,vui lòng thử lại";
    } catch (e) {
      Logger().w("Unknown Exception: $e");
      snackbarMessage = "Thao tác thất bại,vui lòng thử lại";
    } finally {
      Logger().i(snackbarMessage);
      ScaffoldMessenger.of(context).showSnackBar(
```

```
        SnackBar(
          content: Text(snackbarMessage),
        ),
      );
    }
  }
}

Future<void> saveEditedAddressButtonCallback() async {
  if (_formKey.currentState.validate()) {
    _formKey.currentState.save();
    final Address newAddress =
        generateAddressObject(id: widget.addressToEdit.id);

    bool status = false;
    String snackbarMessage;
    try {
      status =
          await UserDatabaseHelper().updateAddressForCurrentUser(newAddress);
      if (status == true) {
        snackbarMessage = "Lưu địa chỉ thành công";
      } else {
        throw "Thao tác thất bại,vui lòng thử lại";
      }
    } on FirebaseException catch (e) {
      Logger().w("Firebase Exception: $e");
      snackbarMessage = "Thao tác thất bại,vui lòng thử lại";
    } catch (e) {
      Logger().w("Unknown Exception: $e");
      snackbarMessage = "Thao tác thất bại,vui lòng thử lại";
    } finally {
      Logger().i(snackbarMessage);
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text(snackbarMessage),
        ),
      );
    }
  }
}

Address generateAddressObject({String id}) {
  return Address(
    id: id,
    title: titleFieldController.text,
    receiver: receiverFieldController.text,
    addresLine1: addressLine1FieldController.text,
    addressLine2: addressLine2FieldController.text,
```

```
      city: cityFieldController.text,
      district: districtFieldController.text,
      state: stateFieldController.text,
      landmark: landmarkFieldController.text,
      pincode: pincodeFieldController.text,
      phone: phoneFieldController.text,
    );
  }
}
```

9. EDIT PRODUCT SCREEN.DART

Edit Product Screen chứa các widget con gồm :

1. Edit_product_provider.dart

```
import 'dart:io';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:enum_to_string/enum_to_string.dart';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/exceptions/local_file/image_picking_exceptions.dart';
import 'package:flutter_online_shop/exceptions/local_file/local_files_exceptions.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/services/file_access/fs_files_access.dart';
import 'package:flutter_online_shop/services/local_files_access/local_files_access.dart';
import 'package:flutter_online_shop/src/edit_product/widget/edit_product_provider.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/default_button.dart';
import 'package:flutter_tags/flutter_tags.dart';
import 'package:future_progress_dialog/future_progress_dialog.dart';
import 'package:logger/logger.dart';
import 'package:provider/provider.dart';

class EditProductScreen extends StatelessWidget {
  final Product productToEdit;

  const EditProductScreen({Key key, this.productToEdit}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (context) => ProductDetails(),
```

```
      child: Scaffold(
        appBar: AppBar(),
        body: SafeArea(
          child: SingleChildScrollView(
            physics: BouncingScrollPhysics(),
            child: Padding(
              padding: EdgeInsets.symmetric(
                  horizontal: getProportionateScreenWidth(screenPadding)),
              child: SizedBox(
                width: double.infinity,
                child: Column(
                  children: [
                    SizedBox(height: getProportionateScreenHeight(10)),
                    SizedBox(height: getProportionateScreenHeight(30)),
                    EditProductForm(product: productToEdit),
                    SizedBox(height: getProportionateScreenHeight(30)),
                  ],
                ),
              ),
            ),
          ),
        ),
      ),
    );
  }
}

class EditProductForm extends StatefulWidget {
  final Product product;
  EditProductForm({
    Key key,
    this.product,
  }) : super(key: key);

  @override
  _EditProductFormState createState() => _EditProductFormState();
}

class _EditProductFormState extends State<EditProductForm> {
  final _basicDetailsFormKey = GlobalKey<FormState>();
  final _describeProductFormKey = GlobalKey<FormState>();
  final _tagStateKey = GlobalKey<TagsState>();

  final TextEditingController titleFieldController = TextEditingController();
  final TextEditingController variantFieldController = TextEditingController();
  final TextEditingController discountPriceFieldController =
      TextEditingController();
  final TextEditingController originalPriceFieldController =
```

```
    TextEditingController();
final TextEditingController highlightsFieldController =
    TextEditingController();
final TextEditingController desciptionFieldController =
    TextEditingController();
final TextEditingController sellerFieldController = TextEditingController();

bool newProduct = true;
Product product;
String s = "";
ProductDetails _productDetails;
@override
void dispose() {
  titleFieldController.dispose();
  variantFieldController.dispose();
  discountPriceFieldController.dispose();
  originalPriceFieldController.dispose();
  highlightsFieldController.dispose();
  desciptionFieldController.dispose();
  sellerFieldController.dispose();

  super.dispose();
}

@override
void initState() {
  super.initState();
  if (widget.product == null) {
    product = Product(null);
    newProduct = true;
  } else {
    product = widget.product;
    newProduct = false;
    final productDetails =
        Provider.of<ProductDetails>(context, listen: false);
    productDetails.initialSelectedImages = widget.product.images
        .map((e) => CustomImage(imgType: ImageType.network, path: e))
        .toList();
    productDetails.initialProductType = product.productType;
    productDetails.initSearchTags = product.searchTags ?? [];
  }
}

@override
Widget build(BuildContext context) {
  final column = Column(
    children: [
      buildBasicDetailsTile(context),
```

```
      SizedBox(height: getProportionateScreenHeight(10)),
      buildDescribeProductTile(context),
      SizedBox(height: getProportionateScreenHeight(10)),
      buildUploadImagesTile(context),
      SizedBox(height: getProportionateScreenHeight(20)),
      buildProductTypeDropdown(),
      SizedBox(height: getProportionateScreenHeight(20)),
      buildProductSearchTagsTile(),
      SizedBox(height: getProportionateScreenHeight(80)),
      DefaultButton(
          text: "Lưu sản phẩm",
          press: () {
            saveProductButtonCallback(context);
          }),
      SizedBox(height: getProportionateScreenHeight(10)),
    ],
  );
  if (newProduct == false) {
    titleFieldController.text = product.title;
    variantFieldController.text = product.variant;
    discountPriceFieldController.text = product.discountPrice.toString();
    originalPriceFieldController.text = product.originalPrice.toString();
    highlightsFieldController.text = product.highlights;
    desciptionFieldController.text = product.description;
    sellerFieldController.text = product.seller;
  }
  return column;
}

Widget buildProductSearchTags() {
  return Consumer<ProductDetails>(
    builder: (context, productDetails, child) {
      return Tags(
        key: _tagStateKey,
        horizontalScroll: true,
        heightHorizontalScroll: getProportionateScreenHeight(80),
        textField: TagsTextField(
          lowerCase: true,
          width: getProportionateScreenWidth(150),
          constraintSuggestion: true,
          hintText: "Thêm thẻ tìm kiếm",
          keyboardType: TextInputType.name,
          onSubmitted: (String str) {
            setState(() {
              s = str;
              _productDetails = productDetails;
            });
            productDetails.addSearchTag(str.toLowerCase());
```

```
          },
        ),
        itemCount: productDetails.searchTags.length,
        itemBuilder: (index) {
          final item = productDetails.searchTags[index];
          return ItemTags(
            index: index,
            title: item,
            active: true,
            activeColor: kPrimaryColor,
            padding: EdgeInsets.symmetric(
              horizontal: 12,
              vertical: 8,
            ),
            alignment: MainAxisAlignment.spaceBetween,
            removeButton: ItemTagsRemoveButton(
              backgroundColor: Colors.white,
              color: kTextColor,
              onRemoved: () {
                productDetails.removeSearchTag(index: index);
                return true;
              },
            ),
          );
        },
      );
    },
  );
}

Widget buildBasicDetailsTile(BuildContext context) {
  return Form(
    key: _basicDetailsFormKey,
    child: ExpansionTile(
      maintainState: true,
      title: Text(
        "Chi tiết cơ bản",
        style: Theme.of(context).textTheme.headline6,
      ),
      leading: Icon(
        Icons.shop,
        color: kPrimaryColor,
      ),
      childrenPadding:
          EdgeInsets.symmetric(vertical: getProportionateScreenHeight(20)),
      children: [
        buildTitleField(),
        SizedBox(height: getProportionateScreenHeight(20)),
```

```
        buildVariantField(),
        SizedBox(height: getProportionateScreenHeight(20)),
        buildOriginalPriceField(),
        SizedBox(height: getProportionateScreenHeight(20)),
        buildDiscountPriceField(),
        SizedBox(height: getProportionateScreenHeight(20)),
        buildSellerField(),
        SizedBox(height: getProportionateScreenHeight(20)),
      ],
    ),
  );
}

bool validateBasicDetailsForm() {
  if (_basicDetailsFormKey.currentState.validate()) {
    _basicDetailsFormKey.currentState.save();
    product.title = titleFieldController.text;
    product.variant = variantFieldController.text;
    product.originalPrice = double.parse(originalPriceFieldController.text);
    product.discountPrice = double.parse(discountPriceFieldController.text);
    product.seller = sellerFieldController.text;
    return true;
  }
  return false;
}

Widget buildDescribeProductTile(BuildContext context) {
  return Form(
    key: _describeProductFormKey,
    child: ExpansionTile(
      maintainState: true,
      title: Text(
        "Mô tả sản phẩm",
        style: Theme.of(context).textTheme.headline6,
      ),
      leading: Icon(
        Icons.description,
        color: kPrimaryColor,
      ),
      childrenPadding:
          EdgeInsets.symmetric(vertical: getProportionateScreenHeight(20)),
      children: [
        buildHighlightsField(),
        SizedBox(height: getProportionateScreenHeight(20)),
        buildDescriptionField(),
        SizedBox(height: getProportionateScreenHeight(20)),
      ],
    ),
```

```
  );
}

bool validateDescribeProductForm() {
  if (_describeProductFormKey.currentState.validate()) {
    _describeProductFormKey.currentState.save();
    product.highlights = highlightsFieldController.text;
    product.description = desciptionFieldController.text;
    return true;
  }
  return false;
}

Widget buildProductTypeDropdown() {
  return Container(
    padding: EdgeInsets.symmetric(
      horizontal: 24,
      vertical: 6,
    ),
    decoration: BoxDecoration(
      border: Border.all(color: kTextColor, width: 1),
      borderRadius: BorderRadius.all(Radius.circular(28)),
    ),
    child: Consumer<ProductDetails>(
      builder: (context, productDetails, child) {
        return DropdownButton(
          value: productDetails.productType,
          items: ProductType.values
              .map(
                (e) => DropdownMenuItem(
                  value: e,
                  child: Text(
                    EnumToString.convertToString(e),
                  ),
                ),
              )
              .toList(),
          hint: Text(
            "Chọn loại sản phẩm",
          ),
          style: TextStyle(
            color: kTextColor,
            fontSize: 16,
          ),
          onChanged: (value) {
            productDetails.productType = value;
          },
          elevation: 0,
```

```
          underline: SizedBox(width: 0, height: 0),
        );
      },
    ),
  );
}

Widget buildProductSearchTagsTile() {
  return ExpansionTile(
    title: Text(
      "Thẻ tìm kiếm",
      style: Theme.of(context).textTheme.headline6,
    ),
    leading: Icon(
      Icons.check_circle_sharp,
      color: kPrimaryColor,
    ),
    childrenPadding:
        EdgeInsets.symmetric(vertical: getProportionateScreenHeight(20)),
    children: [
      Text("Sản phẩm của bạn sẽ được tìm kiếm khi nhập Thẻ này"),
      SizedBox(height: getProportionateScreenHeight(15)),
      buildProductSearchTags(),
      Container(
        decoration: BoxDecoration(
          shape: BoxShape.circle,
          color: kPrimaryColor,
        ),
        child: IconButton(
          icon: Icon(Icons.check),
          onPressed: () {
            _productDetails.addSearchTag(s.toLowerCase());
          },
          color: Colors.white,
        ),
      )
    ],
  );
}

Widget buildUploadImagesTile(BuildContext context) {
  return ExpansionTile(
    title: Text(
      "Đăng hình ảnh",
      style: Theme.of(context).textTheme.headline6,
    ),
    leading: Icon(
      Icons.image,
```

```
  color: kPrimaryColor,
),
childrenPadding:
    EdgeInsets.symmetric(vertical: getProportionateScreenHeight(20)),
children: [
  Padding(
    padding: const EdgeInsets.all(16.0),
    child: IconButton(
        icon: Icon(
          Icons.add_a_photo,
        ),
        color: kTextColor,
        onPressed: () {
          addImageButtonCallback();
        }),
  ),
  Consumer<ProductDetails>(
    builder: (context, productDetails, child) {
      return Row(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          ...List.generate(
            productDetails.selectedImages.length,
            (index) => SizedBox(
              width: 80,
              height: 80,
              child: Padding(
                padding: const EdgeInsets.all(5.0),
                child: GestureDetector(
                  onTap: () {
                    addImageButtonCallback(index: index);
                  },
                  child: productDetails.selectedImages[index].imgType ==
                          ImageType.local
                      ? Image.memory(
                          File(productDetails.selectedImages[index].path)
                              .readAsBytesSync())
                      : Image.network(
                          productDetails.selectedImages[index].path),
                ),
              ),
            ),
          ),
        ],
      );
    },
  ),
],
```

```
  );
}

Widget buildTitleField() {
  return TextFormField(
    controller: titleFieldController,
    keyboardType: TextInputType.name,
    decoration: InputDecoration(
      hintText: "e.g., iPhone 12 Pro Max",
      labelText: "Tiêu đề sản phẩm",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (_) {
      if (titleFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildVariantField() {
  return TextFormField(
    controller: variantFieldController,
    keyboardType: TextInputType.name,
    decoration: InputDecoration(
      hintText: "e.g., Xanh da trời",
      labelText: "Màu",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (_) {
      if (variantFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildHighlightsField() {
  return TextFormField(
    controller: highlightsFieldController,
    keyboardType: TextInputType.multiline,
    decoration: InputDecoration(
      hintText:
          "e.g., RAM: 4GB | Front Camera: 30MP | Rear Camera: Quad Camera Setup",
```

```
        labelText: "Điểm nổi bật",
        floatingLabelBehavior: FloatingLabelBehavior.always,
      ),
      validator: (_) {
        if (highlightsFieldController.text.isEmpty) {
          return FIELD_REQUIRED_MSG;
        }
        return null;
      },
      autovalidateMode: AutovalidateMode.onUserInteraction,
      maxLines: null,
    );
  }

  Widget buildDescriptionField() {
    return TextFormField(
      controller: desciptionFieldController,
      keyboardType: TextInputType.multiline,
      minLines: 5,
      decoration: InputDecoration(
        hintText:
            "e.g., iPhone 12 Pro Max sở hữu diện \nmạo mới với khung viền được vát\nthẳng
và mạnh mẽ như trên iPad Pro\n2020,chấm dứt hơn 6 năm với kiểu \nthiết kế bo cong trên
iPhone 6 được ra mắt lần đầu tiên vào năm 2014.",
        labelText: "Mô tả sản phẩm",
        floatingLabelBehavior: FloatingLabelBehavior.always,
      ),
      validator: (_) {
        if (desciptionFieldController.text.isEmpty) {
          return FIELD_REQUIRED_MSG;
        }
        return null;
      },
      autovalidateMode: AutovalidateMode.onUserInteraction,
      maxLines: null,
    );
  }

  Widget buildSellerField() {
    return TextFormField(
      controller: sellerFieldController,
      keyboardType: TextInputType.name,
      decoration: InputDecoration(
        hintText: "e.g., Zara",
        labelText: "Đơn vị bán hàng",
        floatingLabelBehavior: FloatingLabelBehavior.always,
      ),
      validator: (_) {
```

```
      if (sellerFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildOriginalPriceField() {
  return TextFormField(
    controller: originalPriceFieldController,
    keyboardType: TextInputType.number,
    decoration: InputDecoration(
      hintText: "e.g., 12.000.000.0đ",
      labelText: "Giá gốc (Việt Nam)",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (_) {
      if (originalPriceFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Widget buildDiscountPriceField() {
  return TextFormField(
    controller: discountPriceFieldController,
    keyboardType: TextInputType.number,
    decoration: InputDecoration(
      hintText: "e.g., 11.000.000.0đ",
      labelText: "Giá bán(Đã giảm giá)",
      floatingLabelBehavior: FloatingLabelBehavior.always,
    ),
    validator: (_) {
      if (discountPriceFieldController.text.isEmpty) {
        return FIELD_REQUIRED_MSG;
      }
      return null;
    },
    autovalidateMode: AutovalidateMode.onUserInteraction,
  );
}

Future<void> saveProductButtonCallback(BuildContext context) async {
```

```
if (validateBasicDetailsForm() == false) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text("Lỗi trong Biểu mẫu Chi tiết Cơ bản"),
    ),
  );
  return;
}
if (validateDescribeProductForm() == false) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text("Lỗi trong Mô tả Sản phẩm"),
    ),
  );
  return;
}
final productDetails = Provider.of<ProductDetails>(context, listen: false);
if (productDetails.selectedImages.length < 1) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text("Ít nhất một hình ảnh về sản phẩm được tải lên"),
    ),
  );
  return;
}
if (productDetails.productType == null) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text("Vui lòng chọn Loại sản phẩm"),
    ),
  );
  return;
}
if (productDetails.searchTags.length < 3) {
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text("Cần ít nhất 3 Thẻ tìm kiếm"),
    ),
  );
  return;
}
String productId;
String snackbarMessage;
try {
  product.productType = productDetails.productType;
  product.searchTags = productDetails.searchTags;
  final productUploadFuture = newProduct
      ? ProductDatabaseHelper().addUsersProduct(product)
```

```
      : ProductDatabaseHelper().updateUsersProduct(product);
  productUploadFuture.then((value) {
    productId = value;
  });
  await showDialog(
    context: context,
    builder: (context) {
      return FutureProgressDialog(
        productUploadFuture,
        message: Text(newProduct
            ? "Đang tiến hành đăng tải sản phẩm"
            : "Đang tiến hành cập nhật sản phẩm"),
      );
    },
  );
  if (productId != null) {
    snackbarMessage = "Cập nhật thông tin sản phẩm thành công";
  } else {
    throw "Thao tác thất bại,vui lòng thử lại";
  }
} on FirebaseException catch (e) {
  Logger().w("Firebase Exception: $e");
  snackbarMessage = "Something went wrong";
} catch (e) {
  Logger().w("Unknown Exception: $e");
  snackbarMessage = e.toString();
} finally {
  Logger().i(snackbarMessage);
  ScaffoldMessenger.of(context).showSnackBar(
    SnackBar(
      content: Text(snackbarMessage),
    ),
  );
}
if (productId == null) return;
bool allImagesUploaded = false;
try {
  allImagesUploaded = await uploadProductImages(productId);
  if (allImagesUploaded == true) {
    snackbarMessage = "Đăng tất cả hình ảnh thành công";
  } else {
    throw "Không thể tải lên một số hình ảnh, vui lòng thử lại";
  }
} on FirebaseException catch (e) {
  Logger().w("Firebase Exception: $e");
  snackbarMessage = "Something went wrong";
} catch (e) {
  Logger().w("Unknown Exception: $e");
```

```
    snackbarMessage = "Something went wrong";
  } finally {
    Logger().i(snackbarMessage);
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text(snackbarMessage),
      ),
    );
  }
  List<String> downloadUrls = productDetails.selectedImages
      .map((e) => e.imgType == ImageType.network ? e.path : null)
      .toList();
  bool productFinalizeUpdate = false;
  try {
    final updateProductFuture =
        ProductDatabaseHelper().updateProductsImages(productId, downloadUrls);
    productFinalizeUpdate = await showDialog(
      context: context,
      builder: (context) {
        return FutureProgressDialog(
          updateProductFuture,
          message: Text("Đang tiến hành lưu sản phẩm"),
        );
      },
    );
    if (productFinalizeUpdate == true) {
      snackbarMessage = "Đăng tải sản phẩm thành công";
    } else {
      throw "Thao tác thất bại,vui lòng thử lại";
    }
  } on FirebaseException catch (e) {
    Logger().w("Firebase Exception: $e");
    snackbarMessage = "Something went wrong";
  } catch (e) {
    Logger().w("Unknown Exception: $e");
    snackbarMessage = e.toString();
  } finally {
    Logger().i(snackbarMessage);
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: Text(snackbarMessage),
      ),
    );
  }
  Navigator.pop(context);
}

Future<bool> uploadProductImages(String productId) async {
```

```
  bool allImagesUpdated = true;
  final productDetails = Provider.of<ProductDetails>(context, listen: false);
  for (int i = 0; i < productDetails.selectedImages.length; i++) {
    if (productDetails.selectedImages[i].imgType == ImageType.local) {
      print(
          "Hình ảnh được tải lên: " + productDetails.selectedImages[i].path);
      String downloadUrl;
      try {
        final imgUploadFuture = FirestoreFilesAccess().uploadFileToPath(
            File(productDetails.selectedImages[i].path),
            ProductDatabaseHelper().getPathForProductImage(productId, i));
        downloadUrl = await showDialog(
          context: context,
          builder: (context) {
            return FutureProgressDialog(
              imgUploadFuture,
              message: Text(
                  "Đang tải hình ảnh ${i + 1}/${productDetails.selectedImages.length}"),
            );
          },
        );
      } on FirebaseException catch (e) {
        Logger().w("Firebase Exception: $e");
      } catch (e) {
        Logger().w("Firebase Exception: $e");
      } finally {
        if (downloadUrl != null) {
          productDetails.selectedImages[i] =
              CustomImage(imgType: ImageType.network, path: downloadUrl);
        } else {
          allImagesUpdated = false;
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
              content: Text(
                  "Không thể tải lên hình ảnh ${i + 1} do một số vấn đề"),
            ),
          );
        }
      }
    }
  }
  return allImagesUpdated;
}

