CT484: PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG

XÂY DỰNG ỨNG DỤNG MYSHOP - PHẦN 2

File báo cáo cần nộp là file PDF trong đó có ghi thông tin **mã sinh viên, họ tên, lớp học phần** cùng với **hình minh họa tại các bước kiểm tra kết quả thực thi, các chức năng (không cần chụp hình mã nguồn)**. Cuối file báo cáo ghi đường link đến GitHub mã nguồn của dự án.

Ứng dụng MyShop có các chức năng chính sau:

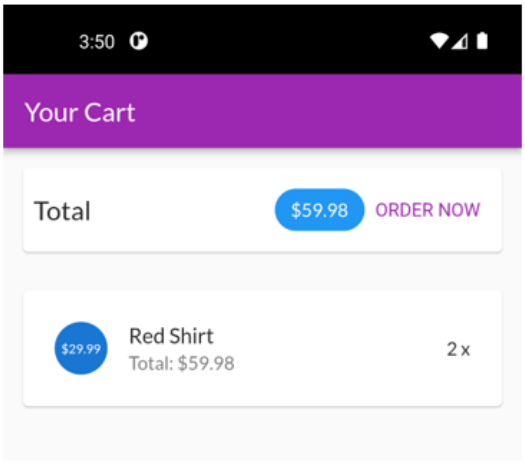
- Hiển thị và cập nhật danh mục các sản phẩm
- Xem chi tiết một sản phẩm
- Đánh dấu sản phẩm được yêu thích
- Thêm, xóa sản phẩm trong giỏ hàng
- Thực hiện đặt hàng, xem lại đơn hàng, chi tiết đơn hàng
- Đăng ký, đăng nhập
- Lưu trữ dữ liệu trên Firebase

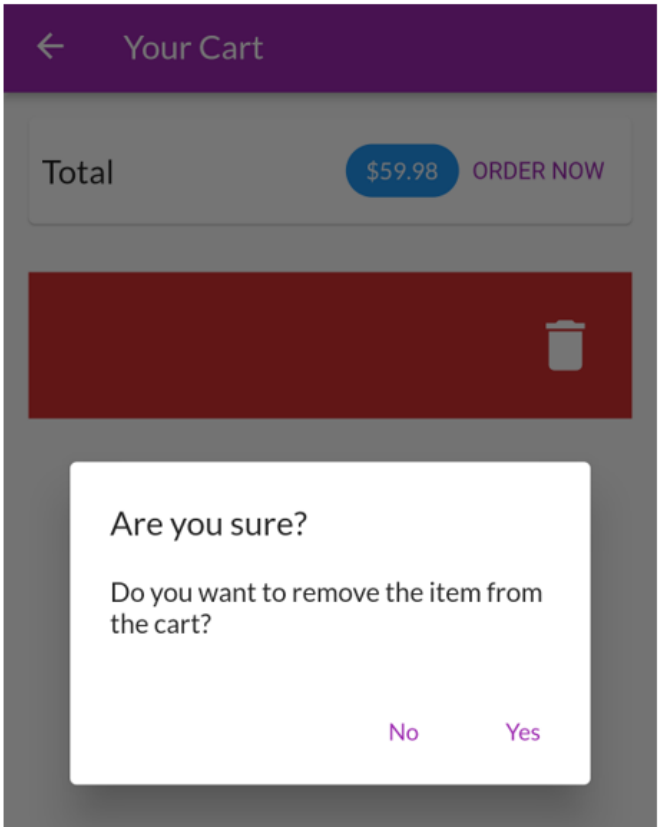
Sinh viên chỉ cần tạo MỘT báo cáo duy nhất cho tất cả các buổi thực hành. Nộp báo cáo lên Google Classroom và đẩy code lên GitHub trong thời hạn được chỉ định của mỗi buổi để báo cáo tiến độ, không tính điểm. Chỉ bài báo cáo nộp vào buổi thực hành cuối cùng sẽ được chấm điểm. Tuy nhiên, không nộp bài báo cáo tiến độ hoặc nộp trễ hạn thì không tính điểm buổi đó.

(Tiếp tục từ kết quả phần 1)

File báo cáo buổi 2 làm tiếp tục từ file báo cáo buổi 1.

