

**BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG
TRƯỜNG ĐẠI HỌC THỦY LỢI**



BÀI TẬP LỚN

**PHÁT TRIỂN ÚNG DỤNG CHO THIẾT BỊ DI ĐỘNG
ĐỀ TÀI: ÚNG DỤNG BÁN QUẦN ÁO**

Giáo viên hướng dẫn: ThS. Kiều Tuấn Dũng

Sinh viên thực hiện:

STT	Mã sinh viên	Họ và tên	Lớp
1	2251061790	Lê Tuấn Hùng	64CNTT1
2	2251061866	Hồ Văn Quang	64CNTT1
3	2251061865	Đỗ Minh Quân	64CNTT1
4	2251061755	Phan Quang Tùng Dương	64CNTT1

Hà Nội, năm 2025

**BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG
TRƯỜNG ĐẠI HỌC THỦY LỢI**



BÀI TẬP LỚN

**PHÁT TRIỂN ÚNG DỤNG CHO THIẾT BỊ DI ĐỘNG
ĐỂ TÀI: ÚNG DỤNG BÁN QUẦN ÁO FLUX STORE**

STT	Mã Sinh Viên	Họ và Tên	Ngày Sinh	Điểm	
				Bằng Số	Bằng Chữ
1	2251061790	Lê Tuấn Hùng	13/06/2004		
2	2251061866	Hồ Văn Quang	06/08/2004		
3	2251061865	Đỗ Minh Quân	29/12/2004		
4	2251061755	Phan Quang Tùng Dương	12/02/2004		

CÁN BỘ CHÁM THI

Hà Nội, năm 2025

LỜI NÓI ĐẦU

Trong thời đại công nghệ số phát triển không ngừng, nhu cầu mua sắm trực tuyến ngày càng trở nên phổ biến và thiết yếu trong đời sống hằng ngày. Các ứng dụng thương mại điện tử không chỉ giúp người tiêu dùng tiết kiệm thời gian, chi phí mà còn mang lại trải nghiệm mua sắm tiện lợi, nhanh chóng và đa dạng hơn bao giờ hết. Đặc biệt, trong lĩnh vực thời trang, việc áp dụng công nghệ vào bán hàng đang trở thành xu hướng tất yếu, giúp doanh nghiệp dễ dàng tiếp cận khách hàng và nâng cao hiệu quả kinh doanh.

Tuy nhiên, trên thị trường hiện nay, không ít người dùng vẫn gặp khó khăn trong việc tìm kiếm một ứng dụng mua sắm thời trang thân thiện, dễ sử dụng và có đầy đủ các chức năng như tìm kiếm, lọc sản phẩm, đánh giá chất lượng hay theo dõi đơn hàng. Điều này dẫn đến trải nghiệm mua sắm thiếu trọn vẹn, gây tâm lý e ngại khi lựa chọn mua hàng online.

Xuất phát từ nhu cầu ngày càng cao của người tiêu dùng về một nền tảng mua sắm hiện đại, thuận tiện và đáng tin cậy, Flux Store được xây dựng nhằm mang đến một ứng dụng bán quần áo thông minh, đa chức năng và thân thiện với người dùng. Ứng dụng không chỉ cung cấp danh mục sản phẩm đa dạng, cập nhật xu hướng thời trang mới nhất, mà còn hỗ trợ các tính năng như lọc theo danh mục, giá cả, đánh giá, giảm giá; tích hợp thanh toán nhanh chóng và quản lý đơn hàng tiện lợi.

Mục Lục

LỜI NÓI ĐẦU	3
Chương 1. TỔNG QUAN VỀ ĐỀ TÀI	7
1.1. Giới thiệu về đề tài	7
1.2. Mục tiêu của đề tài	8
1.3. Phạm vi của đề tài	8
1.4 Phân chia nhiệm vụ	9
Chương 2. KIẾN TRÚC VÀ CÔNG NGHỆ.....	11
2.1. Kiến trúc hệ thống	11
2.2. Giới thiệu về Công nghệ phát triển.....	11
Chương 3. XÂY DỰNG ỨNG DỤNG	13
3.1. Thiết kế Figma	13
3.2. Thiết kế CSDL	31
3.3. Giao diện ứng dụng	31
3.3.1. Màn hình giới thiệu	31
3.3.2. Màn hình đăng nhập & đăng ký	31
3.3.3. Màn hình trang chủ.....	32
3.3.4. Màn hình tìm kiếm	33
3.3.5. Màn hình chi tiết sản phẩm	33
3.3.6. Màn hình giỏ hàng.....	34
3.3.7. Màn hình thanh toán	34
3.3.8. Màn hình lịch sử mua hàng	35
3.3.9. Chi tiết đơn hàng	35
3.3.10. Màn thông tin người dùng	36

3.3.11. Màn hình danh sách voucher	37
3.3.12. Màn hình thống kê.....	37
3.4. Code minh họa các chức năng cốt lõi.....	37
3.4.1. Chức năng đăng ký.....	37
3.4.2. Chức năng đăng nhập.....	39
3.4.3. Chức năng đăng nhập bằng Google.....	40
3.4.4. Chức năng gio hàng.....	42
3.4.5. Chức năng đặt hàng	48
3.4.6. Chức năng hiển thị Collection.....	57
3.4.7. Chức năng hiển thị Product	61
3.4.8. Chức năng hiển thị Category	67
3.4.9. Chức năng tìm kiếm	71
3.4.10. Chức năng lọc sản phẩm.....	76
3.4.11. Chức năng hiển thị chi tiết sản phẩm	80
3.4.12. Chức năng hiển thị lịch sử mua hàng.....	92
3.4.13. Chức năng hiện chi tiết đơn hàng.....	95
3.4.14 Chức năng đánh giá sản phẩm.....	97
KẾT LUẬN	103
1. Kết quả đạt được	103
2. Nhược điểm	103
3. Hướng phát triển	104
TÀI LIỆU THAM KHẢO	106