Future<void> addImageButtonCallback({int index}) async {
  final productDetails = Provider.of<ProductDetails>(context, listen: false);
  if (index == null && productDetails.selectedImages.length >= 3) {
    ScaffoldMessenger.of(context).showSnackBar(
```

```
          SnackBar(content: Text("Có thể tải lên tối đa 3 hình ảnh")));
      return;
    }
    String path;
    String snackbarMessage;
    try {
      path = await choseImageFromLocalFiles(context);
      if (path == null) {
        throw LocalImagePickingUnknownReasonFailureException();
      }
    } on LocalFileHandlingException catch (e) {
      Logger().i("Local File Handling Exception: $e");
      snackbarMessage = e.toString();
    } catch (e) {
      Logger().i("Unknown Exception: $e");
      snackbarMessage = e.toString();
    } finally {
      if (snackbarMessage != null) {
        Logger().i(snackbarMessage);
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text(snackbarMessage),
          ),
        );
      }
    }
    if (path == null) {
      return;
    }
    if (index == null) {
      productDetails.addNewSelectedImage(
          CustomImage(imgType: ImageType.local, path: path));
    } else {
      productDetails.setSelectedImageAtIndex(
          CustomImage(imgType: ImageType.local, path: path), index);
    }
  }
}
```

9.1. EDIT PRODUCT PROVIDER.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';

enum ImageType {
  local,
  network,
```

```
}

class CustomImage {
  final ImageType imgType;
  final String path;
  CustomImage({this.imgType = ImageType.local, @required this.path});
  @override
  String toString() {
    return "Instance of Custom Image: {imgType: $imgType, path: $path}";
  }
}

class ProductDetails extends ChangeNotifier {
  List<CustomImage> _selectedImages = List<CustomImage>();
  ProductType _productType;
  List<String> _searchTags = List<String>();

  List<CustomImage> get selectedImages {
    return _selectedImages;
  }

  set initialSelectedImages(List<CustomImage> images) {
    _selectedImages = images;
  }

  set selectedImages(List<CustomImage> images) {
    _selectedImages = images;
    notifyListeners();
  }

  void setSelectedImageAtIndex(CustomImage image, int index) {
    if (index < _selectedImages.length) {
      _selectedImages[index] = image;
      notifyListeners();
    }
  }

  void addNewSelectedImage(CustomImage image) {
    _selectedImages.add(image);
    notifyListeners();
  }

  ProductType get productType {
    return _productType;
  }

  set initialProductType(ProductType type) {
    _productType = type;
```

```
  }

  set productType(ProductType type) {
    _productType = type;
    notifyListeners();
  }

  List<String> get searchTags {
    return _searchTags;
  }

  set searchTags(List<String> tags) {
    _searchTags = tags;
    notifyListeners();
  }

  set initSearchTags(List<String> tags) {
    _searchTags = tags;
  }

  void addSearchTag(String tag) {
    _searchTags.add(tag);
    notifyListeners();
  }

  void removeSearchTag({int index}) {
    if (index == null)
      _searchTags.removeLast();
    else
      _searchTags.removeAt(index);
    notifyListeners();
  }
}
```

10.MANAGER ADDRESS SCREEN.DART

Manager Address Screen chứa các widget con gồm :

1. Address_box.dart
2. Address_short_details_card.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/services/db/user_db_helper.dart';
import 'package:flutter_online_shop/services/steams/address_stream.dart';
```

```
import 'package:flutter_online_shop/src/edit_address/edit_adress_screen.dart';
import 'package:flutter_online_shop/src/manager_address/widget/address_box.dart';
import
'package:flutter_online_shop/src/manager_address/widget/address_short_details_card.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/default_button.dart';
import 'package:flutter_online_shop/widget/nothing_show.dart';
import 'package:logger/logger.dart';

class ManageAddressesScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: ManagerAddressBody(),
    );
  }
}

class ManagerAddressBody extends StatefulWidget {
  @override
  _ManagerAddressBodyState createState() => _ManagerAddressBodyState();
}

class _ManagerAddressBodyState extends State<ManagerAddressBody> {
  final AddressesStream addressesStream = AddressesStream();
  @override
  void initState() {
    super.initState();
    addressesStream.init();
  }

  @override
  void dispose() {
    super.dispose();
    addressesStream.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: RefreshIndicator(
        onRefresh: refreshPage,
        child: SingleChildScrollView(
          physics: AlwaysScrollableScrollPhysics(),
          child: Container(
            height: SizeConfig.screenHeight,
```

```
child: Padding(
  padding: EdgeInsets.symmetric(
      horizontal: getProportionateScreenWidth(screenPadding)),
  child: SizedBox(
    width: double.infinity,
    child: Column(
      children: [
        SizedBox(height: getProportionateScreenHeight(10)),
        Container(
          width: SizeConfig.screenWidth * .9,
          decoration: BoxDecoration(
            color: Colors.red,
            borderRadius: BorderRadius.circular(20),
          ),
          padding: EdgeInsets.all(SizeConfig.screenWidth * .01),
          child: Row(
            children: [
              Icon(
                Icons.help,
                color: Colors.white,
              ),
              SizedBox(
                width: SizeConfig.screenWidth * .03,
              ),
              Text(
                "Vuốt sang TRÁI để CHỈNH SỬA - Vuốt sang PHẢI để XOÁ"
                    .toUpperCase(),
                style: TextStyle(
                    fontSize: SizeConfig.screenWidth * .025,
                    color: Colors.white),
              ),
            ],
          ),
        ),
        SizedBox(height: getProportionateScreenHeight(10)),
        DefaultButton(
          text: "Thêm địa chỉ mới",
          press: () async {
            Navigator.push(
              context,
              MaterialPageRoute(
                builder: (context) => EditAddressScreen(),
              ),
            );
          },
        ),
        SizedBox(height: getProportionateScreenHeight(10)),
        SizedBox(
```

```
                    height: SizeConfig.screenHeight * 0.7,
                    child: StreamBuilder<List<String>>(
                      stream: addressesStream.stream,
                      builder: (context, snapshot) {
                        if (snapshot.hasData) {
                          final addresses = snapshot.data;
                          if (addresses.length == 0) {
                            return Center(
                              child: NothingToShowContainer(
                                iconPath: "assets/icons/add_location.svg",
                                secondaryMessage: "Thêm địa chỉ đầu tiên",
                              ),
                            );
                          }
                          return ListView.builder(
                              physics: BouncingScrollPhysics(),
                              itemCount: addresses.length,
                              itemBuilder: (context, index) {
                                return buildAddressItemCard(addresses[index]);
                              });
                        } else if (snapshot.connectionState ==
                            ConnectionState.waiting) {
                          return Center(
                            child: CircularProgressIndicator(),
                          );
                        } else if (snapshot.hasError) {
                          final error = snapshot.error;
                          Logger().w(error.toString());
                        }
                        return Center(
                          child: NothingToShowContainer(
                            iconPath: "assets/icons/network_error.svg",
                            primaryMessage: "Something went wrong",
                            secondaryMessage: "Unable to connect to Database",
                          ),
                        );
                      },
                    ),
                  ),
                ],
              ),
            ),
          ),
        ),
      ),
    ),
  );
}
```

```
Future<void> refreshPage() {
  addressesStream.reload();
  return Future<void>.value();
}

Future<bool> deleteButtonCallback(
    BuildContext context, String addressId) async {
  final confirmDeletion = await showDialog(
    context: context,
    builder: (context) {
      return AlertDialog(
        title: Text("Thông boá"),
        content: Text("Bạn có chắc xoá địa chỉ này không?"),
        actions: [
          FlatButton(
            child: Text("Có"),
            onPressed: () {
              Navigator.pop(context, true);
            },
          ),
          FlatButton(
            child: Text("Không"),
            onPressed: () {
              Navigator.pop(context, false);
            },
          ),
        ],
      );
    },
  );
  if (confirmDeletion) {
    bool status = false;
    String snackbarMessage;
    try {
      status =
          await UserDatabaseHelper().deleteAddressForCurrentUser(addressId);
      if (status == true) {
        snackbarMessage = "Xoá địa chỉ thành công";
      } else {
        throw "Thao tác thất bại,vui lòng thử lại";
      }
    } on FirebaseException catch (e) {
      Logger().w("Firebase Exception: $e");
      snackbarMessage = "Thao tác thất bại,vui lòng thử lại";
    } catch (e) {
      Logger().w("Unknown Exception: $e");
      snackbarMessage = e.toString();
```

```
    } finally {
      Logger().i(snackbarMessage);
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text(snackbarMessage),
        ),
      );
    }
    await refreshPage();
    return status;
  }
  return false;
}

Future<bool> editButtonCallback(
    BuildContext context, String addressId) async {
  await Navigator.push(
      context,
      MaterialPageRoute(
          builder: (context) =>
              EditAddressScreen(addressIdToEdit: addressId)));
  await refreshPage();
  return false;
}

Future<void> addressItemTapCallback(String addressId) async {
  await showDialog(
    context: context,
    builder: (context) {
      return SimpleDialog(
        backgroundColor: Colors.transparent,
        title: AddressBox(
          addressId: addressId,
        ),
        titlePadding: EdgeInsets.zero,
      );
    },
  );
  await refreshPage();
}

Widget buildAddressItemCard(String addressId) {
  return Padding(
    padding: const EdgeInsets.symmetric(
      vertical: 6,
    ),
    child: Dismissible(
      key: Key(addressId),
```

```
      direction: DismissDirection.horizontal,
      background: buildDismissibleSecondaryBackground(),
      secondaryBackground: buildDismissiblePrimaryBackground(),
      dismissThresholds: {
        DismissDirection.endToStart: 0.65,
        DismissDirection.startToEnd: 0.65,
      },
      child: AddressShortDetailsCard(
        addressId: addressId,
        onTap: () async {
          await addressItemTapCallback(addressId);
        },
      ),
      confirmDismiss: (direction) async {
        if (direction == DismissDirection.startToEnd) {
          final status = await deleteButtonCallback(context, addressId);
          return status;
        } else if (direction == DismissDirection.endToStart) {
          final status = await editButtonCallback(context, addressId);
          return status;
        }
        return false;
      },
      onDismissed: (direction) async {
        await refreshPage();
      },
    ),
  );
}

Widget buildDismissiblePrimaryBackground() {
  return Container(
    padding: EdgeInsets.only(right: 20),
    decoration: BoxDecoration(
      color: Colors.green,
      borderRadius: BorderRadius.circular(15),
    ),
    child: Row(
      crossAxisAlignment: CrossAxisAlignment.center,
      mainAxisAlignment: MainAxisAlignment.end,
      children: [
        Icon(
          Icons.edit,
          color: Colors.white,
        ),
        SizedBox(width: 4),
        Text(
          "Chỉnh sửa",
```

```
            style: TextStyle(
              color: Colors.white,
              fontWeight: FontWeight.bold,
              fontSize: 15,
            ),
          ),
        ],
      ),
    );
  }

  Widget buildDismissibleSecondaryBackground() {
    return Container(
      padding: EdgeInsets.only(left: 20),
      decoration: BoxDecoration(
        color: Colors.red,
        borderRadius: BorderRadius.circular(15),
      ),
      child: Row(
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
          Text(
            "Xoá",
            style: TextStyle(
              color: Colors.white,
              fontWeight: FontWeight.bold,
              fontSize: 15,
            ),
          ),
          SizedBox(width: 4),
          Icon(
            Icons.delete,
            color: Colors.white,
          ),
        ],
      ),
    );
  }
}
```

10.1.ADDRESS BOX.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/address_model.dart';
import 'package:flutter_online_shop/services/db/user_db_helper.dart';
import 'package:flutter_online_shop/utils/constant.dart';
```

```
import 'package:logger/logger.dart';

class AddressBox extends StatelessWidget {
  const AddressBox({
    Key key,
    @required this.addressId,
  }) : super(key: key);

  final String addressId;

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.symmetric(
        vertical: 8,
      ),
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          SizedBox(
            width: double.infinity,
            child: Container(
              padding: EdgeInsets.all(24),
              decoration: BoxDecoration(
                color: Colors.white,
                borderRadius: BorderRadius.circular(30),
              ),
              child: FutureBuilder<Address>(
                  future: UserDatabaseHelper().getAddressFromId(addressId),
                  builder: (context, snapshot) {
                    if (snapshot.hasData) {
                      final address = snapshot.data;
                      return Column(
                        crossAxisAlignment: CrossAxisAlignment.start,
                        children: [
                          Text(
                            "${address.title}",
                            style: TextStyle(
                              fontSize: 22,
                              color: Colors.black,
                              fontWeight: FontWeight.bold,
                            ),
                            textAlign: TextAlign.center,
                          ),
                          Text(
                            "${address.receiver}",
                            style: TextStyle(
                              fontSize: 16,
```

```
    ),
  ),
  Text(
    "${address.addresLine1}",
    style: TextStyle(
      fontSize: 16,
    ),
  ),
  Text(
    "${address.addressLine2}",
    style: TextStyle(
      fontSize: 16,
    ),
  ),
  Text(
    "City: ${address.city}",
    style: TextStyle(
      fontSize: 16,
    ),
  ),
  Text(
    "District: ${address.district}",
    style: TextStyle(
      fontSize: 16,
    ),
  ),
  Text(
    "State: ${address.state}",
    style: TextStyle(
      fontSize: 16,
    ),
  ),
  Text(
    "Landmark: ${address.landmark}",
    style: TextStyle(
      fontSize: 16,
    ),
  ),
  Text(
    "PIN: ${address.pincode}",
    style: TextStyle(
      fontSize: 16,
    ),
  ),
  Text(
    "Phone: ${address.phone}",
    style: TextStyle(
      fontSize: 16,
```

```
                    ),
                  ),
                ],
              );
            } else if (snapshot.connectionState ==
                ConnectionState.waiting) {
              return Center(
                child: CircularProgressIndicator(),
              );
            } else if (snapshot.hasError) {
              final error = snapshot.error.toString();
              Logger().e(error);
            }
            return Center(
              child: Icon(
                Icons.error,
                color: kTextColor,
                size: 60,
              ),
            );
          }),
      ),
    ),
  ],
),
);
}
}
```

10.2.ADDRESS SHORT DETALS CARD.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/address_model.dart';
import 'package:flutter_online_shop/services/db/user_db_helper.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:logger/logger.dart';

class AddressShortDetailsCard extends StatelessWidget {
  final String addressId;
  final Function onTap;

  const AddressShortDetailsCard(
      {Key key, @required this.addressId, @required this.onTap})
      : super(key: key);
```

```
@override
Widget build(BuildContext context) {
  return InkWell(
    onTap: onTap,
    child: SizedBox(
      width: double.infinity,
      height: SizeConfig.screenHeight * 0.15,
      child: FutureBuilder<Address>(
        future: UserDatabaseHelper().getAddressFromId(addressId),
        builder: (context, snapshot) {
          if (snapshot.hasData) {
            final address = snapshot.data;
            return Row(
              children: [
                Expanded(
                  flex: 3,
                  child: Container(
                    height: double.infinity,
                    padding:
                        EdgeInsets.symmetric(horizontal: 8, vertical: 12),
                    decoration: BoxDecoration(
                      color: kPrimaryColor.withOpacity(0.3),
                      borderRadius: BorderRadius.only(
                        topLeft: Radius.circular(15),
                        bottomLeft: Radius.circular(15),
                      ),
                    ),
                    child: Center(
                      child: Text(
                        address.title,
                        style: TextStyle(
                          color: Colors.black,
                          fontSize: 18,
                          fontWeight: FontWeight.w600,
                        ),
                      ),
                    ),
                  ),
                ),
                Expanded(
                  flex: 8,
                  child: Container(
                    padding: EdgeInsets.symmetric(
                      horizontal: 8,
                      vertical: 12,
                    ),
                    decoration: BoxDecoration(
                      border: Border.all(color: kTextColor.withOpacity(0.24)),
```

```
                    borderRadius: BorderRadius.only(
                      topRight: Radius.circular(15),
                      bottomRight: Radius.circular(15),
                    ),
                  ),
                  child: Column(
                    crossAxisAlignment: CrossAxisAlignment.start,
                    mainAxisAlignment: MainAxisAlignment.center,
                    children: [
                      Text(
                        address.receiver,
                        style: TextStyle(
                          color: Colors.black,
                          fontWeight: FontWeight.bold,
                          fontSize: 15,
                        ),
                      ),
                      Text("City: ${address.city}"),
                      Text("Phone: ${address.phone}"),
                    ],
                  ),
                ),
              ),
            ],
          );
        } else if (snapshot.connectionState == ConnectionState.waiting) {
          return Center(child: CircularProgressIndicator());
        } else if (snapshot.hasError) {
          final error = snapshot.error.toString();
          Logger().e(error);
        }
        return Center(
          child: Icon(
            Icons.error,
            size: 40,
            color: kTextColor,
          ),
        );
      },
    ),
  ),
);
}
}
```

11.MY ORDERS SCREEN.DART

My Orders Screen chứa các widget con gồm :

1. Product_review_dialog.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/order_product_model.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/models/review_model.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/services/db/user_db_helper.dart';
import 'package:flutter_online_shop/services/steams/order_product_stream.dart';
import 'package:flutter_online_shop/src/orders/widget/product_review_dialog.dart';
import 'package:flutter_online_shop/src/product_details/product_details_screen.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/nothing_show.dart';
import 'package:flutter_online_shop/widget/product_short_detail_card.dart';
import 'package:logger/logger.dart';

class MyOrdersScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: OrdersBody(),
    );
  }
}

class OrdersBody extends StatefulWidget {
  @override
  _OrdersBodyState createState() => _OrdersBodyState();
}

class _OrdersBodyState extends State<OrdersBody> {
  final OrderedProductsStream orderedProductsStream = OrderedProductsStream();

  @override
  void initState() {
    super.initState();
    orderedProductsStream.init();
  }

  @override
```

```
void dispose() {
  super.dispose();
  orderedProductsStream.dispose();
}

@override
Widget build(BuildContext context) {
  return SafeArea(
    child: RefreshIndicator(
      onRefresh: refreshPage,
      child: SingleChildScrollView(
        physics: AlwaysScrollableScrollPhysics(),
        child: Padding(
          padding: EdgeInsets.symmetric(
              horizontal: getProportionateScreenWidth(screenPadding)),
          child: SizedBox(
            width: double.infinity,
            child: Column(
              children: [
                SizedBox(height: getProportionateScreenHeight(10)),
                Text(
                  "Đơn Đặt Hàng".toUpperCase(),
                  style: TextStyle(
                    fontSize: getProportionateScreenWidth(30),
                    fontWeight: FontWeight.bold,
                    color: Colors.teal[800],
                    height: 1.5,
                  ),
                ),
                SizedBox(height: getProportionateScreenHeight(20)),
                SizedBox(
                  height: SizeConfig.screenHeight * 0.75,
                  child: buildOrderedProductsList(),
                ),
              ],
            ),
          ),
        ),
      ),
    ),
  );
}

Future<void> refreshPage() {
  orderedProductsStream.reload();
  return Future<void>.value();
}
```

```
Widget buildOrderedProductsList() {
  return StreamBuilder<List<String>>(
    stream: orderedProductsStream.stream,
    builder: (context, snapshot) {
      if (snapshot.hasData) {
        final orderedProductsIds = snapshot.data;
        if (orderedProductsIds.length == 0) {
          return Center(
            child: NothingToShowContainer(
              iconPath: "assets/icons/empty_bag.svg",
              secondaryMessage: "Đặt hàng thứ gì đó để hiển thị ở đây",
            ),
          );
        }
        return ListView.builder(
          physics: BouncingScrollPhysics(),
          itemCount: orderedProductsIds.length,
          itemBuilder: (context, index) {
            return FutureBuilder<OrderedProduct>(
              future: UserDatabaseHelper()
                  .getOrderedProductFromId(orderedProductsIds[index]),
              builder: (context, snapshot) {
                if (snapshot.hasData) {
                  final orderedProduct = snapshot.data;
                  return buildOrderedProductItem(orderedProduct);
                } else if (snapshot.connectionState ==
                    ConnectionState.waiting) {
                  return Center(child: CircularProgressIndicator());
                } else if (snapshot.hasError) {
                  final error = snapshot.error.toString();
                  Logger().e(error);
                }
                return Icon(
                  Icons.error,
                  size: 60,
                  color: kTextColor,
                );
              },
            );
          },
        );
      } else if (snapshot.connectionState == ConnectionState.waiting) {
        return Center(
          child: CircularProgressIndicator(),
        );
      } else if (snapshot.hasError) {
        final error = snapshot.error;
        Logger().w(error.toString());
```

```
      }
      return Center(
        child: NothingToShowContainer(
          iconPath: "assets/icons/network_error.svg",
          primaryMessage: "Something went wrong",
          secondaryMessage: "Unable to connect to Database",
        ),
      );
    },
  );
}

Widget buildOrderedProductItem(OrderedProduct orderedProduct) {
  return FutureBuilder<Product>(
    future:
        ProductDatabaseHelper().getProductWithID(orderedProduct.productUid),
    builder: (context, snapshot) {
      if (snapshot.hasData) {
        final product = snapshot.data;
        return Padding(
          padding: EdgeInsets.symmetric(vertical: 6),
          child: Column(
            children: [
              Container(
                padding: EdgeInsets.symmetric(
                  vertical: 12,
                  horizontal: 16,
                ),
                width: double.infinity,
                decoration: BoxDecoration(
                  color: kTextColor.withOpacity(0.12),
                  borderRadius: BorderRadius.only(
                    topLeft: Radius.circular(16),
                    topRight: Radius.circular(16),
                  ),
                ),
                child: Text.rich(
                  TextSpan(
                    text: "Ngày đặt:  ",
                    style: TextStyle(
                      color: Colors.black,
                      fontSize: 12,
                    ),
                    children: [
                      TextSpan(
                        text: orderedProduct.orderDate,
                        style: TextStyle(
                          fontWeight: FontWeight.bold,
```

```
          ),
        ),
      ],
    ),
  ),
),
Container(
  padding: EdgeInsets.symmetric(
    horizontal: 4,
    vertical: 8,
  ),
  decoration: BoxDecoration(
    border: Border.symmetric(
      vertical: BorderSide(
        color: kTextColor.withOpacity(0.15),
      ),
    ),
  ),
  child: ProductShortDetailCard(
    productId: product.id,
    onPressed: () {
      Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => ProductDetailsScreen(
            productId: product.id,
          ),
        ),
      ).then((_) async {
        await refreshPage();
      });
    },
  ),
),
Container(
  width: double.infinity,
  padding: EdgeInsets.symmetric(
    horizontal: 8,
    vertical: 2,
  ),
  decoration: BoxDecoration(
    color: kPrimaryColor.withOpacity(.8),
    borderRadius: BorderRadius.only(
      bottomLeft: Radius.circular(16),
      bottomRight: Radius.circular(16),
    ),
  ),
  child: FlatButton(
```

```
onPressed: () async {
  String currentUserUid =
      AuthentificationService().currentUser.uid;
  Review prevReview;
  try {
    prevReview = await ProductDatabaseHelper()
        .getProductReviewWithID(product.id, currentUserUid);
  } on FirebaseException catch (e) {
    Logger().w("Firebase Exception: $e");
  } catch (e) {
    Logger().w("Unknown Exception: $e");
  } finally {
    if (prevReview == null) {
      prevReview = Review(
        currentUserUid,
        reviewerUid: currentUserUid,
      );
    }
  }

  final result = await showDialog(
    context: context,
    builder: (context) {
      return ProductReviewDialog(
        review: prevReview,
      );
    },
  );
  if (result is Review) {
    bool reviewAdded = false;
    String snackbarMessage;
    try {
      reviewAdded = await ProductDatabaseHelper()
          .addProductReview(product.id, result);
      if (reviewAdded == true) {
        snackbarMessage = "Gửi đánh giá thành công";
      } else {
        throw "Thao tác thất bại,vui lòng thử lại";
      }
    } on FirebaseException catch (e) {
      Logger().w("Firebase Exception: $e");
      snackbarMessage = e.toString();
    } catch (e) {
      Logger().w("Unknown Exception: $e");
      snackbarMessage = e.toString();
    } finally {
      Logger().i(snackbarMessage);
      ScaffoldMessenger.of(context).showSnackBar(
```

```
                        SnackBar(
                          content: Text(snackbarMessage),
                        ),
                      );
                    }
                  }
                  await refreshPage();
                },
                child: Text(
                  "Đánh giá sản phẩm",
                  style: TextStyle(
                    color: Colors.white,
                    fontWeight: FontWeight.w600,
                    fontSize: 16,
                  ),
                ),
              ),
            ),
          ],
        ),
      );
    } else if (snapshot.connectionState == ConnectionState.waiting) {
      return Center(child: CircularProgressIndicator());
    } else if (snapshot.hasError) {
      final error = snapshot.error.toString();
      Logger().e(error);
    }
    return Icon(
      Icons.error,
      size: 60,
      color: kTextColor,
    );
  },
);
}
}
```