Bước 1: Xây dựng trang hiển thị giỏ hàng





- Định nghĩa lớp **CartItem** miêu tả thông tin một mặt hàng trong giỏ hàng (*lib/models/cart_item.dart*):

```dart
class CartItem {
  final String id;
  final String title;
  final int quantity;
  final double price;

  CartItem({
    required this.id,
    required this.title,
    required this.quantity,
    required this.price,
  });

  CartItem copyWith({
    String? id,
    String? title,
    int? quantity,
    double? price,
  }) {
    return CartItem(
      id: id ?? this.id,
      title: title ?? this.title,
      quantity: quantity ?? this.quantity,
      price: price ?? this.price,
    );
  }
}
```

- Định nghĩa lớp **CartManager** quản lý các mặt hàng trong giỏ hàng (*lib/ui/cart/cart_manager.dart*):

```dart
import '../../models/cart_item.dart';

class CartManager {
  final Map<String, CartItem> _items = {
    'p1': CartItem(
      id: 'c1',
      title: 'Red Shirt',
      price: 29.99,
      quantity: 2,
    ),
  };

  int get productCount {
    return _items.length;
  }

  List<CartItem> get products {
    return _items.values.toList();
  }

  Iterable<MapEntry<String, CartItem>> get productEntries {
    return {..._items}.entries;
  }

  double get totalAmount {
    var total = 0.0;
    _items.forEach((key, cartItem) {
      total += cartItem.price * cartItem.quantity;
    });
    return total;
  }
}
```

- Định nghĩa thư viện các hàm tiện ích về hộp thoại (*lib/ui/shared/dialog_utils.dart*):

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

Future<bool?> showConfirmDialog(BuildContext context, String message) {
  return showDialog(
    context: context,
    builder: (ctx) => AlertDialog(
      title: const Text('Are you sure?'),
      content: Text(message),
      actions: <Widget>[
        TextButton(
          child: const Text('No'),
          onPressed: () {
            Navigator.of(ctx).pop(false);
          },
        ),
        TextButton(
          child: const Text('Yes'),
          onPressed: () {
            Navigator.of(ctx).pop(true);
          },
        ),
      ],
    ),
  );
}
```

Hàm **showDialog**(context, builder): hiển thị một hộp thoại phía trên nội dung hiện thời. Tham số builder là hàm tạo widget Dialog. Hai dạng widget Dialog phổ biến: **AlertDialog** và **SimpleDialog**. Các thuộc tính quan trọng của một **AlertDialog** là **title**: tựa đề hội thoại, **content**: nội dung chính hội thoại và **actions**: dòng các nút tương tác với hội thoại.

- Định nghĩa widget **CartItemCard** hiển thị thông tin một mặt hàng trong giỏ hàng (*lib/ui/cart/cart_item_card.dart*):

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

import '../../models/cart_item.dart';
import '../shared/dialog_utils.dart';

class CartItemCard extends StatelessWidget {
  final String productId;
  final CartItem cardItem;

  const CartItemCard({
    required this.productId,
    required this.cardItem,
    super.key,
  });

  @override
  Widget build(BuildContext context) {
    return Dismissible(
      key: ValueKey(cardItem.id),
      background: Container(
        color: Theme.of(context).colorScheme.error,
        alignment: Alignment.centerRight,
        padding: const EdgeInsets.only(right: 20),
        margin: const EdgeInsets.symmetric(
          horizontal: 15,
          vertical: 4,
        ),
        child: const Icon(
          Icons.delete,
          color: Colors.white,
          size: 40,
        ),
      ),
      direction: DismissDirection.endToStart,
      confirmDismiss: (direction) {
        return showConfirmDialog(
          context,
          'Do you want to remove the item from the cart?',
        );
      },
```

```
        onDismissed: (direction) {
          print('Cart item dismissed');
        },
        child: buildItemCard(),
      );
    }

    Widget buildItemCard() {
      return Card(
        ...
      );
    }
  }
```

Widget Dismissible là widget có thể được loại bỏ (dismiss) bằng cách vuốt theo hướng chỉ định. Một số thuộc tính quan trọng của Dismissable: **key** thuộc tính đùng định danh widget; **direction** chỉ định hướng vuốt widget (DismissDirection); **background** widget nền cho widget con; **confirmDismiss** hàm được gọi để xác nhận loại bỏ, trả về true widget sẽ được loại bỏ, ngược lại quay về vị trí cũ; **onDismissed** hàm được gọi sau khi widget đã được loại bỏ.

Hàm *buildItemCard*():

```
Widget buildItemCard() {
  return Card(
    margin: const EdgeInsets.symmetric(
      horizontal: 15,
      vertical: 4,
    ),
    child: Padding(
      padding: const EdgeInsets.all(8),
      child: ListTile(
        leading: CircleAvatar(
          child: Padding(
            padding: const EdgeInsets.all(5),
            child: FittedBox(
              child: Text('\$${cardItem.price}'),
            ),
          ),
        ),
        title: Text(cardItem.title),
        subtitle: Text('Total: \$${(cardItem.price * cardItem.quantity)}'),
        trailing: Text('${cardItem.quantity} x'),
      ),
    ),
  );
}
```

Widget FittedBox co giãn và bố trí con của nó theo tiêu chí được chỉ định bởi thuộc tính fit (mặc định là BoxFit.contain).