11.1.PRODUCT REVIEW DIALOG.DART

```
import 'package:flutter/material.dart';

import 'package:flutter_online_shop/models/review_model.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/default_button.dart';
import 'package:flutter_rating_bar/flutter_rating_bar.dart';

class ProductReviewDialog extends StatelessWidget {
  final Review review;
```

```
ProductReviewDialog({
  Key key,
  @required this.review,
}) : super(key: key);

@override
Widget build(BuildContext context) {
  return SimpleDialog(
    title: Center(
      child: Text(
        "Đánh giá",
      ),
    ),
    children: [
      Center(
        child: RatingBar.builder(
          initialRating: review.rating.toDouble(),
          minRating: 1,
          direction: Axis.horizontal,
          allowHalfRating: false,
          itemCount: 5,
          itemPadding: EdgeInsets.symmetric(horizontal: 4.0),
          itemBuilder: (context, _) => Icon(
            Icons.star,
            color: Colors.amber,
          ),
          onRatingUpdate: (rating) {
            review.rating = rating.round();
          },
        ),
      ),
      SizedBox(height: getProportionateScreenHeight(20)),
      Center(
        child: TextFormField(
          initialValue: review.feedback,
          decoration: InputDecoration(
            hintText: "Phản hồi về sản phẩm",
            labelText: "Phản hồi",
            floatingLabelBehavior: FloatingLabelBehavior.always,
          ),
          onChanged: (value) {
            review.feedback = value;
          },
          maxLines: null,
          maxLength: 150,
        ),
      ),
      SizedBox(height: getProportionateScreenHeight(10)),
```

```
        Center(
          child: DefaultButton(
            text: "Xác nhận".toUpperCase(),
            press: () {
              Navigator.pop(context, review);
            },
          ),
        ),
      ],
      contentPadding: EdgeInsets.symmetric(
        horizontal: 12,
        vertical: 16,
      ),
      shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(15)),
    );
  }
}
```

12. MY PRODUCT SCREEN.DART

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/services/file_access/fs_files_access.dart';
import 'package:flutter_online_shop/services/steams/user_product_stream.dart';
import 'package:flutter_online_shop/src/edit_product/edit_product_screen.dart';
import 'package:flutter_online_shop/src/product_details/product_details_screen.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/utils/utils.dart';
import 'package:flutter_online_shop/widget/nothing_show.dart';
import 'package:flutter_online_shop/widget/product_short_detail_card.dart';
import 'package:future_progress_dialog/future_progress_dialog.dart';
import 'package:logger/logger.dart';

class MyProductsScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(),
      body: ProductBody(),
      //body: Body(),
    );
  }
}

class ProductBody extends StatefulWidget {
```

```
  @override
  _ProductBodyState createState() => _ProductBodyState();
}

class _ProductBodyState extends State<ProductBody> {
  final UsersProductsStream usersProductsStream = UsersProductsStream();

  @override
  void initState() {
    super.initState();
    usersProductsStream.init();
  }

  @override
  void dispose() {
    super.dispose();
    usersProductsStream.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: RefreshIndicator(
        onRefresh: refreshPage,
        child: SingleChildScrollView(
          physics: AlwaysScrollableScrollPhysics(),
          child: Padding(
            padding: EdgeInsets.symmetric(
                horizontal: getProportionateScreenWidth(screenPadding)),
            child: SizedBox(
              width: double.infinity,
              child: Column(
                children: [
                  SizedBox(height: getProportionateScreenHeight(20)),
                  Container(
                    width: SizeConfig.screenWidth * .9,
                    decoration: BoxDecoration(
                      color: Colors.red,
                      borderRadius: BorderRadius.circular(20),
                    ),
                    padding: EdgeInsets.all(SizeConfig.screenWidth * .01),
                    child: Row(
                      children: [
                        Icon(
                          Icons.help,
                          color: Colors.white,
                        ),
                        SizedBox(
```

```
                    width: SizeConfig.screenWidth * .03,
                  ),
                  Text(
                    "Vuốt sang TRÁI để CHỈNH SỬA - Vuốt sang PHẢI để XOÁ"
                        .toUpperCase(),
                    style: TextStyle(
                        fontSize: SizeConfig.screenWidth * .025,
                        color: Colors.white),
                  ),
                ],
              ),
            ),
            SizedBox(height: getProportionateScreenHeight(30)),
            SizedBox(
              height: SizeConfig.screenHeight * 0.7,
              child: StreamBuilder<List<String>>(
                stream: usersProductsStream.stream,
                builder: (context, snapshot) {
                  if (snapshot.hasData) {
                    final productsIds = snapshot.data;
                    if (productsIds.length == 0) {
                      return Center(
                        child: NothingToShowContainer(
                          secondaryMessage:
                              "Trở thành người bán hàng của chúng tôi bằng cách thêm
sản phẩm đầu tiên nào",
                        ),
                      );
                    }
                    return ListView.builder(
                      physics: BouncingScrollPhysics(),
                      itemCount: productsIds.length,
                      itemBuilder: (context, index) {
                        return buildProductsCard(productsIds[index]);
                      },
                    );
                  } else if (snapshot.connectionState ==
                      ConnectionState.waiting) {
                    return Center(
                      child: CircularProgressIndicator(),
                    );
                  } else if (snapshot.hasError) {
                    final error = snapshot.error;
                    Logger().w(error.toString());
                  }
                  return Center(
                    child: NothingToShowContainer(
                      iconPath: "assets/icons/network_error.svg",
```

```
                        primaryMessage: "Something went wrong",
                        secondaryMessage: "Unable to connect to Database",
                      ),
                    );
                  },
                ),
              ),
              SizedBox(height: getProportionateScreenHeight(60)),
            ],
          ),
        ),
      ),
    ),
  ),
);
}

Future<void> refreshPage() {
  usersProductsStream.reload();
  return Future<void>.value();
}

Widget buildProductsCard(String productId) {
  return Padding(
    padding: const EdgeInsets.symmetric(vertical: 10),
    child: FutureBuilder<Product>(
      future: ProductDatabaseHelper().getProductWithID(productId),
      builder: (context, snapshot) {
        if (snapshot.hasData) {
          final product = snapshot.data;
          return buildProductDismissible(product);
        } else if (snapshot.connectionState == ConnectionState.waiting) {
          return Center(child: CircularProgressIndicator());
        } else if (snapshot.hasError) {
          final error = snapshot.error.toString();
          Logger().e(error);
        }
        return Center(
          child: Icon(
            Icons.error,
            color: kTextColor,
            size: 60,
          ),
        );
      },
    ),
  );
}
```

```
Widget buildProductDismissible(Product product) {
  return Dismissible(
    key: Key(product.id),
    direction: DismissDirection.horizontal,
    background: buildDismissibleSecondaryBackground(),
    secondaryBackground: buildDismissiblePrimaryBackground(),
    dismissThresholds: {
      DismissDirection.endToStart: 0.65,
      DismissDirection.startToEnd: 0.65,
    },
    child: ProductShortDetailCard(
      productId: product.id,
      onPressed: () {
        Navigator.push(
          context,
          MaterialPageRoute(
            builder: (context) => ProductDetailsScreen(
              productId: product.id,
            ),
          ),
        );
      },
    ),
    confirmDismiss: (direction) async {
      if (direction == DismissDirection.startToEnd) {
        final confirmation = await showConfirmationDialog(
            context, "Bạn có chắc muốn xoá sản phẩm này?");
        if (confirmation) {
          for (int i = 0; i < product.images.length; i++) {
            String path =
                ProductDatabaseHelper().getPathForProductImage(product.id, i);
            final deletionFuture =
                FirestoreFilesAccess().deleteFileFromPath(path);
            await showDialog(
              context: context,
              builder: (context) {
                return FutureProgressDialog(
                  deletionFuture,
                  message:
                      Text("Đang xoá ảnh ${i + 1}/${product.images.length}"),
                );
              },
            );
          }

          bool productInfoDeleted = false;
          String snackbarMessage;
```

```
      try {
        final deleteProductFuture =
            ProductDatabaseHelper().deleteUserProduct(product.id);
        productInfoDeleted = await showDialog(
          context: context,
          builder: (context) {
            return FutureProgressDialog(
              deleteProductFuture,
              message: Text("Đang xoá sản phẩm"),
            );
          },
        );
        if (productInfoDeleted == true) {
          snackbarMessage = "Loại bỏ sản phẩm thành công";
        } else {
          throw "Thao tác thất bại,vui lòng thử lại";
        }
      } on FirebaseException catch (e) {
        Logger().w("Firebase Exception: $e");
        snackbarMessage = "Something went wrong";
      } catch (e) {
        Logger().w("Unknown Exception: $e");
        snackbarMessage = e.toString();
      } finally {
        Logger().i(snackbarMessage);
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: Text(snackbarMessage),
          ),
        );
      }
    }
    await refreshPage();
    return confirmation;
  } else if (direction == DismissDirection.endToStart) {
    final confirmation = await showConfirmationDialog(
        context, "Bạn có chắc muốn chỉnh sửa không?");
    if (confirmation) {
      await Navigator.push(
        context,
        MaterialPageRoute(
          builder: (context) => EditProductScreen(
            productToEdit: product,
          ),
        ),
      );
    }
    await refreshPage();
```

```
        return false;
      }
      return false;
    },
    onDismissed: (direction) async {
      await refreshPage();
    },
  );
}

Widget buildDismissiblePrimaryBackground() {
  return Container(
    padding: EdgeInsets.only(right: 20),
    decoration: BoxDecoration(
      color: Colors.green,
      borderRadius: BorderRadius.circular(15),
    ),
    child: Row(
      crossAxisAlignment: CrossAxisAlignment.center,
      mainAxisAlignment: MainAxisAlignment.end,
      children: [
        Icon(
          Icons.edit,
          color: Colors.white,
        ),
        SizedBox(width: 4),
        Text(
          "Chỉnh sửa",
          style: TextStyle(
            color: Colors.white,
            fontWeight: FontWeight.bold,
            fontSize: 15,
          ),
        ),
      ],
    ),
  );
}

Widget buildDismissibleSecondaryBackground() {
  return Container(
    padding: EdgeInsets.only(left: 20),
    decoration: BoxDecoration(
      color: Colors.red,
      borderRadius: BorderRadius.circular(15),
    ),
    child: Row(
      crossAxisAlignment: CrossAxisAlignment.center,
```

```
        mainAxisAlignment: MainAxisAlignment.start,
        children: [
          Text(
            "Xoá",
            style: TextStyle(
              color: Colors.white,
              fontWeight: FontWeight.bold,
              fontSize: 15,
            ),
          ),
          SizedBox(width: 4),
          Icon(
            Icons.delete,
            color: Colors.white,
          ),
        ],
      ),
    );
  }
}
```

13. PRODUCT DETAILS SCREEN.DART

My Orders Screen chứa các widget con gồm :

1. body.dart
2. add_to_cart.dart
3. expand_text_widget.dart
4. product_actions_sections.dart
5. product_description.dart
6. product_image.dart
7. product_review_sections.dart
8. review_box.dart
9. expand_text.dart (Provider)
10. product_actions.dart (Provider)
11. product_img_swiper.dart (Provider)

13.1. BODY.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
```

```
import
'package:flutter_online_shop/src/product_details/widget/product_actions_sections.dart';
import 'package:flutter_online_shop/src/product_details/widget/product_image.dart';
import
'package:flutter_online_shop/src/product_details/widget/product_review_sections.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:logger/logger.dart';

class Body extends StatelessWidget {
  final String productId;

  const Body({
    Key key,
    @required this.productId,
  }) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return SafeArea(
      child: SingleChildScrollView(
        physics: BouncingScrollPhysics(),
        child: Padding(
          padding: EdgeInsets.symmetric(
              horizontal: getProportionateScreenWidth(screenPadding),
              vertical: getProportionateScreenHeight(screenPadding)),
          child: FutureBuilder<Product>(
            future: ProductDatabaseHelper().getProductWithID(productId),
            builder: (context, snapshot) {
              if (snapshot.hasData) {
                final product = snapshot.data;
                return Column(
                  children: [
                    ProductImages(product: product),
                    SizedBox(height: getProportionateScreenHeight(20)),
                    ProductActionsSection(product: product),
                    SizedBox(height: getProportionateScreenHeight(20)),
                    ProductReviewsSection(product: product),
                    SizedBox(height: getProportionateScreenHeight(100)),
                  ],
                );
              } else if (snapshot.connectionState == ConnectionState.waiting) {
                return Center(child: CircularProgressIndicator());
              } else if (snapshot.hasError) {
                final error = snapshot.error.toString();
                Logger().e(error);
              }
              return Center(
                child: Icon(
```

```
                Icons.error,
                color: kTextColor,
                size: 60,
              ),
            );
          },
        ),
      ),
    ),
  );
 }
}
```

13.2. ADD_TO_CART.DART

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/services/db/user_db_helper.dart';
import 'package:flutter_online_shop/utils/utils.dart';
import 'package:future_progress_dialog/future_progress_dialog.dart';
import 'package:logger/logger.dart';

class AddToCartFAB extends StatelessWidget {
  const AddToCartFAB({
    Key key,
    @required this.productId,
  }) : super(key: key);

  final String productId;

  @override
  Widget build(BuildContext context) {
    return FloatingActionButton.extended(
      onPressed: () async {
        bool allowed = AuthentificationService().currentUserVerified;
        if (!allowed) {
          final reverify = await showConfirmationDialog(context,
              "Bạn chưa xác minh địa chỉ email của mình. Hành động này chỉ được phép đối
với những người dùng đã xác minh.",
              positiveResponse: "Gửi email xác minh",
              negativeResponse: "Trở về");
          if (reverify) {
            final future =
                AuthentificationService().sendVerificationEmailToCurrentUser();
            await showDialog(
```

```
          context: context,
          builder: (context) {
            return FutureProgressDialog(
              future,
              message: Text(
                  "Đang tiến hành gửi xác minh.\nVui lòng đợi trong giây lát"),
            );
          },
        );
      }
      return;
    }
    bool addedSuccessfully = false;
    String snackbarMessage;
    try {
      addedSuccessfully =
          await UserDatabaseHelper().addProductToCart(productId);
      if (addedSuccessfully == true) {
        snackbarMessage = "Thêm sản phẩm thành công";
      } else {
        throw "Thao tác thất bại,vui lòng thử lại";
      }
    } on FirebaseException catch (e) {
      Logger().w("Firebase Exception: $e");
      snackbarMessage = "Thao tác thất bại,vui lòng thử lại";
    } catch (e) {
      Logger().w("Unknown Exception: $e");
      snackbarMessage = "Thao tác thất bại,vui lòng thử lại";
    } finally {
      Logger().i(snackbarMessage);
      ScaffoldMessenger.of(context).showSnackBar(
        SnackBar(
          content: Text(snackbarMessage),
        ),
      );
    }
  },
  label: Text(
    "Thêm giỏ hàng",
    style: TextStyle(
      fontWeight: FontWeight.bold,
      fontSize: 16,
    ),
  ),
  icon: Icon(
    Icons.shopping_cart,
  ),
);
```

```
  }
}
```

13.3. EXPAND_TEXT_WIDGET.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/src/product_details/widget/provider/expand_text.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:provider/provider.dart';

class ExpandableText extends StatelessWidget {
  final String title;
  final String content;
  final int maxLines;
  const ExpandableText({
    Key key,
    @required this.title,
    @required this.content,
    this.maxLines = 3,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (context) => ExpandText(),
      child: Consumer<ExpandText>(builder: (context, expandText, child) {
        return Column(
          crossAxisAlignment: CrossAxisAlignment.start,
          children: [
            Text(
              title,
              style: TextStyle(
                color: Colors.black54,
                fontWeight: FontWeight.w600,
                fontSize: 16,
              ),
            ),
            Divider(
              height: 8,
              thickness: 1,
              endIndent: 16,
            ),
            Text(
              content,
              maxLines: expandText.isExpanded ? null : maxLines,
```

```
            textAlign: TextAlign.left,
          ),
          GestureDetector(
            onTap: () {
              expandText.isExpanded ^= true;
            },
            child: Row(
              children: [
                Text(
                  expandText.isExpanded == false
                      ? "Xem thêm".toUpperCase()
                      : "Thu gọn".toUpperCase(),
                  style: TextStyle(
                    color: kPrimaryColor,
                    fontWeight: FontWeight.w600,
                  ),
                ),
                const SizedBox(width: 5),
                Icon(
                  Icons.arrow_forward_ios,
                  size: 12,
                  color: kPrimaryColor,
                ),
              ],
            ),
          ),
        ],
      );
    }),
  );
 }
}
```

13.4. PRODUCT_ACTIONS_SECTIONS.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/services/db/user_db_helper.dart';
import 'package:flutter_online_shop/src/product_details/widget/product_description.dart';
import
'package:flutter_online_shop/src/product_details/widget/provider/product_actions.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/utils/utils.dart';
import 'package:flutter_online_shop/widget/top_round_container.dart';
```

```
import 'package:future_progress_dialog/future_progress_dialog.dart';
import 'package:logger/logger.dart';
import 'package:provider/provider.dart';

class ProductActionsSection extends StatelessWidget {
  final Product product;

  const ProductActionsSection({
    Key key,
    @required this.product,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final column = Column(
      children: [
        Stack(
          children: [
            TopRoundedContainer(
              child: ProductDescription(product: product),
            ),
          ],
        ),
      ],
    );
    UserDatabaseHelper().isProductFavourite(product.id).then(
      (value) {
        final productActions =
            Provider.of<ProductActions>(context, listen: false);
        productActions.productFavStatus = value;
      },
    ).catchError(
      (e) {
        Logger().w("$e");
      },
    );
    return column;
  }

  Widget buildFavouriteButton() {
    return Consumer<ProductActions>(
      builder: (context, productDetails, child) {
        return InkWell(
          onTap: () async {
            bool allowed = AuthentificationService().currentUserVerified;
            if (!allowed) {
              final reverify = await showConfirmationDialog(context,
```

```
                "Bạn chưa xác minh địa chỉ email của mình. Hành động này chỉ được phép
đối với những người dùng đã xác minh.",
                positiveResponse: "Gửi email xác minh",
                negativeResponse: "Trở về");
            if (reverify) {
              final future = AuthentificationService()
                  .sendVerificationEmailToCurrentUser();
              await showDialog(
                context: context,
                builder: (context) {
                  return FutureProgressDialog(
                    future,
                    message: Text(
                        "Đang tiến hành gửi xác minh.\nVui lòng đợi trong giây lát"),
                  );
                },
              );
            }
            return;
          }
          bool success = false;
          final future = UserDatabaseHelper()
              .switchProductFavouriteStatus(
                  product.id, !productDetails.productFavStatus)
              .then(
            (status) {
              success = status;
            },
          ).catchError(
            (e) {
              Logger().e(e.toString());
              success = false;
            },
          );
          await showDialog(
            context: context,
            builder: (context) {
              return FutureProgressDialog(
                future,
                message: Text(
                  productDetails.productFavStatus
                      ? "Removing from Favourites"
                      : "Adding to Favourites",
                ),
              );
            },
          );
          if (success) {
```

```
            productDetails.switchProductFavStatus();
          }
        },
        child: Container(
          padding: EdgeInsets.all(getProportionateScreenWidth(8)),
          decoration: BoxDecoration(
            color: productDetails.productFavStatus
                ? Color(0xFFFFE6E6)
                : Color(0xFFF5F6F9),
            shape: BoxShape.circle,
            border: Border.all(color: Colors.white, width: 4),
          ),
          child: Padding(
            padding: EdgeInsets.all(getProportionateScreenWidth(8)),
            child: Icon(
              Icons.favorite,
              color: productDetails.productFavStatus
                  ? Color(0xFFFF4848)
                  : Color(0xFFD8DEE4),
            ),
          ),
        ),
      );
    },
  );
 }
}
```

13.5.PRODUCT_DESCRIPTION.DART

```
import 'dart:math';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/src/product_details/widget/expand_text_widget.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_svg/flutter_svg.dart';

class ProductDescription extends StatelessWidget {
  const ProductDescription({
    Key key,
    @required this.product,
  }) : super(key: key);
```

```
final Product product;
@override
Widget build(BuildContext context) {
  return Stack(
    children: [
      Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          Text.rich(
            TextSpan(
                text: product.title,
                style: TextStyle(
                  fontSize: 21,
                  color: Colors.black,
                  fontWeight: FontWeight.w600,
                ),
                children: [
                  TextSpan(
                    text: "\nColor:\t${product.variant} ",
                    style: TextStyle(
                      fontWeight: FontWeight.normal,
                      fontSize: 15,
                    ),
                  ),
                ]),
          ),
          const SizedBox(height: 16),
          SizedBox(
            height: getProportionateScreenHeight(64),
            child: Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: [
                Flexible(
                  flex: 4,
                  child: Text.rich(
                    TextSpan(
                      text: "${product.discountPrice}\đ   ",
                      style: TextStyle(
                        color: kPrimaryColor,
                        fontWeight: FontWeight.w900,
                        fontSize: 24,
                      ),
                      children: [
                        TextSpan(
                          text: "\n${product.originalPrice}\đ",
                          style: TextStyle(
                            decoration: TextDecoration.lineThrough,
                            color: kTextColor,
```

```
                fontWeight: FontWeight.normal,
                fontSize: 16,
              ),
            ),
          ],
        ),
      ),
    ),
    Flexible(
      flex: 3,
      child: Stack(
        children: [
          SvgPicture.asset(
            "assets/icons/Discount.svg",
            color: kPrimaryColor,
          ),
          Center(
            child: Padding(
              padding: EdgeInsets.only(
                  right: getProportionateScreenWidth(
                      screenPadding * .7),
                  top: getProportionateScreenHeight(
                      screenPadding * 1.5)),
              child: Transform.rotate(
                angle: 30 * pi / 180,
                child: Text(
                  "${product.calculatePercentageDiscount()}%\n",
                  style: TextStyle(
                    color: Colors.white,
                    fontSize: getProportionateScreenHeight(20),
                    fontWeight: FontWeight.w900,
                  ),
                  textAlign: TextAlign.center,
                ),
              ),
            ),
          ),
        ],
      ),
    ),
  ],
),
),
const SizedBox(height: 16),
ExpandableText(
  title: "Chi tiết sản phẩm".toUpperCase(),
  content: product.highlights,
),
```

```
            const SizedBox(height: 16),
            ExpandableText(
              title: "Miêu tả sản phẩm".toUpperCase(),
              content: product.description,
            ),
            const SizedBox(height: 16),
            Row(
              mainAxisAlignment: MainAxisAlignment.spaceEvenly,
              children: [
                Text(
                  "Người đăng: ",
                  style: TextStyle(
                    fontSize: 15,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                Container(
                  width: SizeConfig.screenWidth * .4,
                  height: SizeConfig.screenHeight * .05,
                  decoration: BoxDecoration(
                      borderRadius: BorderRadius.circular(30),
                      color: kPrimaryColor.withOpacity(.8)),
                  child: Center(
                    child: Text(
                      "${product.seller}",
                      maxLines: 1,
                      overflow: TextOverflow.ellipsis,
                      style: TextStyle(
                        fontSize: 15,
                        color: Colors.white,
                        fontWeight: FontWeight.bold,
                      ),
                    ),
                  ),
                ),
              ],
            ),
          ],
        ),
      ],
    );
  }
}
```

13.6. PRODUCT_IMAGE.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import
'package:flutter_online_shop/src/product_details/widget/provider/product_img_swiper.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:pinch_zoom_image_updated/pinch_zoom_image_updated.dart';
import 'package:provider/provider.dart';
import 'package:swipedetector/swipedetector.dart';

class ProductImages extends StatelessWidget {
  const ProductImages({
    Key key,
    @required this.product,
  }) : super(key: key);

  final Product product;

  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider(
      create: (context) => ProductImageSwiper(),
      child: Consumer<ProductImageSwiper>(
        builder: (context, productImagesSwiper, child) {
          return Column(
            children: [
              SwipeDetector(
                onSwipeLeft: () {
                  productImagesSwiper.currentImageIndex++;
                  productImagesSwiper.currentImageIndex %=
                      product.images.length;
                },
                onSwipeRight: () {
                  productImagesSwiper.currentImageIndex--;
                  productImagesSwiper.currentImageIndex +=
                      product.images.length;
                  productImagesSwiper.currentImageIndex %=
                      product.images.length;
                },
                child: PinchZoomImage(
                  image: Container(
                    padding: const EdgeInsets.all(16),
                    decoration: BoxDecoration(
                      color: Colors.white,
                      borderRadius: BorderRadius.all(
                        Radius.circular(30),
                      ),
                    ),
```

```
                child: SizedBox(
                  height: SizeConfig.screenHeight * 0.35,
                  width: SizeConfig.screenWidth * 0.75,
                  child: ClipRRect(
                    borderRadius: BorderRadius.circular(15),
                    child: Image.network(
                      product.images[productImagesSwiper.currentImageIndex],
                      fit: BoxFit.contain,
                    ),
                  ),
                ),
              ),
            ),
          ),
          SizedBox(height: 16),
          Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: [
              ...List.generate(
                product.images.length,
                (index) =>
                    buildSmallPreview(productImagesSwiper, index: index),
              ),
            ],
          ),
        ],
      );
    },
  ),
);
}

Widget buildSmallPreview(ProductImageSwiper productImagesSwiper,
    {@required int index}) {
  return GestureDetector(
    onTap: () {
      productImagesSwiper.currentImageIndex = index;
    },
    child: Container(
      margin:
          EdgeInsets.symmetric(horizontal: getProportionateScreenWidth(8)),
      padding: EdgeInsets.all(getProportionateScreenHeight(8)),
      height: getProportionateScreenWidth(48),
      width: getProportionateScreenWidth(48),
      decoration: BoxDecoration(
        color: Colors.white,
        borderRadius: BorderRadius.circular(10),
        border: Border.all(
```

```
                width: 3.0,
                color: productImagesSwiper.currentImageIndex == index
                    ? kPrimaryColor
                    : Colors.transparent),
            ),
            child: Image.network(product.images[index]),
          ),
        );
      }
    }
```

13.7. PRODUCT_REVIEW_SECTIONS.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/models/review_model.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/src/product_details/widget/review_box.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_online_shop/widget/top_round_container.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:logger/logger.dart';

class ProductReviewsSection extends StatelessWidget {
  const ProductReviewsSection({
    Key key,
    @required this.product,
  }) : super(key: key);

  final Product product;

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: getProportionateScreenHeight(320),
      child: Stack(
        children: [
          TopRoundedContainer(
            child: Column(
              children: [
                Text(
                  "Đánh giá sản phẩm",
                  style: TextStyle(
                    fontSize: 21,
                    color: Colors.black,
                    fontWeight: FontWeight.w600,
                  ),
                ),
                SizedBox(height: getProportionateScreenHeight(20)),
                Expanded(
                  child: StreamBuilder<List<Review>>(
                    stream: ProductDatabaseHelper()
                        .getAllReviewsStreamForProductId(product.id),
                    builder: (context, snapshot) {
                      if (snapshot.hasData) {
                        final reviewsList = snapshot.data;
```

```
        if (reviewsList.length == 0) {
          return Center(
            child: Column(
              mainAxisAlignment: MainAxisAlignment.center,
              children: [
                SvgPicture.asset(
                  "assets/icons/review.svg",
                  color: kTextColor,
                  width: 40,
                ),
                SizedBox(height: 8),
                Text(
                  "Chưa có đánh giá",
                  style: TextStyle(
                    fontWeight: FontWeight.bold,
                  ),
                ),
              ],
            ),
          );
        }
        return ListView.builder(
          physics: BouncingScrollPhysics(),
          itemCount: reviewsList.length,
          itemBuilder: (context, index) {
            return ReviewBox(
              review: reviewsList[index],
            );
          },
        );
      } else if (snapshot.connectionState ==
          ConnectionState.waiting) {
        return Center(
          child: CircularProgressIndicator(),
        );
      } else if (snapshot.hasError) {
        final error = snapshot.error;
        Logger().w(error.toString());
      }
      return Center(
        child: Icon(
          Icons.error,
          color: kTextColor,
          size: 50,
        ),
      );
    },
  ),
```

```
                ),
              ],
            ),
          ),
          Padding(
            padding: EdgeInsets.only(
                right: getProportionateScreenWidth(screenPadding)),
            child: Align(
              alignment: Alignment.topRight,
              child: buildProductRatingWidget(product.rating),
            ),
          ),
        ],
      ),
    );
  }

  Widget buildProductRatingWidget(num rating) {
    return Container(
      width: getProportionateScreenWidth(80),
      padding: EdgeInsets.all(12),
      decoration: BoxDecoration(
        color: Colors.yellow[900],
        borderRadius: BorderRadius.circular(14),
      ),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          Expanded(
            child: Text(
              "$rating",
              style: TextStyle(
                color: Colors.white,
                fontWeight: FontWeight.w900,
                fontSize: getProportionateScreenWidth(16),
              ),
            ),
          ),
          SizedBox(width: 5),
          Icon(
            Icons.star,
            color: Colors.white,
          ),
        ],
      ),
    );
  }
}
```

13.8. REVIEW_BOX.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/review_model.dart';
import 'package:flutter_online_shop/utils/constant.dart';

class ReviewBox extends StatelessWidget {
  final Review review;
  const ReviewBox({
    Key key,
    @required this.review,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      padding: EdgeInsets.symmetric(
        horizontal: 16,
        vertical: 8,
      ),
      margin: EdgeInsets.symmetric(
        vertical: 4,
      ),
      decoration: BoxDecoration(
        color: kPrimaryColor.withOpacity(0.15),
        borderRadius: BorderRadius.circular(15),
      ),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: [
          Expanded(
            child: Text(
              review.feedback ?? '',
              style: TextStyle(
                color: Colors.black,
                fontSize: 14,
              ),
            ),
          ),
          SizedBox(width: 16),
          Column(
            children: [
              Icon(
```

```
                Icons.star,
                color: Colors.amber,
              ),
              Text(
                "${review.rating.toString()}",
                style: TextStyle(
                  fontWeight: FontWeight.bold,
                  color: Colors.black,
                ),
              ),
            ],
          ),
        ],
      ),
    );
  }
}
```

13.9. EXPAND_TEXT.DART

```
import 'package:flutter/material.dart';

class ExpandText extends ChangeNotifier {
  bool _isExpanded = false;
  bool get isExpanded {
    return _isExpanded;
  }

  set isExpanded(bool status) {
    _isExpanded = status;
    notifyListeners();
  }
}
```

13.10. PRODUCT_ACTIONS.DART

```
import 'package:flutter/material.dart';

class ProductActions extends ChangeNotifier {
  bool _productFavStatus = false;
```

```
  bool get productFavStatus {
    return _productFavStatus;
  }

  set initialProductFavStatus(bool status) {
    _productFavStatus = status;
  }

  set productFavStatus(bool status) {
    _productFavStatus = status;
    notifyListeners();
  }

  void switchProductFavStatus() {
    _productFavStatus = !_productFavStatus;
    notifyListeners();
  }
}
```

13.11. PRODUCT_IMG_SWIPER.DART

```
import 'package:flutter/material.dart';

class ProductImageSwiper extends ChangeNotifier {
  int _currentImageIndex = 0;
  int get currentImageIndex {
    return _currentImageIndex;
  }

  set currentImageIndex(int index) {
    _currentImageIndex = index;
    notifyListeners();
  }
}
```

14.CÁC LỚP MODEL

14.1.BASE MODEL

```
abstract class Model {
  final String id;

  Model(this.id);

  Map<String, dynamic> toMap();
  Map<String, dynamic> toUpdateMap();

  @override
  String toString() {
    return this.toMap().toString();
  }
}
```

14.2.ADDRESS

```
import 'package:flutter_online_shop/models/base_model.dart';
class Address extends Model {
  static const String TITLE_KEY = "title";
  static const String ADDRESS_LINE_1_KEY = "address_line_1";
  static const String ADDRESS_LINE_2_KEY = "address_line_2";
  static const String CITY_KEY = "city";
  static const String DISTRICT_KEY = "district";
  static const String STATE_KEY = "state";
  static const String LANDMARK_KEY = "landmark";
  static const String PINCODE_KEY = "pincode";
  static const String RECEIVER_KEY = "receiver";
  static const String PHONE_KEY = "phone";

  String title;
  String receiver;

  String addresLine1;
  String addressLine2;
  String city;
  String district;
  String state;
  String landmark;
  String pincode;
  String phone;
```

```
Address({
  String id,
  this.title,
  this.receiver,
  this.addresLine1,
  this.addressLine2,
  this.city,
  this.district,
  this.state,
  this.landmark,
  this.pincode,
  this.phone,
}) : super(id);

factory Address.fromMap(Map<String, dynamic> map, {String id}) {
  return Address(
    id: id,
    title: map[TITLE_KEY],
    receiver: map[RECEIVER_KEY],
    addresLine1: map[ADDRESS_LINE_1_KEY],
    addressLine2: map[ADDRESS_LINE_2_KEY],
    city: map[CITY_KEY],
    district: map[DISTRICT_KEY],
    state: map[STATE_KEY],
    landmark: map[LANDMARK_KEY],
    pincode: map[PINCODE_KEY],
    phone: map[PHONE_KEY],
  );
}

@override
Map<String, dynamic> toMap() {
  final map = <String, dynamic>{
    TITLE_KEY: title,
    RECEIVER_KEY: receiver,
    ADDRESS_LINE_1_KEY: addresLine1,
    ADDRESS_LINE_2_KEY: addressLine2,
    CITY_KEY: city,
    DISTRICT_KEY: district,
    STATE_KEY: state,
    LANDMARK_KEY: landmark,
    PINCODE_KEY: pincode,
    PHONE_KEY: phone,
  };

  return map;
}
```

```
  @override
  Map<String, dynamic> toUpdateMap() {
    final map = <String, dynamic>{};
    if (title != null) map[TITLE_KEY] = title;
    if (receiver != null) map[RECEIVER_KEY] = receiver;
    if (addresLine1 != null) map[ADDRESS_LINE_1_KEY] = addresLine1;
    if (addressLine2 != null) map[ADDRESS_LINE_2_KEY] = addressLine2;
    if (city != null) map[CITY_KEY] = city;
    if (district != null) map[DISTRICT_KEY] = district;
    if (state != null) map[STATE_KEY] = state;
    if (landmark != null) map[LANDMARK_KEY] = landmark;
    if (pincode != null) map[PINCODE_KEY] = pincode;
    if (phone != null) map[PHONE_KEY] = phone;
    return map;
  }
}
```

14.3.APP REVIEW

```
import 'package:flutter_online_shop/models/base_model.dart';
class AppReview extends Model {
  static const String LIKED_KEY = "liked";
  static const String FEEDBACK_KEY = "feedback";

  bool liked;
  String feedback;

  AppReview(
    String id, {
    this.liked,
    this.feedback,
  }) : super(id);

  factory AppReview.fromMap(Map<String, dynamic> map, {String id}) {
    return AppReview(
      id,
      liked: map[LIKED_KEY],
      feedback: map[FEEDBACK_KEY],
    );
  }

  @override
  Map<String, dynamic> toMap() {
    final map = <String, dynamic>{
      LIKED_KEY: liked,
      FEEDBACK_KEY: feedback,
    };
```

```
    return map;
  }

  @override
  Map<String, dynamic> toUpdateMap() {
    final map = <String, dynamic>{};
    if (liked != null) map[LIKED_KEY] = liked;
    if (feedback != null) map[FEEDBACK_KEY] = feedback;
    return map;
  }
}
```

14.4.CART

```
import 'package:flutter_online_shop/models/base_model.dart';
class CartItem extends Model {
  static const String PRODUCT_ID_KEY = "product_id";
  static const String ITEM_COUNT_KEY = "item_count";

  int itemCount;
  CartItem({
    String id,
    this.itemCount = 0,
  }) : super(id);

  factory CartItem.fromMap(Map<String, dynamic> map, {String id}) {
    return CartItem(
      id: id,
      itemCount: map[ITEM_COUNT_KEY],
    );
  }

  @override
  Map<String, dynamic> toMap() {
    final map = <String, dynamic>{
      ITEM_COUNT_KEY: itemCount,
    };
    return map;
  }

  @override
  Map<String, dynamic> toUpdateMap() {
    final map = <String, dynamic>{};
    if (itemCount != null) map[ITEM_COUNT_KEY] = itemCount;
    return map;
  }
}
```

14.5.ORDERED PRODUCT

```
import 'package:flutter_online_shop/models/base_model.dart';
class OrderedProduct extends Model {
  static const String PRODUCT_UID_KEY = "product_uid";
  static const String ORDER_DATE_KEY = "order_date";

  String productUid;
  String orderDate;
  OrderedProduct(
    String id, {
    this.productUid,
    this.orderDate,
  }) : super(id);

  factory OrderedProduct.fromMap(Map<String, dynamic> map, {String id}) {
    return OrderedProduct(
      id,
      productUid: map[PRODUCT_UID_KEY],
      orderDate: map[ORDER_DATE_KEY],
    );
  }

  @override
  Map<String, dynamic> toMap() {
    final map = <String, dynamic>{
      PRODUCT_UID_KEY: productUid,
      ORDER_DATE_KEY: orderDate,
    };
    return map;
  }

  @override
  Map<String, dynamic> toUpdateMap() {
    final map = <String, dynamic>{};
    if (productUid != null) map[PRODUCT_UID_KEY] = productUid;
    if (orderDate != null) map[ORDER_DATE_KEY] = orderDate;
    return map;
  }
}
```

14.6.PRODUCT

```
import 'package:enum_to_string/enum_to_string.dart';
import 'package:flutter_online_shop/models/base_model.dart';

enum ProductType {
  Electronics,
  Books,
  Fashion,
  Groceries,
  Furniture,
  Others,
}

class Product extends Model {
  static const String IMAGES_KEY = "images";
  static const String TITLE_KEY = "title";
  static const String VARIANT_KEY = "variant";
  static const String DISCOUNT_PRICE_KEY = "discount_price";
  static const String ORIGINAL_PRICE_KEY = "original_price";
  static const String RATING_KEY = "rating";
  static const String HIGHLIGHTS_KEY = "highlights";
  static const String DESCRIPTION_KEY = "description";
  static const String SELLER_KEY = "seller";
  static const String OWNER_KEY = "owner";
  static const String PRODUCT_TYPE_KEY = "product_type";
  static const String SEARCH_TAGS_KEY = "search_tags";

  List<String> images;
  String title;
  String variant;
  num discountPrice;
  num originalPrice;
  num rating;
  String highlights;
  String description;
  String seller;
  bool favourite;
  String owner;
  ProductType productType;
  List<String> searchTags;

  Product(
    String id, {
    this.images,
    this.title,
    this.variant,
```