- Định nghĩa trang hiển thị thông tin một giỏ hàng (*lib/ui/cart/cart_screen.dart*):

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

import 'cart_manager.dart';
import 'cart_item_card.dart';

class CartScreen extends StatelessWidget {
  static const routeName = '/cart';

  const CartScreen({super.key});

  @override
  Widget build(BuildContext context) {
    final cart = CartManager();
    return Scaffold(
      appBar: AppBar(
        title: const Text('Your Cart'),
      ),
      body: Column(
        children: <Widget>[
          buildCartSummary(cart, context),
          const SizedBox(height: 10),
          Expanded(
            child: buildCartDetails(cart),
          )
        ],
      ),
    );
  }

  Widget buildCartDetails(CartManager cart) {
    return ListView(
      ...
    );
  }

  Widget buildCartSummary(CartManager cart, BuildContext context) {
    return Card(
      ...
    );
  }
}
```

Hàm *buildCartDetails*() và *buildCartSummary*():

```
Widget buildCartDetails(CartManager cart) {
  return ListView(
    children: cart.productEntries
        .map(
          (entry) => CartItemCard(
            productId: entry.key,
            cardItem: entry.value,
          ),
        )
        .toList(),
  );
}

Widget buildCartSummary(CartManager cart, BuildContext context) {
  return Card(
    margin: const EdgeInsets.all(15),
    child: Padding(
      padding: const EdgeInsets.all(8),
      child: Row(
        mainAxisAlignment: MainAxisAlignment.spaceBetween,
        children: <Widget>[
          const Text(
            'Total',
            style: TextStyle(fontSize: 20),
          ),
          const Spacer(),
          Chip(
            label: Text(
              '\$${cart.totalAmount.toStringAsFixed(2)}',
              style: TextStyle(
                color: Theme.of(context).primaryTextTheme.titleLarge?.color,
              ),
            ),
            backgroundColor: Theme.of(context).primaryColor,
          ),
          TextButton(
            onPressed: () {
              print('An order has been added');
            },
            style: TextButton.styleFrom(
              textStyle: TextStyle(color: Theme.of(context).primaryColor),
            ),
            child: const Text('ORDER NOW'),
          )
        ],
      ),
    ),
  );
}
```

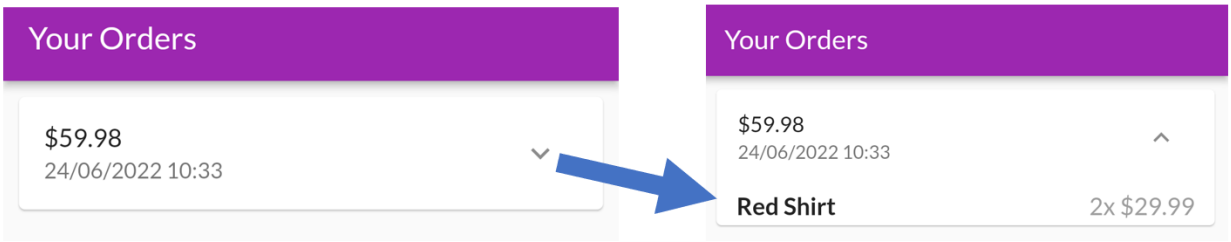
- Hiệu chỉnh *lib/main.dart* kiểm tra trang hiển thị giỏ hàng:

```
...
import 'ui/cart/cart_screen.dart';
...
class MyApp extends StatelessWidget {
    ...
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            ...
            home: const SafeArea(
                child: CartScreen(),
            ),
        );
    }
}
```

- Sau khi kiểm tra, lưu mã nguồn vào repo git và lên GitHub:

```
git add -u
git add lib/models lib/ui
git commit -m "Xay dung trang gio hang"
git push origin master
```

Bước 2: Xây dựng trang hiển thị các đặt hàng



- Định nghĩa lớp **OrderItem** miêu tả thông tin một đặt hàng (*lib/models/order_item.dart*):

```dart
import 'cart_item.dart';

class OrderItem {
  final String? id;
  final double amount;
  final List<CartItem> products;
  final DateTime dateTime;

  int get productCount {
    return products.length;
  }

  OrderItem({
    this.id,
    required this.amount,
    required this.products,
    DateTime? dateTime,
  }) : dateTime = dateTime ?? DateTime.now();

  OrderItem copyWith({
    String? id,
    double? amount,
    List<CartItem>? products,
    DateTime? dateTime,
  }) {
    return OrderItem(
      id: id ?? this.id,
      amount: amount ?? this.amount,
      products: products ?? this.products,
      dateTime: dateTime ?? this.dateTime,
    );
  }
}
```

- Định nghĩa lớp **OrderManager** quản lý các đặt hàng (*lib/ui/orders/order_manager.dart*):