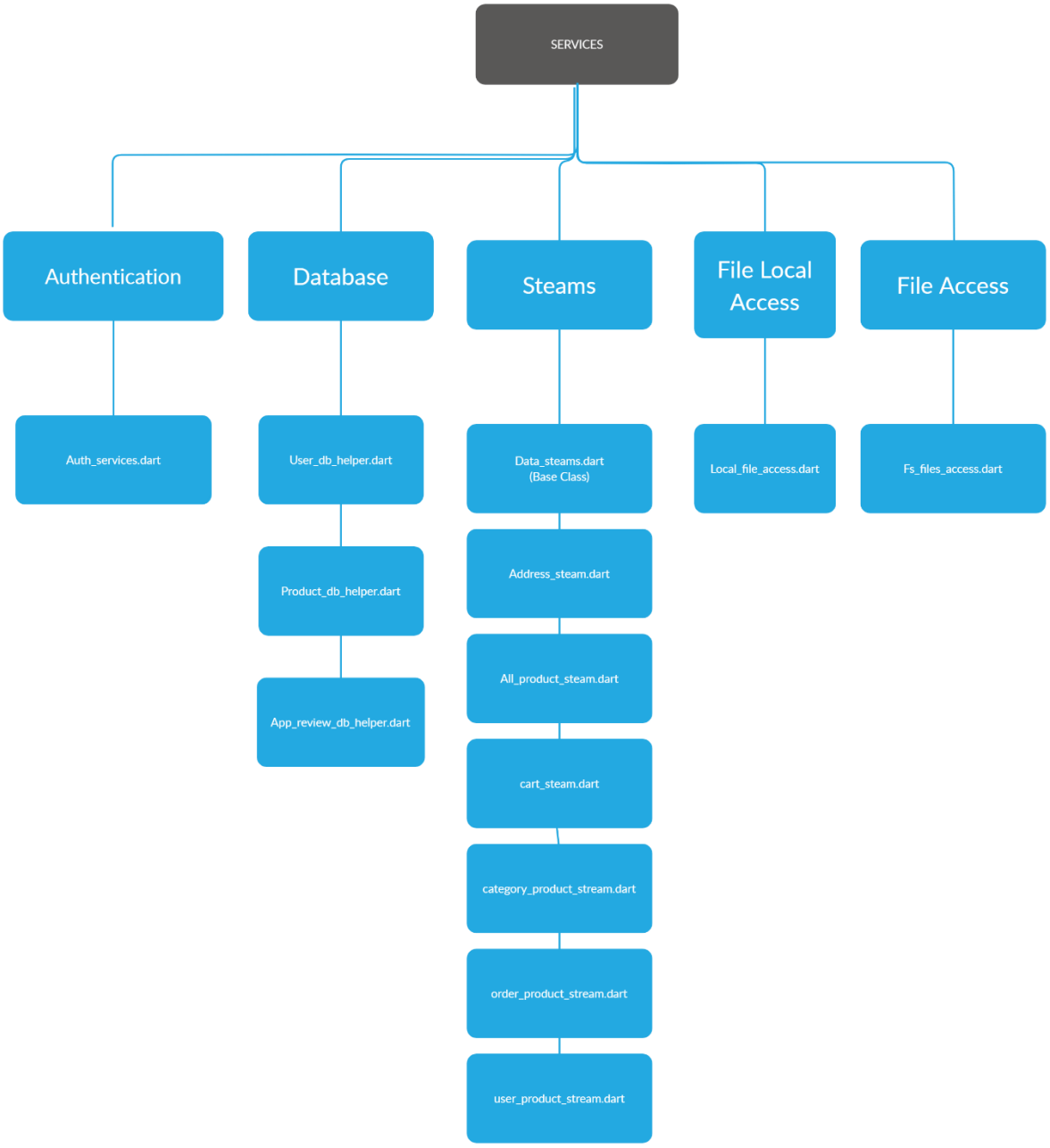
```
    this.productType,
    this.discountPrice,
    this.originalPrice,
    this.rating = 0.0,
    this.highlights,
    this.description,
    this.seller,
    this.owner,
    this.searchTags,
  }) : super(id);

  int calculatePercentageDiscount() {
    int discount =
        (((originalPrice - discountPrice) * 100) / originalPrice).round();
    return discount;
  }



factory Product.fromMap(Map<String, dynamic> map, {String id}) {
    if (map[SEARCH_TAGS_KEY] == null) {
      map[SEARCH_TAGS_KEY] = List<String>();
    }
    return Product(
      id,
      images: map[IMAGES_KEY].cast<String>(),
      title: map[TITLE_KEY],
      variant: map[VARIANT_KEY],
      productType:
          EnumToString.fromString(ProductType.values, map[PRODUCT_TYPE_KEY]),
      discountPrice: map[DISCOUNT_PRICE_KEY],
      originalPrice: map[ORIGINAL_PRICE_KEY],
      rating: map[RATING_KEY],
      highlights: map[HIGHLIGHTS_KEY],
      description: map[DESCRIPTION_KEY],
      seller: map[SELLER_KEY],
      owner: map[OWNER_KEY],
      searchTags: map[SEARCH_TAGS_KEY].cast<String>(),
    );
  }
```

```
@override
  Map<String, dynamic> toMap() {
    final map = <String, dynamic>{
      IMAGES_KEY: images,
      TITLE_KEY: title,
      VARIANT_KEY: variant,
      PRODUCT_TYPE_KEY: EnumToString.convertToString(productType),
      DISCOUNT_PRICE_KEY: discountPrice,
      ORIGINAL_PRICE_KEY: originalPrice,
      RATING_KEY: rating,
      HIGHLIGHTS_KEY: highlights,
      DESCRIPTION_KEY: description,
      SELLER_KEY: seller,
      OWNER_KEY: owner,
      SEARCH_TAGS_KEY: searchTags,
    };

    return map;
  }

  @override
  Map<String, dynamic> toUpdateMap() {
    final map = <String, dynamic>{};
    if (images != null) map[IMAGES_KEY] = images;
    if (title != null) map[TITLE_KEY] = title;
    if (variant != null) map[VARIANT_KEY] = variant;
    if (discountPrice != null) map[DISCOUNT_PRICE_KEY] = discountPrice;
    if (originalPrice != null) map[ORIGINAL_PRICE_KEY] = originalPrice;
    if (rating != null) map[RATING_KEY] = rating;
    if (highlights != null) map[HIGHLIGHTS_KEY] = highlights;
    if (description != null) map[DESCRIPTION_KEY] = description;
    if (seller != null) map[SELLER_KEY] = seller;
    if (productType != null)
      map[PRODUCT_TYPE_KEY] = EnumToString.convertToString(productType);
    if (owner != null) map[OWNER_KEY] = owner;
    if (searchTags != null) map[SEARCH_TAGS_KEY] = searchTags;

    return map;
  }
}
```

14.7.REVIEW

```
import 'package:flutter_online_shop/models/base_model.dart';

class Review extends Model {
  static const String REVIEWER_UID_KEY = "reviewer_uid";
  static const String RATING_KEY = "rating";
  static const String FEEDBACK_KEY = "review";

  String reviewerUid;
  int rating;
  String feedback;
  Review(
    String id, {
    this.reviewerUid,
    this.rating = 3,
    this.feedback,
  }) : super(id);

  factory Review.fromMap(Map<String, dynamic> map, {String id}) {
    return Review(
      id,
      reviewerUid: map[REVIEWER_UID_KEY],
      rating: map[RATING_KEY],
      feedback: map[FEEDBACK_KEY],
    );
  }

  @override
  Map<String, dynamic> toMap() {
    final map = <String, dynamic>{
      REVIEWER_UID_KEY: reviewerUid,
      RATING_KEY: rating,
      FEEDBACK_KEY: feedback,
    };
    return map;
  }

  @override
  Map<String, dynamic> toUpdateMap() {
    final map = <String, dynamic>{};
    if (reviewerUid != null) map[REVIEWER_UID_KEY] = reviewerUid;
    if (rating != null) map[RATING_KEY] = rating;
    if (feedback != null) map[FEEDBACK_KEY] = feedback;
    return map;
  }
}
```

15.CÁC LỚP SERVICES (LỚP XỬ LÝ)



15.1.AUTH_SERVICES (AUTHENTICATION)

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/exceptions/auth/actions_exception.dart';
import 'package:flutter_online_shop/exceptions/auth/firebase_exception.dart';
import 'package:flutter_online_shop/exceptions/auth/reauth_exception.dart';
import 'package:flutter_online_shop/exceptions/auth/sign_in_exception.dart';
import 'package:flutter_online_shop/exceptions/auth/sign_up_exception.dart';
import 'package:flutter_online_shop/services/db/user_db_helper.dart';

class AuthentificationService {
  //Khai báo các biến exception
  static const String USER_NOT_FOUND_EXCEPTION_CODE = "user-not-found";
  static const String WRONG_PASSWORD_EXCEPTION_CODE = "wrong-password";
  static const String EMAIL_ALREADY_IN_USE_EXCEPTION_CODE =
      "email-already-in-use";
  static const String OPERATION_NOT_ALLOWED_EXCEPTION_CODE =
      "operation-not-allowed";
  static const String WEAK_PASSWORD_EXCEPTION_CODE = "weak-password";
  static const String USER_MISMATCH_EXCEPTION_CODE = "user-mismatch";
  static const String INVALID_CREDENTIALS_EXCEPTION_CODE = "invalid-credential";
  static const String INVALID_EMAIL_EXCEPTION_CODE = "invalid-email";
  static const String USER_DISABLED_EXCEPTION_CODE = "user-disabled";
  static const String INVALID_VERIFICATION_CODE_EXCEPTION_CODE =
      "invalid-verification-code";
  static const String INVALID_VERIFICATION_ID_EXCEPTION_CODE =
      "invalid-verification-id";
  static const String REQUIRES_RECENT_LOGIN_EXCEPTION_CODE =
      "requires-recent-login";

  FirebaseAuth _firebaseAuth;

  AuthentificationService._privateConstructor();
  static AuthentificationService _instance =
      AuthentificationService._privateConstructor();

  FirebaseAuth get firebaseAuth {
    if (_firebaseAuth == null) {
      _firebaseAuth = FirebaseAuth.instance;
    }
    return _firebaseAuth;
  }

  factory AuthentificationService() {
    return _instance;
  }
```

```
Stream<User> get authStateChanges => firebaseAuth.authStateChanges();

Stream<User> get userChanges => firebaseAuth.userChanges();

Future<void> deleteUserAccount() async {
  await currentUser.delete();
  await signOut();
}

Future<bool> reauthCurrentUser(password) async {
  try {
    UserCredential userCredential =
        await firebaseAuth.signInWithEmailAndPassword(
            email: currentUser.email, password: password);
    userCredential = await currentUser
        .reauthenticateWithCredential(userCredential.credential);
  } on FirebaseAuthException catch (e) {
    if (e.code == WRONG_PASSWORD_EXCEPTION_CODE) {
      throw FirebaseSignInAuthWrongPasswordException();
    } else {
      throw FirebaseSignInAuthException(message: e.code);
    }
  } catch (e) {
    throw FirebaseReauthUnknownReasonFailureException(message: e.toString());
  }
  return true;
}

//Hàm đăng nhập dùng Email
Future<bool> signIn({String email, String password}) async {
  try {
    final UserCredential userCredential = await firebaseAuth
        .signInWithEmailAndPassword(email: email, password: password);
    if (userCredential.user.emailVerified) {
      return true;
    } else {
      await userCredential.user.sendEmailVerification();
      throw FirebaseSignInAuthUserNotVerifiedException();
    }
  } on MessagedFirebaseAuthException {
    rethrow;
  } on FirebaseAuthException catch (e) {
    switch (e.code) {
      case INVALID_EMAIL_EXCEPTION_CODE:
        throw FirebaseSignInAuthInvalidEmailException();

      case USER_DISABLED_EXCEPTION_CODE:
```

```
        throw FirebaseSignInAuthUserDisabledException();

      case USER_NOT_FOUND_EXCEPTION_CODE:
        throw FirebaseSignInAuthUserNotFoundException();

      case WRONG_PASSWORD_EXCEPTION_CODE:
        throw FirebaseSignInAuthWrongPasswordException();

      default:
        throw FirebaseSignInAuthException(message: e.code);
    }
  } catch (e) {
    rethrow;
  }
}

//Hàm đăng kí
Future<bool> signUp({String email, String password}) async {
  try {
    final UserCredential userCredential = await firebaseAuth
        .createUserWithEmailAndPassword(email: email, password: password);
    final String uid = userCredential.user.uid;
    if (userCredential.user.emailVerified == false) {
      await userCredential.user.sendEmailVerification();
    }
    await UserDatabaseHelper().createNewUser(uid);
    return true;
  } on MessagedFirebaseAuthException {
    rethrow;
  } on FirebaseAuthException catch (e) {
    switch (e.code) {
      case EMAIL_ALREADY_IN_USE_EXCEPTION_CODE:
        throw FirebaseSignUpAuthEmailAlreadyInUseException();
      case INVALID_EMAIL_EXCEPTION_CODE:
        throw FirebaseSignUpAuthInvalidEmailException();
      case OPERATION_NOT_ALLOWED_EXCEPTION_CODE:
        throw FirebaseSignUpAuthOperationNotAllowedException();
      case WEAK_PASSWORD_EXCEPTION_CODE:
        throw FirebaseSignUpAuthWeakPasswordException();
      default:
        throw FirebaseSignInAuthException(message: e.code);
    }
  } catch (e) {
    rethrow;
  }
}

//Đăng xuất
```

```
Future<void> signOut() async {
  await firebaseAuth.signOut();
}

//Kiểm tra nguòi dùng đã xác minh
bool get currentUserVerified {
  currentUser.reload();
  return currentUser.emailVerified;
}

//Gửi mã xác minh
Future<void> sendVerificationEmailToCurrentUser() async {
  await firebaseAuth.currentUser.sendEmailVerification();
}

//Lấy information của User hiện tại
User get currentUser {
  return firebaseAuth.currentUser;
}

Future<void> updateCurrentUserDisplayName(String updatedDisplayName) async {
  await currentUser.updateProfile(displayName: updatedDisplayName);
}

//Reset mật khẩu
Future<bool> resetPasswordForEmail(String email) async {
  try {
    await firebaseAuth.sendPasswordResetEmail(email: email);
    return true;
  } on MessagedFirebaseAuthException {
    rethrow;
  } on FirebaseAuthException catch (e) {
    if (e.code == USER_NOT_FOUND_EXCEPTION_CODE) {
      throw FirebaseCredentialActionAuthUserNotFoundException();
    } else {
      throw FirebaseCredentialActionAuthException(message: e.code);
    }
  } catch (e) {
    rethrow;
  }
}

//Đổi password
Future<bool> changePasswordForCurrentUser(
    {String oldPassword, @required String newPassword}) async {
  try {
    bool isOldPasswordProvidedCorrect = true;
    if (oldPassword != null) {
```

```
      isOldPasswordProvidedCorrect =
          await verifyCurrentUserPassword(oldPassword);
    }
    if (isOldPasswordProvidedCorrect) {
      await firebaseAuth.currentUser.updatePassword(newPassword);

      return true;
    } else {
      throw FirebaseReauthWrongPasswordException();
    }
  } on MessagedFirebaseAuthException {
    rethrow;
  } on FirebaseAuthException catch (e) {
    switch (e.code) {
      case WEAK_PASSWORD_EXCEPTION_CODE:
        throw FirebaseCredentialActionAuthWeakPasswordException();
      case REQUIRES_RECENT_LOGIN_EXCEPTION_CODE:
        throw FirebaseCredentialActionAuthRequiresRecentLoginException();
      default:
        throw FirebaseCredentialActionAuthException(message: e.code);
    }
  } catch (e) {
    rethrow;
  }
}

//Đổi email
Future<bool> changeEmailForCurrentUser(
    {String password, String newEmail}) async {
  try {
    bool isPasswordProvidedCorrect = true;
    if (password != null) {
      isPasswordProvidedCorrect = await verifyCurrentUserPassword(password);
    }
    if (isPasswordProvidedCorrect) {
      await currentUser.verifyBeforeUpdateEmail(newEmail);

      return true;
    } else {
      throw FirebaseReauthWrongPasswordException();
    }
  } on MessagedFirebaseAuthException {
    rethrow;
  } on FirebaseAuthException catch (e) {
    throw FirebaseCredentialActionAuthException(message: e.code);
  } catch (e) {
    rethrow;
  }
```

```
  }

  //Kiểm tra mật khẩu đã được xác minh
  Future<bool> verifyCurrentUserPassword(String password) async {
    try {
      final AuthCredential authCredential = EmailAuthProvider.credential(
        email: currentUser.email,
        password: password,
      );

      final authCredentials =
          await currentUser.reauthenticateWithCredential(authCredential);
      return authCredentials != null;
    } on MessagedFirebaseAuthException {
      rethrow;
    } on FirebaseAuthException catch (e) {
      switch (e.code) {
        case USER_MISMATCH_EXCEPTION_CODE:
          throw FirebaseReauthUserMismatchException();
        case USER_NOT_FOUND_EXCEPTION_CODE:
          throw FirebaseReauthUserNotFoundException();
        case INVALID_CREDENTIALS_EXCEPTION_CODE:
          throw FirebaseReauthInvalidCredentialException();
        case INVALID_EMAIL_EXCEPTION_CODE:
          throw FirebaseReauthInvalidEmailException();
        case WRONG_PASSWORD_EXCEPTION_CODE:
          throw FirebaseReauthWrongPasswordException();
        case INVALID_VERIFICATION_CODE_EXCEPTION_CODE:
          throw FirebaseReauthInvalidVerificationCodeException();
        case INVALID_VERIFICATION_ID_EXCEPTION_CODE:
          throw FirebaseReauthInvalidVerificationIdException();
        default:
          throw FirebaseReauthException(message: e.code);
      }
    } catch (e) {
      rethrow;
    }
  }
}
```

15.2.USER_DB_HELPER (DATABASE)

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter_online_shop/models/address_model.dart';
import 'package:flutter_online_shop/models/cart_model.dart';
```

```dart
import 'package:flutter_online_shop/models/order_product_model.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';

class UserDatabaseHelper {
  static const String USERS_COLLECTION_NAME = "users";
  static const String ADDRESSES_COLLECTION_NAME = "addresses";
  static const String CART_COLLECTION_NAME = "cart";
  static const String ORDERED_PRODUCTS_COLLECTION_NAME = "ordered_products";

  static const String PHONE_KEY = 'phone';
  static const String DP_KEY = "display_picture";
  static const String FAV_PRODUCTS_KEY = "favourite_products";

  UserDatabaseHelper._privateConstructor();
  static UserDatabaseHelper _instance =
      UserDatabaseHelper._privateConstructor();
  factory UserDatabaseHelper() {
    return _instance;
  }
  FirebaseFirestore _firebaseFirestore;
  FirebaseFirestore get firestore {
    if (_firebaseFirestore == null) {
      _firebaseFirestore = FirebaseFirestore.instance;
    }
    return _firebaseFirestore;
  }

  //Tạo người dùng mới
  Future<void> createNewUser(String uid) async {
    await firestore.collection(USERS_COLLECTION_NAME).doc(uid).set({
      DP_KEY: null,
      PHONE_KEY: null,
      FAV_PRODUCTS_KEY: List<String>(),
    });
  }

  //Xoá dữ liệu người dùng
  Future<void> deleteCurrentUserData() async {
    final uid = AuthentificationService().currentUser.uid;
    final docRef = firestore.collection(USERS_COLLECTION_NAME).doc(uid);
    final cartCollectionRef = docRef.collection(CART_COLLECTION_NAME);
    final addressCollectionRef = docRef.collection(ADDRESSES_COLLECTION_NAME);
    final ordersCollectionRef =
        docRef.collection(ORDERED_PRODUCTS_COLLECTION_NAME);

    final cartDocs = await cartCollectionRef.get();
    for (final cartDoc in cartDocs.docs) {
```

```
    await cartCollectionRef.doc(cartDoc.id).delete();
  }
  final addressesDocs = await addressCollectionRef.get();
  for (final addressDoc in addressesDocs.docs) {
    await addressCollectionRef.doc(addressDoc.id).delete();
  }
  final ordersDoc = await ordersCollectionRef.get();
  for (final orderDoc in ordersDoc.docs) {
    await ordersCollectionRef.doc(orderDoc.id).delete();
  }

  await docRef.delete();
}

//Sản phẩm yêu thích
Future<bool> isProductFavourite(String productId) async {
  String uid = AuthentificationService().currentUser.uid;
  final userDocSnapshot =
      firestore.collection(USERS_COLLECTION_NAME).doc(uid);
  final userDocData = (await userDocSnapshot.get()).data();
  final favList = userDocData[FAV_PRODUCTS_KEY].cast<String>();
  if (favList.contains(productId)) {
    return true;
  } else {
    return false;
  }
}

//lấy sản phẩm yêu thích đưa vào list
Future<List> get usersFavouriteProductsList async {
  String uid = AuthentificationService().currentUser.uid;
  final userDocSnapshot =
      firestore.collection(USERS_COLLECTION_NAME).doc(uid);
  final userDocData = (await userDocSnapshot.get()).data();
  final favList = userDocData[FAV_PRODUCTS_KEY];
  return favList;
}

Future<bool> switchProductFavouriteStatus(
    String productId, bool newState) async {
  String uid = AuthentificationService().currentUser.uid;
  final userDocSnapshot =
      firestore.collection(USERS_COLLECTION_NAME).doc(uid);

  if (newState == true) {
    userDocSnapshot.update({
      FAV_PRODUCTS_KEY: FieldValue.arrayUnion([productId])
    });
```

```
  } else {
    userDocSnapshot.update({
      FAV_PRODUCTS_KEY: FieldValue.arrayRemove([productId])
    });
  }
  return true;
}

//Lấy list địa chỉ giao hàng
Future<List<String>> get addressesList async {
  String uid = AuthentificationService().currentUser.uid;
  final snapshot = await firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(ADDRESSES_COLLECTION_NAME)
      .get();
  final addresses = List<String>();
  snapshot.docs.forEach((doc) {
    addresses.add(doc.id);
  });

  return addresses;
}

//Lấy Id địa chỉ giao hàng
Future<Address> getAddressFromId(String id) async {
  String uid = AuthentificationService().currentUser.uid;
  final doc = await firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(ADDRESSES_COLLECTION_NAME)
      .doc(id)
      .get();
  final address = Address.fromMap(doc.data(), id: doc.id);
  return address;
}

//THêm địa chỉ giao hàng
Future<bool> addAddressForCurrentUser(Address address) async {
  String uid = AuthentificationService().currentUser.uid;
  final addressesCollectionReference = firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(ADDRESSES_COLLECTION_NAME);
  await addressesCollectionReference.add(address.toMap());
  return true;
}
```

```
//Xoá địa chỉ giao hàng
Future<bool> deleteAddressForCurrentUser(String id) async {
  String uid = AuthentificationService().currentUser.uid;
  final addressDocReference = firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(ADDRESSES_COLLECTION_NAME)
      .doc(id);
  await addressDocReference.delete();
  return true;
}

//Sửa địa chỉ giao hàng
Future<bool> updateAddressForCurrentUser(Address address) async {
  String uid = AuthentificationService().currentUser.uid;
  final addressDocReference = firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(ADDRESSES_COLLECTION_NAME)
      .doc(address.id);
  await addressDocReference.update(address.toMap());
  return true;
}

//Lấy dữ liệu giỏ hàng bằng ID
Future<CartItem> getCartItemFromId(String id) async {
  String uid = AuthentificationService().currentUser.uid;
  final cartCollectionRef = firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(CART_COLLECTION_NAME);
  final docRef = cartCollectionRef.doc(id);
  final docSnapshot = await docRef.get();
  final cartItem = CartItem.fromMap(docSnapshot.data(), id: docSnapshot.id);
  return cartItem;
}

//Thêm vào giỏ hàng
Future<bool> addProductToCart(String productId) async {
  String uid = AuthentificationService().currentUser.uid;
  final cartCollectionRef = firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(CART_COLLECTION_NAME);
  final docRef = cartCollectionRef.doc(productId);
  final docSnapshot = await docRef.get();
  bool alreadyPresent = docSnapshot.exists;
  if (alreadyPresent == false) {
```

```
    docRef.set(CartItem(itemCount: 1).toMap());
  } else {
    docRef.update({CartItem.ITEM_COUNT_KEY: FieldValue.increment(1)});
  }
  return true;
}

//Giỏ hàng rỗng
Future<List<String>> emptyCart() async {
  String uid = AuthentificationService().currentUser.uid;
  final cartItems = await firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(CART_COLLECTION_NAME)
      .get();
  List orderedProductsUid = List<String>();
  for (final doc in cartItems.docs) {
    orderedProductsUid.add(doc.id);
    await doc.reference.delete();
  }
  return orderedProductsUid;
}

//Tổng hàng trong giỏ hàng
Future<num> get cartTotal async {
  String uid = AuthentificationService().currentUser.uid;
  final cartItems = await firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(CART_COLLECTION_NAME)
      .get();
  num total = 0.0;
  for (final doc in cartItems.docs) {
    num itemsCount = doc.data()[CartItem.ITEM_COUNT_KEY];
    final product = await ProductDatabaseHelper().getProductWithID(doc.id);
    total += (itemsCount * product.discountPrice);
  }
  return total;
}

//Xoá khỏi giỏ hàng
Future<bool> removeProductFromCart(String cartItemID) async {
  String uid = AuthentificationService().currentUser.uid;
  final cartCollectionReference = firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(CART_COLLECTION_NAME);
  await cartCollectionReference.doc(cartItemID).delete();
```

```
  return true;
}

//Tăng số lượng từng món hàng
Future<bool> increaseCartItemCount(String cartItemID) async {
  String uid = AuthentificationService().currentUser.uid;
  final cartCollectionRef = firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(CART_COLLECTION_NAME);
  final docRef = cartCollectionRef.doc(cartItemID);
  docRef.update({CartItem.ITEM_COUNT_KEY: FieldValue.increment(1)});
  return true;
}

//Giảm số lượng từng món hàng
Future<bool> decreaseCartItemCount(String cartItemID) async {
  String uid = AuthentificationService().currentUser.uid;
  final cartCollectionRef = firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(CART_COLLECTION_NAME);
  final docRef = cartCollectionRef.doc(cartItemID);
  final docSnapshot = await docRef.get();
  int currentCount = docSnapshot.data()[CartItem.ITEM_COUNT_KEY];
  if (currentCount <= 1) {
    return removeProductFromCart(cartItemID);
  } else {
    docRef.update({CartItem.ITEM_COUNT_KEY: FieldValue.increment(-1)});
  }
  return true;
}

//Lấy list giỏ hàng
Future<List<String>> get allCartItemsList async {
  String uid = AuthentificationService().currentUser.uid;
  final querySnapshot = await firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(CART_COLLECTION_NAME)
      .get();
  List itemsId = List<String>();
  for (final item in querySnapshot.docs) {
    itemsId.add(item.id);
  }
  return itemsId;
}
```

```
//Lấy list sản phẩm đã đặt hàng
Future<List<String>> get orderedProductsList async {
  String uid = AuthentificationService().currentUser.uid;
  final orderedProductsSnapshot = await firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(ORDERED_PRODUCTS_COLLECTION_NAME)
      .get();
  List orderedProductsId = List<String>();
  for (final doc in orderedProductsSnapshot.docs) {
    orderedProductsId.add(doc.id);
  }
  return orderedProductsId;
}

//Thêm vào đơn đặt hàng
Future<bool> addToMyOrders(List<OrderedProduct> orders) async {
  String uid = AuthentificationService().currentUser.uid;
  final orderedProductsCollectionRef = firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(ORDERED_PRODUCTS_COLLECTION_NAME);
  for (final order in orders) {
    await orderedProductsCollectionRef.add(order.toMap());
  }
  return true;
}

//Lấy dữ liệu đơn đặt hàng bằng id
Future<OrderedProduct> getOrderedProductFromId(String id) async {
  String uid = AuthentificationService().currentUser.uid;
  final doc = await firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .collection(ORDERED_PRODUCTS_COLLECTION_NAME)
      .doc(id)
      .get();
  final orderedProduct = OrderedProduct.fromMap(doc.data(), id: doc.id);
  return orderedProduct;
}

Stream<DocumentSnapshot> get currentUserDataStream {
  String uid = AuthentificationService().currentUser.uid;
  return firestore
      .collection(USERS_COLLECTION_NAME)
      .doc(uid)
      .get()
      .asStream();
```

```
  }

  Future<bool> updatePhoneForCurrentUser(String phone) async {
    String uid = AuthentificationService().currentUser.uid;
    final userDocSnapshot =
        firestore.collection(USERS_COLLECTION_NAME).doc(uid);
    await userDocSnapshot.update({PHONE_KEY: phone});
    return true;
  }

  String getPathForCurrentUserDisplayPicture() {
    final String currentUserUid = AuthentificationService().currentUser.uid;
    return "user/display_picture/$currentUserUid";
  }

  Future<bool> uploadDisplayPictureForCurrentUser(String url) async {
    String uid = AuthentificationService().currentUser.uid;
    final userDocSnapshot =
        firestore.collection(USERS_COLLECTION_NAME).doc(uid);
    await userDocSnapshot.update(
      {DP_KEY: url},
    );
    return true;
  }

  Future<bool> removeDisplayPictureForCurrentUser() async {
    String uid = AuthentificationService().currentUser.uid;
    final userDocSnapshot =
        firestore.collection(USERS_COLLECTION_NAME).doc(uid);
    await userDocSnapshot.update(
      {
        DP_KEY: FieldValue.delete(),
      },
    );
    return true;
  }

  Future<String> get displayPictureForCurrentUser async {
    String uid = AuthentificationService().currentUser.uid;
    final userDocSnapshot =
        await firestore.collection(USERS_COLLECTION_NAME).doc(uid).get();
    return userDocSnapshot.data()[DP_KEY];
  }
}
```