```dart
import '../../models/cart_item.dart';
import '../../models/order_item.dart';

class OrdersManager with ChangeNotifier {
  final List<OrderItem> _orders = [
    OrderItem(
      id: 'o1',
      amount: 59.98,
      products: [
        CartItem(
          id: 'c1',
          title: 'Red Shirt',
          price: 29.99,
          quantity: 2,
        )
      ],
      dateTime: DateTime.now(),
    )
  ];

  int get orderCount {
    return _orders.length;
  }

  List<OrderItem> get orders {
    return [..._orders];
  }
}
```

- Định nghĩa widget **OrderItemCard** hiển thị thông tin một đặt hàng (*lib/ui/orders/order_item_card.dart*):

```dart
import 'dart:math';

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

import '../../models/order_item.dart';

class OrderItemCard extends StatefulWidget {
  final OrderItem order;

  const OrderItemCard(this.order, {super.key});

  @override
  State<OrderItemCard> createState() => _OrderItemCardState();
}

class _OrderItemCardState extends State<OrderItemCard> {
  var _expanded = false;

  @override
  Widget build(BuildContext context) {
    return Card(
      margin: const EdgeInsets.all(10),
      child: Column(
        children: <Widget>[
          buildOrderSummary(),
          if (_expanded) buildOrderDetails()
        ],
      ),
    );
  }

  Widget buildOrderDetails() {
    return Container(
      ...
    );
  }

  Widget buildOrderSummary() {
    return ListTile(
      ...
    );
  }
}
```

Hàm *buildOrderDetails*() và *buildOrderSummary*():

```
Widget buildOrderDetails() {
  return Container(
    padding: const EdgeInsets.symmetric(horizontal: 15, vertical: 4),
    height: min(widget.order.productCount * 20.0 + 10, 100),
    child: ListView(
      children: widget.order.products
          .map(
            (prod) => Row(
              mainAxisAlignment: MainAxisAlignment.spaceBetween,
              children: <Widget>[
                Text(
                  prod.title,
                  style: const TextStyle(
                    fontSize: 18,
                    fontWeight: FontWeight.bold,
                  ),
                ),
                Text(
                  '${prod.quantity}x \$${prod.price}',
                  style: const TextStyle(
                    fontSize: 18,
                    color: Colors.grey,
                  ),
                )
              ],
            ),
          )
          .toList(),
    ),
  );
}

Widget buildOrderSummary() {
  return ListTile(
    title: Text('\$${widget.order.amount}'),
    subtitle: Text(
      DateFormat('dd/MM/yyyy hh:mm').format(widget.order.dateTime),
    ),
    trailing: IconButton(
      icon: Icon(_expanded ? Icons.expand_less : Icons.expand_more),
      onPressed: () {
        setState(() {
          _expanded = !_expanded;
        });
      },
    ),
  );
}
```

Widget **OrderItemCard** sử dụng thư viện `intl` để định dạng ngày tháng, khai báo thư viện trong *pubspec.yaml* (chọn phiên bản mới nhất):

```
...
dependencies:
  flutter:
    sdk: flutter
  intl: ^0.17.0
...
```

(Hoặc có thể thêm thư viện `intl` bằng cách thực thi câu lệnh sau: `flutter pub add intl`).

- Định nghĩa trang hiển thị thông tin các đặt hàng (*lib/ui/orders/orders_screen.dart*):

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

import 'orders_manager.dart';
import 'order_item_card.dart';

class OrdersScreen extends StatelessWidget {
  const OrdersScreen({super.key});

  @override
  Widget build(BuildContext context) {
    print('building orders');
    final ordersManager = OrdersManager();
    return Scaffold(
      appBar: AppBar(
        title: const Text('Your Orders'),
      ),
      body: ListView.builder(
        itemCount: ordersManager.orderCount,
        itemBuilder: (ctx, i) => OrderItemCard(ordersManager.orders[i]),
      ),
    );
  }
}
```

- Hiệu chỉnh *lib/main.dart* kiểm tra trang hiển thị các đặt hàng:

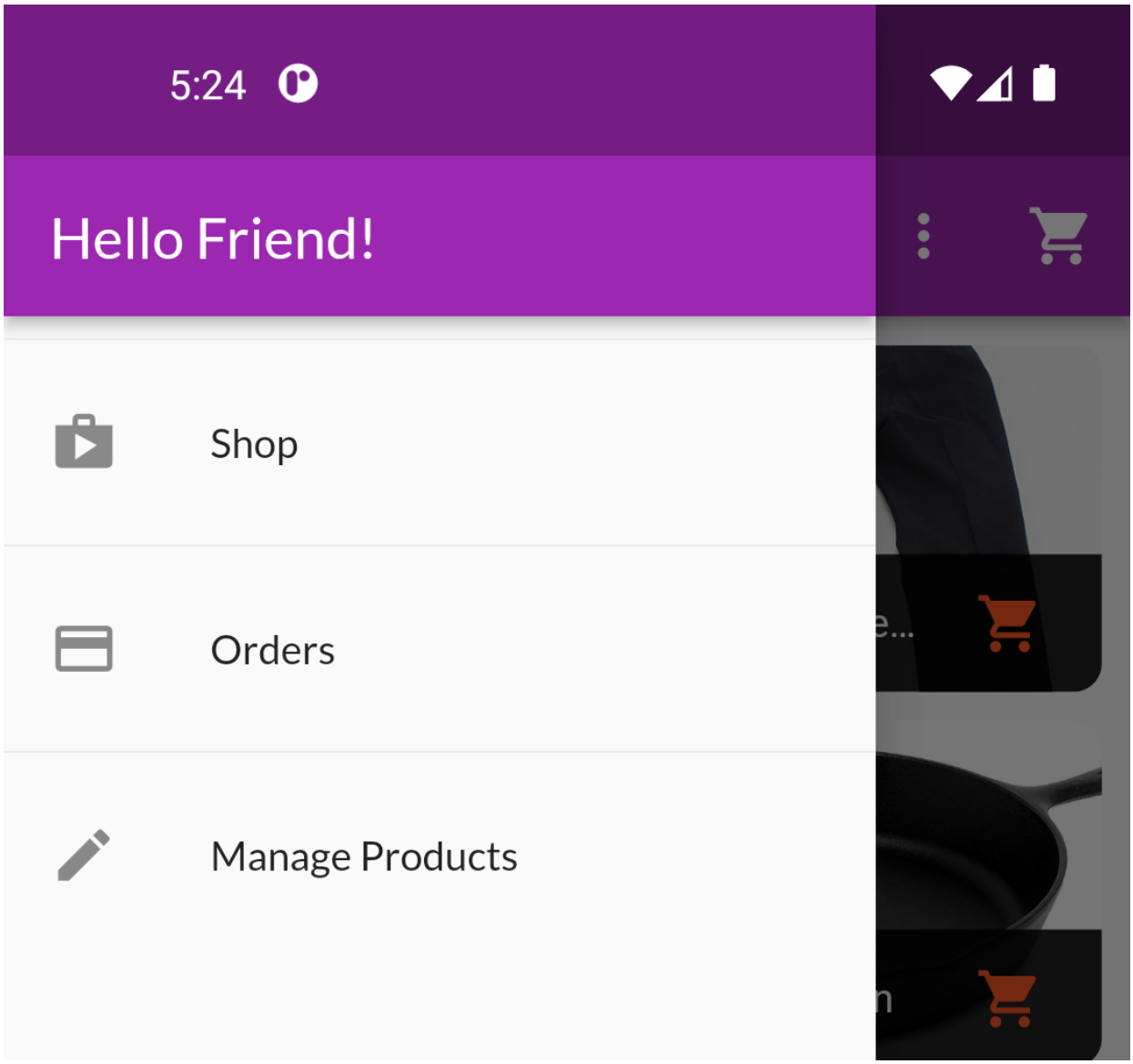
```
...
import 'ui/orders/orders_screen.dart';
...
class MyApp extends StatelessWidget {
    ...
    @override
    Widget build(BuildContext context) {
        return MaterialApp(
            ...
            home: const SafeArea(
```

```
                    child: OrdersScreen(),
                ),
            );
        }
    }
```

- Sau khi kiểm tra, lưu mã nguồn vào repo git và lên GitHub:

```
git add -u
git add lib/models lib/ui
git commit -m "Xay dung trang cac dat hang"
git push origin master
```

Bước 3: Định tuyến theo tên



- Tạo tập tin *lib/ui/screens.dart* là tập tin import tập trung của thư mục *lib/ui*:

```
export 'products/products_overview_screen.dart';
export 'products/product_detail_screen.dart';
export 'products/user_products_screen.dart';
export 'products/products_manager.dart';

export 'orders/orders_screen.dart';
export 'orders/orders_manager.dart';

export 'cart/cart_screen.dart';
export 'cart/cart_manager.dart';
```

Hiệu chỉnh tập tin *lib/main.dart*, thay vì import từng tập tin *_screen riêng lẻ, chỉ cần import tập tin *lib/ui/screens.dart*