15.3.PRODUCT_DB_HELPER (DATABASE)

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:enum_to_string/enum_to_string.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/models/review_model.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';

class ProductDatabaseHelper {
  static const String PRODUCTS_COLLECTION_NAME = "products";
  static const String REVIEWS_COLLECTION_NAME = "reviews";

  ProductDatabaseHelper._privateConstructor();
  static ProductDatabaseHelper _instance =
      ProductDatabaseHelper._privateConstructor();
  factory ProductDatabaseHelper() {
    return _instance;
  }
  FirebaseFirestore _firebaseFirestore;
  FirebaseFirestore get firestore {
    if (_firebaseFirestore == null) {
      _firebaseFirestore = FirebaseFirestore.instance;
    }
    return _firebaseFirestore;
  }

  //tìm sản phẩm
  Future<List<String>> searchInProducts(String query,
      {ProductType productType}) async {
    Query queryRef;
    if (productType == null) {
      queryRef = firestore.collection(PRODUCTS_COLLECTION_NAME);
    } else {
      final productTypeStr = EnumToString.convertToString(productType);
      print(productTypeStr);
      queryRef = firestore
          .collection(PRODUCTS_COLLECTION_NAME)
          .where(Product.PRODUCT_TYPE_KEY, isEqualTo: productTypeStr);
    }

    Set productsId = Set<String>();
    final querySearchInTags = await queryRef
        .where(Product.SEARCH_TAGS_KEY, arrayContains: query)
        .get();
    for (final doc in querySearchInTags.docs) {
      productsId.add(doc.id);
    }
```

```
  final queryRefDocs = await queryRef.get();
  for (final doc in queryRefDocs.docs) {
    final product = Product.fromMap(doc.data(), id: doc.id);
    if (product.title.toString().toLowerCase().contains(query) ||
        product.description.toString().toLowerCase().contains(query) ||
        product.highlights.toString().toLowerCase().contains(query) ||
        product.variant.toString().toLowerCase().contains(query) ||
        product.seller.toString().toLowerCase().contains(query)) {
      productsId.add(product.id);
    }
  }
  return productsId.toList();
}

//Thêm đánh giá về sản phẩm (Văn bản)
Future<bool> addProductReview(String productId, Review review) async {
  final reviewesCollectionRef = firestore
      .collection(PRODUCTS_COLLECTION_NAME)
      .doc(productId)
      .collection(REVIEWS_COLLECTION_NAME);
  final reviewDoc = reviewesCollectionRef.doc(review.reviewerUid);
  if ((await reviewDoc.get()).exists == false) {
    reviewDoc.set(review.toMap());
    return await addUsersRatingForProduct(
      productId,
      review.rating,
    );
  } else {
    int oldRating = 0;
    oldRating = (await reviewDoc.get()).data()[Product.RATING_KEY];
    reviewDoc.update(review.toUpdateMap());
    return await addUsersRatingForProduct(productId, review.rating,
        oldRating: oldRating);
  }
}

//Thêm đánh giá về sản phẩm(Số sao)
Future<bool> addUsersRatingForProduct(String productId, int rating,
    {int oldRating}) async {
  final productDocRef =
      firestore.collection(PRODUCTS_COLLECTION_NAME).doc(productId);
  final ratingsCount =
      (await productDocRef.collection(REVIEWS_COLLECTION_NAME).get())
          .docs
          .length;
  final productDoc = await productDocRef.get();
  final prevRating = productDoc.data()[Review.RATING_KEY];
  double newRating;
```

```
  if (oldRating == null) {
    newRating = (prevRating * (ratingsCount - 1) + rating) / ratingsCount;
  } else {
    newRating =
        (prevRating * (ratingsCount) + rating - oldRating) / ratingsCount;
  }
  final newRatingRounded = double.parse(newRating.toStringAsFixed(1));
  await productDocRef.update({Product.RATING_KEY: newRatingRounded});
  return true;
}

//Lấy ID đánh giá của sản phẩm
Future<Review> getProductReviewWithID(
    String productId, String reviewId) async {
  final reviewesCollectionRef = firestore
      .collection(PRODUCTS_COLLECTION_NAME)
      .doc(productId)
      .collection(REVIEWS_COLLECTION_NAME);
  final reviewDoc = await reviewesCollectionRef.doc(reviewId).get();
  if (reviewDoc.exists) {
    return Review.fromMap(reviewDoc.data(), id: reviewDoc.id);
  }
  return null;
}

//Lấy tất cả giá trị đánh giá
Stream<List<Review>> getAllReviewsStreamForProductId(
    String productId) async* {
  final reviewesQuerySnapshot = firestore
      .collection(PRODUCTS_COLLECTION_NAME)
      .doc(productId)
      .collection(REVIEWS_COLLECTION_NAME)
      .get()
      .asStream();
  await for (final querySnapshot in reviewesQuerySnapshot) {
    List<Review> reviews = List<Review>();
    for (final reviewDoc in querySnapshot.docs) {
      Review review = Review.fromMap(reviewDoc.data(), id: reviewDoc.id);
      reviews.add(review);
    }
    yield reviews;
  }
}

//Lấy thôgn tin sản phẩm dựa vào ID
Future<Product> getProductWithID(String productId) async {
  final docSnapshot = await firestore
      .collection(PRODUCTS_COLLECTION_NAME)
```

```
      .doc(productId)
      .get();

  if (docSnapshot.exists) {
    return Product.fromMap(docSnapshot.data(), id: docSnapshot.id);
  }
  return null;
}

//Lấy thông tin sản phẩm
Future<String> addUsersProduct(Product product) async {
  String uid = AuthentificationService().currentUser.uid;
  final productMap = product.toMap();
  product.owner = uid;
  final productsCollectionReference =
      firestore.collection(PRODUCTS_COLLECTION_NAME);
  final docRef = await productsCollectionReference.add(product.toMap());
  await docRef.update({
    Product.SEARCH_TAGS_KEY: FieldValue.arrayUnion(
        [productMap[Product.PRODUCT_TYPE_KEY].toString().toLowerCase()])
  });
  return docRef.id;
}

//Xoá sản phẩm
Future<bool> deleteUserProduct(String productId) async {
  final productsCollectionReference =
      firestore.collection(PRODUCTS_COLLECTION_NAME);
  await productsCollectionReference.doc(productId).delete();
  return true;
}

//Sửa sản phẩm
Future<String> updateUsersProduct(Product product) async {
  final productMap = product.toUpdateMap();
  final productsCollectionReference =
      firestore.collection(PRODUCTS_COLLECTION_NAME);
  final docRef = productsCollectionReference.doc(product.id);
  await docRef.update(productMap);
  if (product.productType != null) {
    await docRef.update({
      Product.SEARCH_TAGS_KEY: FieldValue.arrayUnion(
          [productMap[Product.PRODUCT_TYPE_KEY].toString().toLowerCase()])
    });
  }
  return docRef.id;
}
```

```
//Lấy dữ liệu danh mục sản phẩm
Future<List<String>> getCategoryProductsList(ProductType productType) async {
  final productsCollectionReference =
      firestore.collection(PRODUCTS_COLLECTION_NAME);
  final queryResult = await productsCollectionReference
      .where(Product.PRODUCT_TYPE_KEY,
          isEqualTo: EnumToString.convertToString(productType))
      .get();
  List productsId = List<String>();
  for (final product in queryResult.docs) {
    final id = product.id;
    productsId.add(id);
  }
  return productsId;
}

Future<List<String>> get usersProductsList async {
  String uid = AuthentificationService().currentUser.uid;
  final productsCollectionReference =
      firestore.collection(PRODUCTS_COLLECTION_NAME);
  final querySnapshot = await productsCollectionReference
      .where(Product.OWNER_KEY, isEqualTo: uid)
      .get();
  List usersProducts = List<String>();
  querySnapshot.docs.forEach((doc) {
    usersProducts.add(doc.id);
  });
  return usersProducts;
}

Future<List<String>> get allProductsList async {
  final products = await firestore.collection(PRODUCTS_COLLECTION_NAME).get();
  List productsId = List<String>();
  for (final product in products.docs) {
    final id = product.id;
    productsId.add(id);
  }
  return productsId;
}

//Sửa hình ảnh sản phẩm
Future<bool> updateProductsImages(
    String productId, List<String> imgUrl) async {
  final Product updateProduct = Product(null, images: imgUrl);
  final docRef =
      firestore.collection(PRODUCTS_COLLECTION_NAME).doc(productId);
  await docRef.update(updateProduct.toUpdateMap());
  return true;
```

```
  }

  //Lấy đường dẫn hình ảnh
  String getPathForProductImage(String id, int index) {
    String path = "products/images/$id";
    return path + "_$index";
  }
}
```

15.4.APPREVIEW_DB_HELPER (DATABASE)

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter_online_shop/models/appReview_model.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';

class AppReviewDatabaseHelper {
  static const String APP_REVIEW_COLLECTION_NAME = "app_reviews";

  AppReviewDatabaseHelper._privateConstructor();
  static AppReviewDatabaseHelper _instance =
      AppReviewDatabaseHelper._privateConstructor();
  factory AppReviewDatabaseHelper() {
    return _instance;
  }
  FirebaseFirestore _firebaseFirestore;
  FirebaseFirestore get firestore {
    if (_firebaseFirestore == null) {
      _firebaseFirestore = FirebaseFirestore.instance;
    }
    return _firebaseFirestore;
  }

  //Hàm edit đánh giá
  Future<bool> editAppReview(AppReview appReview) async {
    final uid = AuthentificationService().currentUser.uid;
    final docRef = firestore.collection(APP_REVIEW_COLLECTION_NAME).doc(uid);
    final docData = await docRef.get();
    if (docData.exists) {
      docRef.update(appReview.toUpdateMap());
    } else {
      docRef.set(appReview.toMap());
    }
    return true;
  }

  //Hàm lấy giá trị đánh giá của user hiện tại
  Future<AppReview> getAppReviewOfCurrentUser() async {
```

```
    final uid = AuthentificationService().currentUser.uid;
    final docRef = firestore.collection(APP_REVIEW_COLLECTION_NAME).doc(uid);
    final docData = await docRef.get();
    if (docData.exists) {
      final appReview = AppReview.fromMap(docData.data(), id: docData.id);
      return appReview;
    } else {
      final appReview = AppReview(uid, liked: true, feedback: "");
      docRef.set(appReview.toMap());
      return appReview;
    }
  }
}
```

15.5.FS_FILES_ACCESS (FILE ACCESS)

```
import 'dart:io';
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_storage/firebase_storage.dart';

class FirestoreFilesAccess {
  FirestoreFilesAccess._privateConstructor();
  static FirestoreFilesAccess _instance =
      FirestoreFilesAccess._privateConstructor();
  factory FirestoreFilesAccess() {
    return _instance;
  }
  FirebaseFirestore _firebaseFirestore;
  FirebaseFirestore get firestore {
    if (_firebaseFirestore == null) {
      _firebaseFirestore = FirebaseFirestore.instance;
    }
    return _firebaseFirestore;
  }

  Future<String> uploadFileToPath(File file, String path) async {
    final Reference firestorageRef = FirebaseStorage.instance.ref();
    final snapshot = await firestorageRef.child(path).putFile(file);
    final downloadUrl = await snapshot.ref.getDownloadURL();
    return downloadUrl;
  }

  Future<bool> deleteFileFromPath(String path) async {
    final Reference firestorageRef = FirebaseStorage.instance.ref();
    await firestorageRef.child(path).delete();
    return true;
  }
```

```
  Future<String> getDeveloperImage() async {
    const filename = "about_developer/developer";
    List<String> extensions = <String>["jpg", "jpeg", "jpe", "jfif"];
    final Reference firestorageRef = FirebaseStorage.instance.ref();
    for (final ext in extensions) {
      try {
        final url =
            await firestorageRef.child("$filename.$ext").getDownloadURL();
        return url;
      } catch (_) {
        continue;
      }
    }
    throw FirebaseException(
        message: "No JPEG Image found for Developer",
        plugin: 'Firebase Storage');
  }
}
```

15.6.LOCAL_FILES_ACCESS (LOCAL FILE ACCESS)

```
import 'dart:io';
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/exceptions/local_file/image_picking_exceptions.dart';
import 'package:flutter_online_shop/exceptions/local_file/local_files_exceptions.dart';
import 'package:image_picker/image_picker.dart';
import 'package:permission_handler/permission_handler.dart';

Future<String> choseImageFromLocalFiles(
  BuildContext context, {
  int maxSizeInKB = 1024,
  int minSizeInKB = 5,
}) async {
  final PermissionStatus photoPermissionStatus =
      await Permission.photos.request();
  if (!photoPermissionStatus.isGranted) {
    throw LocalFileHandlingStorageReadPermissionDeniedException(
        message: "Bạn cần cho phép sử dụng quyền truy cập bộ nhớ");
  }

  final imgPicker = ImagePicker();
  final imgSource = await showDialog(
    builder: (context) {
      return AlertDialog(
        title: Text("Chọn hình ảnh"),
        actions: [
```

```
          FlatButton(
            child: Text("Camera"),
            onPressed: () {
              Navigator.pop(context, ImageSource.camera);
            },
          ),
          FlatButton(
            child: Text("Bộ sưu tập"),
            onPressed: () {
              Navigator.pop(context, ImageSource.gallery);
            },
          ),
        ],
      );
    },
    context: context,
  );
  if (imgSource == null)
    throw LocalImagePickingInvalidImageException(message: "Chưa chọn hình ảnh");
  final PickedFile imagePicked = await imgPicker.getImage(source: imgSource);
  if (imagePicked == null) {
    throw LocalImagePickingInvalidImageException();
  } else {
    final fileLength = await File(imagePicked.path).length();
    if (fileLength > (maxSizeInKB * 1024) ||
        fileLength < (minSizeInKB * 1024)) {
      throw LocalImagePickingFileSizeOutOfBoundsException(
          message: "Kích thước ảnh không được vượt quá 1MB");
    } else {
      return imagePicked.path;
    }
  }
}
```

15.7.DATA_STEAMS (STEAMS)

```
import 'dart:async';
abstract class DataStream<T> {
  StreamController<T> streamController;
  void init() {
    streamController = StreamController();
    reload();
  }
  Stream get stream => streamController.stream;
  void addData(T data) {
    streamController.sink.add(data);
  }
```

```
  void addError(dynamic e) {
    streamController.sink.addError(e);
  }
  void reload();
  void dispose() {
    streamController.close();
  }
}
```

15.8.ADDRESS_STEAM (STEAMS)

```
import 'package:flutter_online_shop/services/db/user_db_helper.dart';
import 'package:flutter_online_shop/services/steams/data_stream.dart';

class AddressesStream extends DataStream<List<String>> {
  @override
  void reload() {
    final addressesList = UserDatabaseHelper().addressesList;
    addressesList.then((list) {
      addData(list);
    }).catchError((e) {
      addError(e);
    });
  }
}
```

15.9.ALL_PRODUCT_STEAM (STEAMS)

```
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/services/steams/data_stream.dart';

class AllProductsStream extends DataStream<List<String>> {
  @override
  void reload() {
    final allProductsFuture = ProductDatabaseHelper().allProductsList;
    allProductsFuture.then((favProducts) {
      addData(favProducts);
    }).catchError((e) {
      addError(e);
    });
  }
}
```