- Đặt tên cho các trang màn hình:
  - *lib/ui/products/product_detail_screen.dart*:

```
class ProductDetailScreen extends StatelessWidget {
    static const routeName = '/product-detail';
    ...
}
```

  - *lib/ui/products/user_products_screen.dart*:

```
class UserProductsScreen extends StatelessWidget {
    static const routeName = '/user-products';
    ...
}
```

  - *lib/ui/cart/cart_screen.dart*:

```
class CartScreen extends StatelessWidget {
    static const routeName = '/cart';
    ...
}
```

  - *lib/ui/orders/orders_screen.dart*:

```
class OrdersScreen extends StatelessWidget {
    static const routeName = '/orders';
    ...
}
```

- Hiệu chỉnh *lib/main.dart*, khai báo các trang của ứng dụng:

```
class MyApp extends StatelessWidget {
    ...
    @override
```

```
        Widget build(BuildContext context) {
            return MaterialApp(
                ...
                home: const ProductsOverviewScreen(),
                routes: {
                    CartScreen.routeName:
                        (ctx) => const CartScreen(),
                    OrdersScreen.routeName:
                        (ctx) => const OrdersScreen(),
                    UserProductsScreen.routeName:
                        (ctx) => const UserProductsScreen(),
                },
                onGenerateRoute: (settings) {
                    if (settings.name == ProductDetailScreen.routeName) {
                        final productId = settings.arguments as String;
                        return MaterialPageRoute(
                            builder: (ctx) {
                                return ProductDetailScreen(
                                    ProductsManager().findById(productId)!,
                                );
                            },
                        );
                    }

                    return null;
                },
            );
        }
    }
```

Trong đoạn mã nguồn trên, phương thức *findById()* định nghĩa trong lớp **ProductsManager** (*lib/ui/products/products_manager.dart*) như sau:

```
    class ProductsManager {
        ...
        Product? findById(String id) {
            try {
                return _items.firstWhere((item) => item.id == id);
            } catch (error) {
                return null;
            }
        }
    }
```

- Định nghĩa widget **AppDrawer** thực hiện điều hướng trong ứng dụng (*lib/ui/shared/app_drawer.dart*):

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

import '../orders/orders_screen.dart';
import '../products/user_products_screen.dart';

class AppDrawer extends StatelessWidget {
  const AppDrawer({super.key});

  @override
  Widget build(BuildContext context) {
    return Drawer(
      child: Column(
        children: <Widget>[
          AppBar(
            title: const Text('Hello Friend!'),
            automaticallyImplyLeading: false,
          ),
          const Divider(),
          ListTile(
            leading: const Icon(Icons.shop),
            title: const Text('Shop'),
            onTap: () {
              Navigator.of(context).pushReplacementNamed('/');
            },
          ),
          const Divider(),
          ListTile(
            leading: const Icon(Icons.payment),
            title: const Text('Orders'),
            onTap: () {
              Navigator.of(context)
                  .pushReplacementNamed(OrdersScreen.routeName);
            },
          ),
          const Divider(),
          ListTile(
            leading: const Icon(Icons.edit),
            title: const Text('Manage Products'),
            onTap: () {
              Navigator.of(context)
                  .pushReplacementNamed(UserProductsScreen.routeName);
            },
          ),
        ],
      ),
    );
  }
}

```

- Thêm drawer vào các trang, điều chỉnh điều hướng giữa các trang:
    - *lib/ui/products/product_grid_tile.dart*:

```
child: GestureDetector(
    onTap: () {
        Navigator.of(context).pushNamed(
            ProductDetailScreen.routeName,
            arguments: product.id,
        );
    },
    ...
),
```

- *lib/ui/products/products_overview_screen.dart*:

```
...
import '../shared/app_drawer.dart';
...
    return Scaffold(
        appBar: AppBar(
            title: const Text('MyShop'),
            actions: <Widget>[
              buildProductFilterMenu(),
              buildShoppingCartIcon(),
            ],
        ),
        drawer: const AppDrawer(),
        body: ProductsGrid(_showOnlyFavorites),
    );
...

Widget buildShoppingCartIcon() {
    return IconButton(
        ...
        onPressed: () {
          Navigator.of(context).pushNamed(CartScreen.routeName);
        },
      ),
    );
}
```

- *lib/ui/products/user_products_screen.dart*:

```
...
import '../shared/app_drawer.dart';
...
    return Scaffold(
        ...
        drawer: const AppDrawer(),
        body: RefreshIndicator(...),
    );
...
```

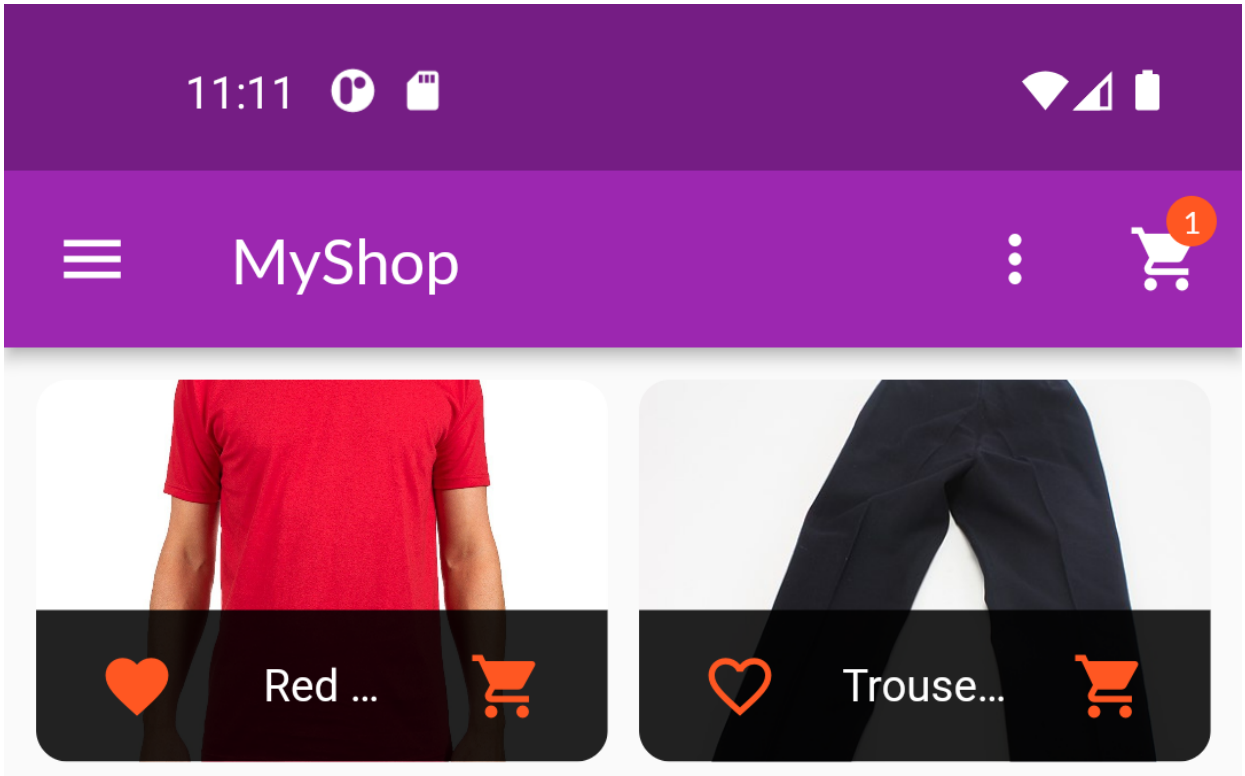
- *lib/ui/orders/orders_screen.dart*:

```
...
import '../shared/app_drawer.dart';
...
    return Scaffold(
        ...
        drawer: const AppDrawer(),
        body: ListView.builder(...),
    );
...
```

- Sau khi kiểm tra, lưu mã nguồn vào repo git và lên GitHub:

```
git add -u
git add lib/ui
git commit -m "Xay dung drawer va dinh tuyen theo ten"
git push origin master
```

Bước 4: Thêm thông tin số sản phẩm trong giỏ hàng



- Định nghĩa widget **TopRightBadge** (*lib/ui/products/top_right_badge.dart*) nhận vào một widget con và một giá trị. Giá trị sẽ được hiển thị ở góc trên bên phải của widget con:

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

class TopRightBadge extends StatelessWidget {
  const TopRightBadge({
    super.key,
    required this.child,
    required this.data,
    this.color,
  });

  final Widget child;
  final Object data;
  final Color? color;

  @override
  Widget build(BuildContext context) {
    return Stack(
      alignment: Alignment.center,
      children: [
        child,
        Positioned(
          right: 8,
          top: 8,
          child: Container(
            padding: const EdgeInsets.all(2.0),
            decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(10.0),
              color: color ?? Theme.of(context).colorScheme.secondary,
            ),
            constraints: const BoxConstraints(
              minWidth: 16,
              minHeight: 16,
            ),
            child: Text(
              data.toString(),
              textAlign: TextAlign.center,
              style: const TextStyle(
                fontSize: 10,
              ),
            ),
          ),
        )
      ],
    );
  }
}
```