15.10.CART_STEAM (STEAMS)

```
import 'package:flutter_online_shop/services/db/user_db_helper.dart';
import 'package:flutter_online_shop/services/steams/data_stream.dart';

class CartItemsStream extends DataStream<List<String>> {
  @override
  void reload() {
    final allProductsFuture = UserDatabaseHelper().allCartItemsList;
    allProductsFuture.then((favProducts) {
      addData(favProducts);
    }).catchError((e) {
      addError(e);
    });
  }
}
```

15.11.CATEGORY_PRODUCT_STEAM (STEAMS)

```
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/services/steams/data_stream.dart';

class CategoryProductsStream extends DataStream<List<String>> {
  final ProductType category;

  CategoryProductsStream(this.category);
  @override
  void reload() {
    final allProductsFuture =
        ProductDatabaseHelper().getCategoryProductsList(category);
    allProductsFuture.then((favProducts) {
      addData(favProducts);
    }).catchError((e) {
      addError(e);
    });
  }
}
```

15.12.ORDER_PRODUCT_STEAM (STEAMS)

```
import 'package:flutter_online_shop/services/db/user_db_helper.dart';
import 'package:flutter_online_shop/services/steams/data_stream.dart';

class OrderedProductsStream extends DataStream<List<String>> {
  @override
  void reload() {
    final orderedProductsFuture = UserDatabaseHelper().orderedProductsList;
    orderedProductsFuture.then((data) {
      addData(data);
    }).catchError((e) {
      addError(e);
    });
  }
}
```

15.13.USER_PRODUCT_STEAM (STEAMS)

```
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/services/steams/data_stream.dart';

class UsersProductsStream extends DataStream<List<String>> {
  @override
  void reload() {
    final usersProductsFuture = ProductDatabaseHelper().usersProductsList;
    usersProductsFuture.then((data) {
      addData(data);
    }).catchError((e) {
      addError(e);
    });
  }
}
```

16.CÁC LỚP EXCEPTION

CLASS EXCEPTION Gồm :

- Database Exception
- Local File Exception

16.1. ACTIONS_EXCEPTION.DART

```
import 'package:flutter_online_shop/exceptions/auth/firebase_exception.dart';

///Kế thừa class abstract MessagedFirebaseAuthException

class FirebaseCredentialActionAuthException
    extends MessagedFirebaseAuthException {
  FirebaseCredentialActionAuthException(
      {String message = "Instance of FirebasePasswordActionAuthException"})
      : super(message);
}

//Bắt exception [Không tồn tại user]

class FirebaseCredentialActionAuthUserNotFoundException
    extends FirebaseCredentialActionAuthException {
  FirebaseCredentialActionAuthUserNotFoundException(
      {String message = "No such user exist"})
      : super(message: message);
}

//Bắt exception [Mật khẩu yếu]

class FirebaseCredentialActionAuthWeakPasswordException
    extends FirebaseCredentialActionAuthException {
  FirebaseCredentialActionAuthWeakPasswordException(
      {String message = "Password is weak, try something better"})
      : super(message: message);
}

//Bắt exception [Yêu cầu đăng nhập lại]

class FirebaseCredentialActionAuthRequiresRecentLoginException
    extends FirebaseCredentialActionAuthException {
  FirebaseCredentialActionAuthRequiresRecentLoginException(
      {String message = "This action requires re-Login"})
      : super(message: message);
```

```
}

//Bắt exception [Đăng nhập thất bại bởi lỗi không xác định]

class FirebaseCredentialActionAuthUnknownReasonFailureException
    extends FirebaseCredentialActionAuthException {
  FirebaseCredentialActionAuthUnknownReasonFailureException(
      {String message = "The action can't be performed due to unknown reason"})
      : super(message: message);
}
```

16.2. FIREBASE_EXCEPTION.DART

```
import 'package:firebase_auth/firebase_auth.dart';

abstract class MessagedFirebaseAuthException extends FirebaseAuthException {
  String _message;
  MessagedFirebaseAuthException(this._message);
  String get message => _message;
  @override
  String toString() {
    return message;
  }
}
```

16.3. REAUTH_EXCEPTION.DART

```
import 'package:flutter_online_shop/exceptions/auth/firebase_exception.dart';

///Kế thừa class abstract [MessagedFirebaseAuthException]

class FirebaseReauthException extends MessagedFirebaseAuthException {
  FirebaseReauthException(
      {String message: "Instance of FirebaseReauthException"})
      : super(message);
}

///Kế thừa class [FirebaseReauthException]

//Bắt exception [Không trùng khớp user hiện tại]

class FirebaseReauthUserMismatchException extends FirebaseReauthException {
  FirebaseReauthUserMismatchException(
      {String message: "User not matching with current user"})
      : super(message: message);
}
```

```
//Bắt exception [Không tồn tại user]

class FirebaseReauthUserNotFoundException extends FirebaseReauthException {
  FirebaseReauthUserNotFoundException({String message = "No such user exists"})
      : super(message: message);
}

//Bắt exception [Thông tin không hợp lệ]

class FirebaseReauthInvalidCredentialException extends FirebaseReauthException {
  FirebaseReauthInvalidCredentialException(
      {String message = "Invalid Credentials"})
      : super(message: message);
}

//Bắt exception [Email không hợp lệ]

class FirebaseReauthInvalidEmailException extends FirebaseReauthException {
  FirebaseReauthInvalidEmailException({String message = "Invalid Email"})
      : super(message: message);
}

//Bắt exception [Sai password]

class FirebaseReauthWrongPasswordException extends FirebaseReauthException {
  FirebaseReauthWrongPasswordException({String message = "Wrong password"})
      : super(message: message);
}

//Bắt exception [Mã xác nhận không hợp lệ]

class FirebaseReauthInvalidVerificationCodeException
    extends FirebaseReauthException {
  FirebaseReauthInvalidVerificationCodeException(
      {String message = "Invalid Verification Code"})
      : super(message: message);
}

//Bắt exception [ID xác nhận không hợp lệ]

class FirebaseReauthInvalidVerificationIdException
    extends FirebaseReauthException {
  FirebaseReauthInvalidVerificationIdException(
      {String message = "Invalid Verification ID"})
      : super(message: message);
}
```

```
//Bắt exception [Xác nhận người dùng thất bại]

class FirebaseReauthUnknownReasonFailureException
    extends FirebaseReauthException {
  FirebaseReauthUnknownReasonFailureException(
      {String message = "Reauthentification Failed due to unknown reason"})
      : super(message: message);
}
```

16.4. SIGN_IN_EXCEPTION.DART

```
import 'package:flutter_online_shop/exceptions/auth/firebase_exception.dart';

class FirebaseSignInAuthException extends MessagedFirebaseAuthException {
  FirebaseSignInAuthException(
      {String message: "Instance of FirebaseSignInAuthException"})
      : super(message);
}

//Bắt exception [Người dùng vô hiệu hoá]

class FirebaseSignInAuthUserDisabledException
    extends FirebaseSignInAuthException {
  FirebaseSignInAuthUserDisabledException(
      {String message = "Người dùng này đã bị vô hiệu hóa"})
      : super(message: message);
}

//Bắt exception [Không tồn tại người dùng]

class FirebaseSignInAuthUserNotFoundException
    extends FirebaseSignInAuthException {
  FirebaseSignInAuthUserNotFoundException(
      {String message = "Không tìm thấy người dùng này"})
      : super(message: message);
}

//Bắt exception [Email không hợp lệ]

class FirebaseSignInAuthInvalidEmailException
    extends FirebaseSignInAuthException {
  FirebaseSignInAuthInvalidEmailException(
      {String message = "Email không hợp lệ"})
      : super(message: message);
}

//Bắt exception [Sai mật khẩu]
```

```
class FirebaseSignInAuthWrongPasswordException
    extends FirebaseSignInAuthException {
  FirebaseSignInAuthWrongPasswordException({String message = "Sai mật khẩu"})
      : super(message: message);
}
//Bắt exception [Người dùng chưa được xác thực]

class FirebaseSignInAuthUserNotVerifiedException
    extends FirebaseSignInAuthException {
  FirebaseSignInAuthUserNotVerifiedException(
      {String message = "Người dùng này chưa được xác minh"})
      : super(message: message);
}

//Bắt exception [Đăng nhập thất bại]

class FirebaseSignInAuthUnknownReasonFailure
    extends FirebaseSignInAuthException {
  FirebaseSignInAuthUnknownReasonFailure(
      {String message = "Đăng nhập thất bại"})
      : super(message: message);
}
```

16.5. SIGN_UP_EXCEPTION.DART

```
import 'package:flutter_online_shop/exceptions/auth/firebase_exception.dart';

class FirebaseSignUpAuthException extends MessagedFirebaseAuthException {
  FirebaseSignUpAuthException(
      {String message: "Instance of FirebaseSignUpAuthException"})
      : super(message);
}

//Bắt exception [Xác nhận người dùng thất bại]

class FirebaseSignUpAuthEmailAlreadyInUseException
    extends FirebaseSignUpAuthException {
  FirebaseSignUpAuthEmailAlreadyInUseException(
      {String message = "Email đã tồn tại"})
      : super(message: message);
}
```

```
//Bắt exception [Email không hợp lệ]

class FirebaseSignUpAuthInvalidEmailException
    extends FirebaseSignUpAuthException {
  FirebaseSignUpAuthInvalidEmailException(
      {String message = "Email không hợp lệ"})
      : super(message: message);
}

//Bắt exception [Đăng kí bị hạn chế]

class FirebaseSignUpAuthOperationNotAllowedException
    extends FirebaseSignUpAuthException {
  FirebaseSignUpAuthOperationNotAllowedException(
      {String message = "Đăng ký bị hạn chế đối với người dùng này"})
      : super(message: message);
}

//Bắt exception [Mật khẩu yếu]

class FirebaseSignUpAuthWeakPasswordException
    extends FirebaseSignUpAuthException {
  FirebaseSignUpAuthWeakPasswordException(
      {String message = "Mật khẩu yếu, hãy thử mật khẩu tốt hơn"})
      : super(message: message);
}

//Bắt exception [Đăng kí tài khoản thất bại]

class FirebaseSignUpAuthUnknownReasonFailureException
    extends FirebaseSignUpAuthException {
  FirebaseSignUpAuthUnknownReasonFailureException(
      {String message = "Đăng kí tài khoản thất bại"})
      : super(message: message);
}
```

16.5. IMAGE_PICKING_EXCEPTION.DART

```
import 'package:flutter_online_shop/exceptions/local_file/local_files_exceptions.dart';

///Các exception về hình ảnh

class LocalImagePickingException extends LocalFileHandlingException {
  LocalImagePickingException(
      {String message = "Instance of ImagePickingException"})
      : super(message);
}

//Bắt exception [Hình ảnh không hợp lệ]

class LocalImagePickingInvalidImageException
    extends LocalImagePickingException {
  LocalImagePickingInvalidImageException(
      {String message = "Image chosen is invalid"})
      : super(message: message);
}

//Bắt exception [Kích thước hình ảnh vượt quá giới hạn cho phép]

class LocalImagePickingFileSizeOutOfBoundsException
    extends LocalImagePickingException {
  LocalImagePickingFileSizeOutOfBoundsException(
      {String message = "Image size not in given range"})
      : super(message: message);
}

//Bắt exception [Nguồn ảnh không hợp lệ]

class LocalImagePickingInvalidImageSourceException
    extends LocalImagePickingException {
  LocalImagePickingInvalidImageSourceException(
      {String message = "Image source is invalid"})
      : super(message: message);
}

//Bắt exception [Thật bại]

class LocalImagePickingUnknownReasonFailureException
    extends LocalImagePickingException {
  LocalImagePickingUnknownReasonFailureException(
      {String message = "Failed due to unknown reason"})
      : super(message: message);
}
```

16.6.LOCAL_FILES_EXCEPTION.DART

```
abstract class LocalFileHandlingException {
  String _message;
  LocalFileHandlingException(this._message);
  String get message => _message;
  @override
  String toString() {
    return message;
  }
}

class LocalFileHandlingStorageReadPermissionDeniedException
    extends LocalFileHandlingException {
  LocalFileHandlingStorageReadPermissionDeniedException(
      {String message = "Storage Read permissions not granted"})
      : super(message);
}
```

16.CÁC LỚP WIDGET CON KHÁC

Chứa các widget gồm :

1. Change_info_button.dart
2. CircleContainer.dart
3. Custom_stufix_icon.dart
4. Default_button.dart
5. Icon_button_counter.dart
6. No_acc_text.dart
7. Nothing_show.dart
8. Product_card_listview.dart
9. Product_short_details_card.dart
10. Round_icon_button.dart
11. Search_textfield.dart
12. Social_card.dart
13. Top_round_container.dart
14. Product_card.dart

16.1.CHANG_INFO_BUTTON.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';

class ChangeInfoButton extends StatelessWidget {
  final String text;
  final Function press;
  final Color color;
  final IconData icon;
  const ChangeInfoButton(
      {Key key, this.text, this.press, this.color, this.icon})
      : super(key: key);

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      width: double.infinity,
      height: getProportionateScreenHeight(56),
      child: FlatButton(
        color: kPrimaryColor,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.only(
            topLeft: Radius.circular(1),
            bottomRight: Radius.circular(25),
            topRight: Radius.circular(1),
            bottomLeft: Radius.circular(25),
          ),
        ),
        onPressed: press,
        child: Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            Text(
              text,
              style: TextStyle(
                fontSize: getProportionateScreenWidth(18),
                fontWeight: FontWeight.bold,
                color: Colors.white,
              ),
            ),
            icon != null
                ? SizedBox(
                    width: SizeConfig.screenWidth * .02,
                  )
                : Container(),
```

```
            Icon(
              icon ?? Icons.no_cell,
              color: icon != null ? Colors.white : Colors.transparent,
              size: 30,
            )
          ],
        ),
      ),
    );
  }
}
```

16.2.CIRCLECONTAINER.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/size_config.dart';

class CircleContainer extends StatelessWidget {
  final double sizeRate;
  final Color color;

  CircleContainer({this.color, this.sizeRate});
  @override
  Widget build(BuildContext context) {
    return Container(
      width: SizeConfig.screenWidth * sizeRate,
      height: SizeConfig.screenWidth * sizeRate,
      decoration: BoxDecoration(shape: BoxShape.circle, color: color),
    );
  }
}
```

16.3.CUSTOM_STUFIX_ICON.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_svg/flutter_svg.dart';

class CustomSuffixIcon extends StatelessWidget {
  final String svgIcon;
  const CustomSuffixIcon({
    Key key,
    @required this.svgIcon,
  }) : super(key: key);
```

```
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: EdgeInsets.fromLTRB(
        0,
        getProportionateScreenWidth(20),
        getProportionateScreenWidth(20),
        getProportionateScreenWidth(20),
      ),
      child: SvgPicture.asset(
        svgIcon,
        height: getProportionateScreenWidth(18),
      ),
    );
  }
}
```

16.4.DEFAUT_BUTTON.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';

class DefaultButton extends StatelessWidget {
  final String text;
  final Function press;
  final Color color;
  const DefaultButton({Key key, this.text, this.press, this.color})
      : super(key: key);

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      width: double.infinity,
      height: getProportionateScreenHeight(56),
      child: FlatButton(
        color: kPrimaryColor,
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.only(
            topLeft: Radius.circular(25),
            bottomRight: Radius.circular(45),
            topRight: Radius.circular(25),
            bottomLeft: Radius.circular(45),
          ),
        ),
```

```
        onPressed: press,
        child: Text(
          text,
          style: TextStyle(
            fontSize: getProportionateScreenWidth(18),
            fontWeight: FontWeight.bold,
            color: Colors.white,
          ),
        ),
      ),
    );
  }
}
```

16.5.ICON_BUTTON_COUNTER.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter/size_config.dart';
import 'package:flutter_svg/flutter_svg.dart';

class IconButtonWithCounter extends StatelessWidget {
  final String svgSrc;
  final int numOfItems;
  final GestureTapCallback press;
  const IconButtonWithCounter({
    Key key,
    @required this.svgSrc,
    this.numOfItems = 0,
    @required this.press,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return InkWell(
      onTap: press,
      borderRadius: BorderRadius.circular(50),
      child: Stack(
        clipBehavior: Clip.none,
        children: [
          Container(
            padding: EdgeInsets.all(getProportionateScreenWidth(12)),
            width: getProportionateScreenWidth(46),
            height: getProportionateScreenWidth(46),
            decoration: BoxDecoration(
              color: kSecondaryColor.withOpacity(0.1),
              shape: BoxShape.circle,
```

```
            ),
            child: SvgPicture.asset(
              svgSrc,
              color: Colors.black87,
            ),
          ),
          if (numOfItems > 0)
            Positioned(
              right: 0,
              top: -3,
              child: Container(
                width: getProportionateScreenWidth(20),
                height: getProportionateScreenWidth(20),
                decoration: BoxDecoration(
                  color: Color(0xFFFF4848),
                  shape: BoxShape.circle,
                ),
                child: Center(
                  child: Text(
                    "$numOfItems",
                    style: TextStyle(
                      fontSize: getProportionateScreenWidth(10),
                      color: Colors.white,
                      height: 1,
                      fontWeight: FontWeight.w600,
                    ),
                  ),
                ),
              ),
            ),
        ],
      ),
    );
  }
}
```

16.6.NO_ACC_TEXT.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/src/sign_up/sign_up_screen.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';

class NoAccountText extends StatelessWidget {
  const NoAccountText({
    Key key,
  }) : super(key: key);
```

```
  @override
  Widget build(BuildContext context) {
    return Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        GestureDetector(
          onTap: () {
            Navigator.push(context,
                MaterialPageRoute(builder: (context) => SignUpScreen()));
          },
          child: Text(
            "Đăng kí ngay",
            style: TextStyle(
              fontSize: getProportionateScreenWidth(16),
              fontWeight: FontWeight.w800,
              color: kPrimaryColor,
            ),
          ),
        ),
      ],
    );
  }
}
```

16.7.NOTHING_SHOW.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_svg/flutter_svg.dart';

class NothingToShowContainer extends StatelessWidget {
  final String iconPath;
  final String primaryMessage;
  final String secondaryMessage;

  const NothingToShowContainer({
    Key key,
    this.iconPath = "assets/icons/empty_box.svg",
    this.primaryMessage = "Hiện tại đang trống",
    this.secondaryMessage = "",
  }) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return SizedBox(
      width: SizeConfig.screenWidth * 0.75,
```

```
      height: SizeConfig.screenHeight * 0.25,
      child: Column(
        children: [
          SvgPicture.asset(
            iconPath,
            color: kTextColor,
            width: getProportionateScreenWidth(75),
          ),
          SizedBox(height: 16),
          Text(
            "$primaryMessage",
            style: TextStyle(
              color: kTextColor,
              fontSize: 15,
            ),
          ),
          SizedBox(height: 5),
          Text(
            "$secondaryMessage",
            textAlign: TextAlign.center,
            style: TextStyle(
              color: kPrimaryColor,
              fontSize: 17,
              fontWeight: FontWeight.bold,
            ),
          ),
        ],
      ),
    );
  }
}
```

16.8.PRODUCT_CARD_LISTVIEW.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:logger/logger.dart';

class ProductCardListView extends StatelessWidget {
  final String productId;
  final GestureTapCallback press;
  const ProductCardListView({
    Key key,
```

```
  @required this.productId,
  @required this.press,
}) : super(key: key);

@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: press,
    child: Padding(
      padding: EdgeInsets.fromLTRB(10, 10, 10, 10),
      child: Container(
        height: SizeConfig.screenHeight * .22,
        decoration: BoxDecoration(
          color: Colors.white,
          border: Border.all(color: kTextColor.withOpacity(0.15)),
          borderRadius: BorderRadius.all(
            Radius.circular(15),
          ),
        ),
        child: Padding(
          padding: const EdgeInsets.symmetric(horizontal: 15, vertical: 10),
          child: FutureBuilder<Product>(
            future: ProductDatabaseHelper().getProductWithID(productId),
            builder: (context, snapshot) {
              if (snapshot.hasData) {
                final Product product = snapshot.data;
                return buildProductCardItems(product);
              } else if (snapshot.connectionState ==
                  ConnectionState.waiting) {
                return Center(
                  child: Center(child: CircularProgressIndicator()),
                );
              } else if (snapshot.hasError) {
                final error = snapshot.error.toString();
                Logger().e(error);
              }
              return Center(
                child: Icon(
                  Icons.error,
                  color: kTextColor,
                  size: 60,
                ),
              );
            },
          ),
        ),
      ),
    ),
```

```
  );
}

Row buildProductCardItems(Product product) {
  return Row(
    children: [
      Padding(
        padding: const EdgeInsets.all(8.0),
        child: ClipRRect(
          borderRadius: BorderRadius.circular(5),
          child: Image.network(
            product.images[0],
            width: SizeConfig.screenWidth * .2,
            fit: BoxFit.fitWidth,
          ),
        ),
      ),
      SizedBox(
        width: getProportionateScreenWidth(10),
      ),
      Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: [
          Container(
            width: SizeConfig.screenWidth * .3,
            child: Text(
              '${product.title}',
              style: TextStyle(
                color: kTextColor,
                fontSize: 13,
                fontWeight: FontWeight.bold,
              ),
              maxLines: 2,
              softWrap: true,
              overflow: TextOverflow.ellipsis,
            ),
          ),
          SizedBox(
            height: getProportionateScreenWidth(15),
          ),
          Row(
            children: [
              Text.rich(
                TextSpan(
                  text: "${product.discountPrice}\đ\n",
                  style: TextStyle(
                    color: kPrimaryColor,
                    fontWeight: FontWeight.w700,
```

```
            fontSize: 12,
          ),
          children: [
            TextSpan(
              text: "${product.originalPrice}\đ",
              style: TextStyle(
                color: kTextColor,
                decoration: TextDecoration.lineThrough,
                fontWeight: FontWeight.normal,
                fontSize: 11,
              ),
            ),
          ],
        ),
      ),
      SizedBox(
        width: getProportionateScreenWidth(20),
      ),
      Stack(
        children: [
          Positioned(
            child: SvgPicture.asset(
              "assets/icons/DiscountTag.svg",
              color: kPrimaryColor.withOpacity(.4),
              width: SizeConfig.screenWidth * .1,
            ),
          ),
          Center(
            child: Padding(
              padding: EdgeInsets.only(
                  top: getProportionateScreenHeight(6),
                  left: getProportionateScreenWidth(8)),
              child: Text(
                "${product.calculatePercentageDiscount()}%",
                style: TextStyle(
                  color: Colors.red,
                  fontSize: SizeConfig.screenWidth * .04,
                  fontWeight: FontWeight.w700,
                ),
                textAlign: TextAlign.center,
              ),
            ),
          ),
        ],
      ),
    ],
  ),
],
```

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

16.9.PRODUCT_SHORT_DETAILS_CARD.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:logger/logger.dart';

class ProductShortDetailCard extends StatelessWidget {
  final String productId;
  final VoidCallback onPressed;
  const ProductShortDetailCard({
    Key key,
    @required this.productId,
    @required this.onPressed,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return InkWell(
      onTap: onPressed,
      child: FutureBuilder<Product>(
        future: ProductDatabaseHelper().getProductWithID(productId),
        builder: (context, snapshot) {
          if (snapshot.hasData) {
            final product = snapshot.data;
            return Row(
              children: [
                SizedBox(
                  width: getProportionateScreenWidth(88),
                  child: AspectRatio(
                    aspectRatio: 0.88,
                    child: Padding(
                      padding: EdgeInsets.all(10),
                      child: product.images.length > 0
                          ? Image.network(
                              product.images[0],
                              fit: BoxFit.contain,
                            )
```

```
                : Text("No Image"),
            ),
          ),
        ),
        SizedBox(width: getProportionateScreenWidth(20)),
        Expanded(
          child: Column(
            crossAxisAlignment: CrossAxisAlignment.start,
            children: [
              Text(
                product.title,
                softWrap: true,
                overflow: TextOverflow.ellipsis,
                style: TextStyle(
                  fontSize: 13,
                  fontWeight: FontWeight.bold,
                  color: kTextColor,
                ),
                maxLines: 2,
              ),
              SizedBox(height: 10),
              Text.rich(
                TextSpan(
                    text: "${product.discountPrice}\đ    ",
                    style: TextStyle(
                      color: kPrimaryColor,
                      fontWeight: FontWeight.w700,
                      fontSize: 12,
                    ),
                    children: [
                      TextSpan(
                        text: "${product.originalPrice}\đ",
                        style: TextStyle(
                          color: kTextColor,
                          decoration: TextDecoration.lineThrough,
                          fontWeight: FontWeight.normal,
                          fontSize: 11,
                        ),
                      ),
                    ]),
              ),
            ],
          ),
        ),
      ],
    );
  } else if (snapshot.connectionState == ConnectionState.waiting) {
    return Center(child: CircularProgressIndicator());
```

```
          } else if (snapshot.hasError) {
            final errorMessage = snapshot.error.toString();
            Logger().e(errorMessage);
          }
          return Center(
            child: Icon(
              Icons.error,
              color: kTextColor,
              size: 60,
            ),
          );
        },
      ),
    );
  }
}
```

16.10.ROUND_ICON_BUTTON.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';

class RoundedIconButton extends StatelessWidget {
  final IconData iconData;
  final GestureTapCallback press;
  const RoundedIconButton({
    Key key,
    @required this.iconData,
    @required this.press,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: getProportionateScreenWidth(40),
      width: getProportionateScreenWidth(40),
      child: FlatButton(
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(50),
        ),
        padding: EdgeInsets.zero,
        color: Colors.white,
        onPressed: press,
        child: Icon(
          iconData,
          color: kTextColor,
```

```
        ),
      ),
    );
  }
}
```

16.11.SEARCH_TEXTFIELD.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';

class SearchField extends StatelessWidget {
  final Function onSubmit;
  const SearchField({
    Key key,
    @required this.onSubmit,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      decoration: BoxDecoration(
        color: kSecondaryColor.withOpacity(0.1),
        borderRadius: BorderRadius.circular(15),
      ),
      child: TextField(
        textInputAction: TextInputAction.search,
        decoration: InputDecoration(
          enabledBorder: InputBorder.none,
          focusedBorder: InputBorder.none,
          hintText: "Tìm kiếm sản phẩm",
          prefixIcon: Icon(
            Icons.search,
            color: kPrimaryColor,
          ),
          contentPadding: EdgeInsets.symmetric(
              horizontal: getProportionateScreenWidth(20),
              vertical: getProportionateScreenWidth(9)),
        ),
        onSubmitted: onSubmit,
      ),
    );
  }
}
```

16.12.SOCIAL_CARD.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_svg/flutter_svg.dart';

class SocialCard extends StatelessWidget {
  final String icon;
  final Function press;
  const SocialCard({
    Key key,
    @required this.icon,
    this.press,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
      onTap: press,
      child: Container(
        height: getProportionateScreenHeight(40),
        width: getProportionateScreenWidth(40),
        margin:
            EdgeInsets.symmetric(horizontal: getProportionateScreenWidth(10)),
        padding: EdgeInsets.all(getProportionateScreenWidth(12)),
        decoration: BoxDecoration(
          color: Color(0xFFF5F6F9),
          shape: BoxShape.circle,
        ),
        child: SvgPicture.asset(icon),
      ),
    );
  }
}
```

16.13.TOP_ROUND_CONTAINER.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/size_config.dart';

class TopRoundedContainer extends StatelessWidget {
  final Color color;
  final Widget child;
  const TopRoundedContainer({
    Key key,
    this.color = Colors.white,
```

```
    @required this.child,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      margin: EdgeInsets.only(top: getProportionateScreenWidth(10)),
      padding: EdgeInsets.symmetric(
        horizontal: getProportionateScreenWidth(30),
        vertical: getProportionateScreenWidth(40),
      ),
      decoration: BoxDecoration(
        color: color,
        borderRadius: BorderRadius.only(
          topLeft: Radius.circular(64),
          topRight: Radius.circular(15),
          bottomRight: Radius.circular(64),
          bottomLeft: Radius.circular(15),
        ),
      ),
      child: child,
    );
  }
}
```

16.14.PRODUCT_CARD.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/models/product_model.dart';
import 'package:flutter_online_shop/services/db/product_db_helper.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/size_config.dart';
import 'package:flutter_svg/flutter_svg.dart';
import 'package:logger/logger.dart';

class ProductCard extends StatelessWidget {
  final String productId;
  final GestureTapCallback press;
  const ProductCard({
    Key key,
    @required this.productId,
    @required this.press,
  }) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return GestureDetector(
```

```
    onTap: press,
    child: Container(
      decoration: BoxDecoration(
        color: Colors.white,
        border: Border.all(color: kTextColor.withOpacity(0.15)),
        borderRadius: BorderRadius.all(
          Radius.circular(16),
        ),
      ),
      child: Padding(
        padding: const EdgeInsets.symmetric(horizontal: 15, vertical: 10),
        child: FutureBuilder<Product>(
          future: ProductDatabaseHelper().getProductWithID(productId),
          builder: (context, snapshot) {
            if (snapshot.hasData) {
              final Product product = snapshot.data;
              return buildProductCardItems(product);
            } else if (snapshot.connectionState == ConnectionState.waiting) {
              return Center(
                child: Center(child: CircularProgressIndicator()),
              );
            } else if (snapshot.hasError) {
              final error = snapshot.error.toString();
              Logger().e(error);
            }
            return Center(
              child: Icon(
                Icons.error,
                color: kTextColor,
                size: 60,
              ),
            );
          },
        ),
      ),
    ),
  );
}

Column buildProductCardItems(Product product) {
  return Column(
    mainAxisAlignment: MainAxisAlignment.center,
    children: [
      Flexible(
        flex: 2,
        child: Padding(
          padding: const EdgeInsets.all(8.0),
          child: ClipRRect(
```

```
        borderRadius: BorderRadius.circular(10),
        child: Image.network(
          product.images[0],
          fit: BoxFit.contain,
        ),
      ),
    ),
  ),
  SizedBox(height: 10),
  Flexible(
    flex: 2,
    child: Column(
      crossAxisAlignment: CrossAxisAlignment.start,
      children: [
        Flexible(
          flex: 1,
          child: Text(
            "${product.title}\n",
            style: TextStyle(
              color: kTextColor,
              fontSize: 13,
              fontWeight: FontWeight.bold,
            ),
            maxLines: 2,
            overflow: TextOverflow.ellipsis,
          ),
        ),
        SizedBox(height: 5),
        Flexible(
          flex: 1,
          child: Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              Flexible(
                flex: 5,
                child: Text.rich(
                  TextSpan(
                    text: "${product.discountPrice}\đ\n",
                    style: TextStyle(
                      color: kPrimaryColor,
                      fontWeight: FontWeight.w700,
                      fontSize: 12,
                    ),
                    children: [
                      TextSpan(
                        text: "${product.originalPrice}\đ",
                        style: TextStyle(
                          color: kTextColor,
```

```
                    decoration: TextDecoration.lineThrough,
                    fontWeight: FontWeight.normal,
                    fontSize: 11,
                  ),
                ),
              ],
            ),
          ),
        ),
        Flexible(
          flex: 3,
          child: Stack(
            children: [
              SvgPicture.asset(
                "assets/icons/DiscountTag.svg",
                color: kPrimaryColor.withOpacity(.4),
                width: SizeConfig.screenWidth * .4,
              ),
              Center(
                child: Padding(
                  padding: EdgeInsets.only(
                      left: getProportionateScreenWidth(
                          screenPadding)),
                  child: Text(
                    "${product.calculatePercentageDiscount()}%",
                    style: TextStyle(
                      color: Colors.red,
                      fontSize: SizeConfig.screenWidth * .04,
                      fontWeight: FontWeight.w700,
                    ),
                    textAlign: TextAlign.center,
                  ),
                ),
              ),
            ],
          ),
        ),
      ],
    ),
  ),
),
),
],
);
}
}
```

17.CÁC LỚP UTILS (LỚP ĐỊNH DẠNG)

17.1.CONSTANT.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/size_config.dart';

const String appName = "Globlux";

Color kPrimaryColor = Colors.lightGreen[800];
const kPrimaryLightColor = Color(0xFFFFECDF);
const kPrimaryGradientColor = LinearGradient(
  begin: Alignment.topLeft,
  end: Alignment.bottomRight,
  colors: [Color(0xFFFFA53E), Color(0xFFFF7643)],
);
const kSecondaryColor = Color(0xFF979797);
const kTextColor = Colors.black87;

const kAnimationDuration = Duration(milliseconds: 200);
final kCircleContainer = Colors.lightGreen[100];

const double screenPadding = 10;

final headingStyle = TextStyle(
  fontSize: getProportionateScreenWidth(40),
  fontWeight: FontWeight.bold,
  color: Colors.teal[800],
  fontFamily: 'Roboto',
  letterSpacing: 1.5,
  height: 1.5,
);

final titleStyle = TextStyle(
  fontSize: getProportionateScreenWidth(20),
  fontWeight: FontWeight.bold,
  color: Colors.white70,
  height: 1.5,
);

final outlineTextField = OutlineInputBorder(
  borderSide: BorderSide(color: kPrimaryColor),
  borderRadius: BorderRadius.only(
    topLeft: Radius.circular(25),
    bottomRight: Radius.circular(0),
```

```
    topRight: Radius.circular(0),
    bottomLeft: Radius.circular(25),
  ),
);

const defaultDuration = Duration(milliseconds: 250);

// Form Error
final RegExp emailValidatorRegExp =
    RegExp(r"^[a-zA-Z0-9.]+@[a-zA-Z0-9]+\.[a-zA-Z]+");
const String kEmailNullError = "Vui lòng nhập Email";
const String kInvalidEmailError = "Vui lòng nhập Email hợp lệ";
const String kPassNullError = "Vui lòng nhập mật khẩu";
const String kShortPassError = "Mật khẩu quá ngắn";
const String kMatchPassError = "Mật khẩu không trùng khớp";
const String kNamelNullError = "Vui lòng nhập tên của bạn";
const String kPhoneNumberNullError = "Vui lòng nhập số điện thoại";
const String kAddressNullError = "Vui lòng nhập địa chỉ";
const String FIELD_REQUIRED_MSG = "Bắt buộc";

final otpInputDecoration = InputDecoration(
  contentPadding:
      EdgeInsets.symmetric(vertical: getProportionateScreenWidth(15)),
  border: outlineInputBorder(),
  focusedBorder: outlineInputBorder(),
  enabledBorder: outlineInputBorder(),
);

OutlineInputBorder outlineInputBorder() {
  return OutlineInputBorder(
    borderRadius: BorderRadius.circular(getProportionateScreenWidth(15)),
    borderSide: BorderSide(color: kTextColor),
  );
}
```

17.2.SIZECONFIG.DART

```
import 'package:flutter/material.dart';

class SizeConfig {
  static MediaQueryData _mediaQueryData;
  static double screenWidth;
  static double screenHeight;
  static double defaultSize;
```

```
  static Orientation orientation;

  void init(BuildContext context) {
    _mediaQueryData = MediaQuery.of(context);
    screenWidth = _mediaQueryData.size.width;
    screenHeight = _mediaQueryData.size.height;
    orientation = _mediaQueryData.orientation;
  }
}
double getProportionateScreenHeight(double inputHeight) {
  double screenHeight = SizeConfig.screenHeight;
  return (inputHeight / 812.0) * screenHeight;
}
double getProportionateScreenWidth(double inputWidth) {
  double screenWidth = SizeConfig.screenWidth;
  return (inputWidth / 375.0) * screenWidth;
}
```

17.3.THEME.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/utils/constant.dart';

ThemeData theme() {
  return ThemeData(
    scaffoldBackgroundColor: Colors.white,
    fontFamily: "Muli",
    primarySwatch: Colors.deepOrange,
    appBarTheme: appBarTheme(),
    textTheme: textTheme(),
    inputDecorationTheme: inputDecorationTheme(),
    visualDensity: VisualDensity.adaptivePlatformDensity,
    snackBarTheme: SnackBarThemeData(
      backgroundColor: Colors.black,
      contentTextStyle: TextStyle(
        color: Colors.white,
      ),
    ),
  );
}

InputDecorationTheme inputDecorationTheme() {
  OutlineInputBorder outlineInputBorder = OutlineInputBorder(
    borderRadius: BorderRadius.circular(28),
    borderSide: BorderSide(color: kTextColor),
```

```
    gapPadding: 10,
  );
  return InputDecorationTheme(
    contentPadding: EdgeInsets.symmetric(horizontal: 42, vertical: 20),
    enabledBorder: outlineInputBorder,
    focusedBorder: outlineInputBorder,
    border: outlineInputBorder,
  );
}

TextTheme textTheme() {
  return TextTheme(
    bodyText1: TextStyle(
      color: kTextColor,
      fontSize: 16,
    ),
    bodyText2: TextStyle(
      color: kTextColor,
      fontSize: 14,
    ),
  );
}

AppBarTheme appBarTheme() {
  return AppBarTheme(
    color: Colors.white,
    elevation: 0,
    brightness: Brightness.light,
    iconTheme: IconThemeData(color: Colors.black),
    textTheme: TextTheme(
      headline6: TextStyle(color: Color(0XFF8B8B8B), fontSize: 18),
    ),
    centerTitle: true,
  );
}
```

17.4. UTILS.DART

```
import 'package:flutter/material.dart';

Future<bool> showConfirmationDialog(
  BuildContext context,
  String messege, {
  String positiveResponse = "Có",
  String negativeResponse = "Không",
```

```
}) async {
  var result = await showDialog(
    context: context,
    builder: (context) {
      return AlertDialog(
        content: Text(messege),
        shape: RoundedRectangleBorder(
          borderRadius: BorderRadius.circular(15),
        ),
        actions: [
          FlatButton(
            child: Text(
              positiveResponse,
              style: TextStyle(
                fontWeight: FontWeight.bold,
              ),
            ),
            onPressed: () {
              Navigator.pop(context, true);
            },
          ),
          FlatButton(
            child: Text(
              negativeResponse,
              style: TextStyle(
                fontWeight: FontWeight.bold,
              ),
            ),
            onPressed: () {
              Navigator.pop(context, false);
            },
          ),
        ],
      );
    },
  );
  if (result == null) result = false;
  return result;
}
```

18.LỚP WRAPPER & MAIN

18.1.AUTH_WRAPPER.DART

```
import 'package:flutter/material.dart';
import 'package:flutter_online_shop/services/auth/auth_services.dart';
import 'package:flutter_online_shop/src/home/home_screen.dart';
import 'package:flutter_online_shop/src/sign_in/sign_in_screen.dart';

///Đặt Route màn hình Home và màn hình Đăng nhập

class AuthentificationWrapper extends StatelessWidget {
  static const String routeName = "/authentification_wrapper";
  @override
  Widget build(BuildContext context) {
    return StreamBuilder(
      stream: AuthentificationService().authStateChanges,
      builder: (context, snapshot) {
        if (snapshot.hasData) {
          return HomeScreen();
        } else {
          return SignInScreen();
        }
      },
    );
  }
}
```

18.2.MAIN.DART

```
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_online_shop/utils/constant.dart';
import 'package:flutter_online_shop/utils/theme.dart';
import 'package:flutter_online_shop/wrappers/auth_wrapper.dart';

Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  SystemChrome.setPreferredOrientations([
    DeviceOrientation.portraitUp,
  ]);
  await Firebase.initializeApp();

  runApp(MyApp());
}
```

```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: appName,
      theme: theme(),
      debugShowCheckedModeBanner: false,
      home: AuthentificationWrapper(),
    );
  }
}
```

19.PHIÊN BẢN SDK & CÁC THƯ VIỆN

SDK: >=2.10.0 & <3.0.0

Thư viện:

```
  cloud_firestore: ^2.4.0
  cupertino_icons: ^1.0.0
  enum_to_string:
  firebase_auth: ^3.0.1
  firebase_core: ^1.4.0
  firebase_storage: ^10.0.1
  flutter_launcher_icons:
  flutter_rating_bar:
  flutter_svg:
  flutter_tags:
  future_progress_dialog:
  image_picker:
  intl: ^0.17.0
  logger:
  permission_handler: ^8.1.4+2
  pinch_zoom_image_updated:
  provider:
  string_validator:
  swipedetector:
  url_launcher: ^6.0.9
  blobs: ^2.0.0
  carousel_slider: ^4.0.0
```

PHẦN 4:BẢNG PHÂN CÔNG CÔNG VIỆC

4.1.Phân công tổng quát

Tên công việc	Phụ trách	Ngày thực hiện	Ngày kết thúc	Trạng thái
Ý tưởng phần mềm	Cao Hồng Chương Võ Hồng Quân Nguyễn Hoàng Hải	15/6/2021	20/6/2021	Hoàn thành
Phân tích & thiết kế	Cao Hồng Chương Võ Hồng Quân Nguyễn Hoàng Hải	24/6/2021	14/7/2021	Hoàn thành
Phác thảo giao diện	Võ Hồng Quân	15/7/2021	20/7/2021	Hoàn thành
Thiết kế giao diện	Cao Hồng Chương Võ Hồng Quân Nguyễn Hoàng Hải	13/8/2021	30/8/2021	Hoàn thành
Lập trình và hoàn thiện các chức năng	Cao Hồng Chương Võ Hồng Quân Nguyễn Hoàng Hải	14/8/2021	15/11/2021	Hoàn thành
Kiểm tra và sửa lỗi	Cao Hồng Chương Võ Hồng Quân Nguyễn Hoàng Hải	16/11/2021	14/3/2022	Hoàn thành
Báo cáo đồ án	Cao Hồng Chương Võ Hồng Quân Nguyễn Hoàng Hải	5/3/2022	10/3/2022	Hoàn thành

4.2.Phân công chi tiết

Công việc	Phụ trách	Trạng thái
Đăng nhập	Cao Hồng Chương Nguyễn Hoàng Hải	Hoàn thành
Đăng kí	Cao Hồng Chương Nguyễn Hoàng Hải	Hoàn thành
Các lớp xử lý(Services)	Võ Hồng Quân Cao Hồng Chương	Hoàn thành
Các lớp Model	Cao Hồng Chương	Hoàn thành
Màn hình chính	Võ Hồng Quân Cao Hồng Chương Nguyễn Hoàng Hải	Hoàn thành
Chỉnh sửa thông tin	Võ Hồng Quân Nguyễn Hoàng Hải	Hoàn thành
Tìm kiếm sản phẩm	Võ Hồng Quân	Hoàn thành
Chế độ bán hàng	Võ Hồng Quân Cao Hồng Chương Nguyễn Hoàng Hải	Hoàn thành
Đơn đặc hàng	Cao Hồng Chương Nguyễn Hoàng Hải	Hoàn thành

Công việc	Phụ trách	Trạng thái
Tra cứu sản phẩm	Võ Hồng Quân Cao Hồng Chương Nguyễn Hoàng Hải	Hoàn thành
Giỏ hàng	Võ Hồng Quân Cao Hồng Chương	Hoàn thành
Quản lý địa chỉ giao hàng	Cao Hồng Chương Nguyễn Hoàng Hải	Hoàn thành
Quên mật khẩu	Võ Hồng Quân	Hoàn thành
Đánh giá sản phẩm	Võ Hồng Quân Cao Hồng Chương	Hoàn thành
Sản phẩm yêu thích	Võ Hồng Quân	Chưa hoàn thành
Đánh giá phần mềm bằng AlertDialog	Võ Hồng Quân	Chưa hoàn thành