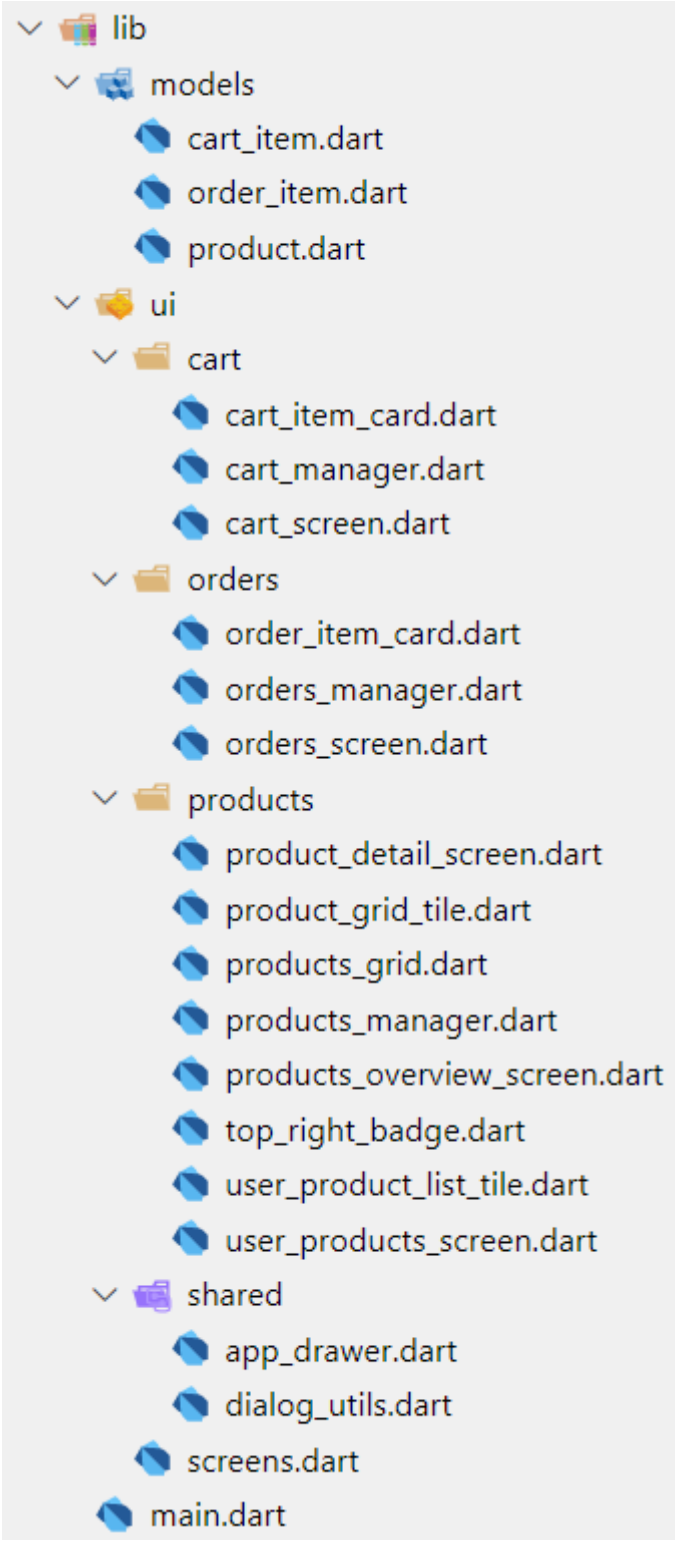
- Hiệu chỉnh trang tổng quan các sản phẩm (*lib/ui/products/products_overview_screen.dart*) bao widget **IconButton** với widget **TopRightBadge**:

```dart
...
import '../cart/cart_manager.dart';
```

```
import 'top_right_badge.dart';
...
class _ProductsOverviewScreenState extends State<ProductsOverviewScreen> {
    ...
    Widget buildShoppingCartIcon() {
        return TopRightBadge(
          data: CartManager().productCount,
          child: IconButton(
            icon: const Icon(
              Icons.shopping_cart,
            ),
            onPressed: () {
              Navigator.of(context).pushNamed(CartScreen.routeName);
            },
          ),
        );
    }
}
```

- Sau khi kiểm tra, lưu mã nguồn vào repo git và lên GitHub.

Cấu trúc thư mục dự án hiện tại:


lib
models
cart_item.dart
order_item.dart
product.dart
ui
cart
cart_item_card.dart
cart_manager.dart
cart_screen.dart
orders
order_item_card.dart
orders_manager.dart
orders_screen.dart
products
product_detail_screen.dart
product_grid_tile.dart
products_grid.dart
products_manager.dart
products_overview_screen.dart
top_right_badge.dart
user_product_list_tile.dart
user_products_screen.dart
shared
app_drawer.dart
dialog_utils.dart
screens.dart
main.dart