DANH MỤC CÁC TỪ VIẾT TẮT

STT	TỪ VIẾT TẮT	VIẾT ĐẦY ĐỦ
1	RWD	Responsive Web Design
2		

Chương 1. TỔNG QUAN VỀ ĐỀ TÀI

1.1. Giới thiệu về đề tài

Ngày nay, sự phát triển mạnh mẽ của công nghệ thông tin đã tạo ra những chuyển biến tích cực trong nhiều lĩnh vực của đời sống, đặc biệt là trong ngành thương mại điện tử. Những ứng dụng công nghệ hiện đại không chỉ nâng cao trải nghiệm mua sắm của người tiêu dùng mà còn giúp các doanh nghiệp quản lý hiệu quả hơn, tiết kiệm thời gian và chi phí vận hành.

Trong lĩnh vực thời trang, hình thức mua sắm truyền thống tại cửa hàng vẫn đang được duy trì. Tuy nhiên, phương thức này dần bộc lộ nhiều hạn chế trong thời đại số. Khách hàng phải trực tiếp đến cửa hàng, mất thời gian di chuyển, lựa chọn sản phẩm và chờ đợi thanh toán. Đặc biệt, vào các dịp khuyến mãi hoặc lễ tết, tình trạng chen chúc, quá tải khiến trải nghiệm mua sắm trở nên kém thoải mái. Điều này phần nào ảnh hưởng đến sự hài lòng của khách hàng và khiến họ có xu hướng chuyển sang các nền tảng mua sắm trực tuyến tiện lợi hơn.

Một trong những giải pháp thiết thực nhằm nâng cao trải nghiệm người dùng hiện nay là phát triển các ứng dụng mua sắm thời trang online – nơi người dùng có thể dễ dàng lựa chọn sản phẩm yêu thích, lọc theo nhu cầu cá nhân, xem đánh giá, so sánh giá cả và tiến hành thanh toán chỉ với vài thao tác đơn giản.

Xuất phát từ thực tế đó, ứng dụng bán quần áo Flux Store được phát triển nhằm giúp khách hàng dễ dàng tiếp cận các sản phẩm thời trang một cách nhanh chóng, thuận tiện và hiện đại. Ứng dụng không chỉ cung cấp giao diện thân thiện, hệ thống sản phẩm đa dạng, mà còn hỗ trợ các chức năng như lọc theo danh mục, theo giá, theo đánh giá, tìm kiếm thông minh và thanh toán trực tuyến an toàn.

Thông qua ứng dụng này, chúng tôi mong muốn mang đến một giải pháp thương mại điện tử hiện đại, góp phần nâng cao trải nghiệm mua sắm của người dùng và đồng thời hỗ trợ các cửa hàng thời trang tối ưu hóa hoạt động kinh doanh, mở rộng thị trường trong thời đại số.

1.2. Mục tiêu của đề tài

Đề tài hướng đến việc nghiên cứu, phân tích và xây dựng một “Ứng dụng bán quần áo Flux Store” với giao diện hiện đại, tiện lợi, nhằm tối ưu hóa quy trình mua sắm trực tuyến và nâng cao trải nghiệm của người dùng. Các mục tiêu cụ thể bao gồm:

Tối ưu hóa trải nghiệm mua sắm của người dùng thông qua giao diện thân thiện, dễ sử dụng.

- Hỗ trợ khách hàng tìm kiếm nhanh chóng các sản phẩm theo danh mục hoặc từ khóa.
- Cho phép người dùng lọc sản phẩm theo giá, đánh giá và mức giảm giá.
- Hiển thị đầy đủ thông tin sản phẩm như tên, giá, mô tả, chất liệu, hình ảnh, các biến thể (màu/kích cỡ).
- Người dùng có thể theo dõi trạng thái đơn hàng của mình
- Cung cấp tính năng thêm vào giỏ hàng để mua sau.
- Hỗ trợ người dùng đánh giá sản phẩm và xem phản hồi từ khách hàng khác.

Hỗ trợ thanh toán trực tuyến tiện lợi, nhanh chóng và an toàn.

- Đảm bảo an toàn và bảo mật thông tin cá nhân và thông tin thanh toán của khách hàng.

Phát triển ứng dụng Flux Store không chỉ giúp khách hàng mua sắm dễ dàng mọi lúc, mọi nơi mà còn hỗ trợ doanh nghiệp tối ưu hóa hoạt động bán hàng, mở rộng thị trường và nâng cao năng lực cạnh tranh trong thời đại số hóa.

1.3. Phạm vi của đề tài

Đề tài tập trung vào việc phát triển “Ứng dụng bán quần áo Flux Store” trên nền tảng Android và Website, sử dụng Android Studio làm công cụ phát triển, dùng ngôn ngữ Java, Kotlin và Firebase để lưu trữ và quản lý dữ liệu.

Ứng dụng được thiết kế nhằm phục vụ nhu cầu của khách hàng có nhu cầu mua quần áo của cửa hàng, với các tính năng như:

Yêu cầu chức năng:

- Phân tài khoản:
 - ✓ Người dùng có thể tạo tài khoản để đăng nhập
 - ✓ Người dùng có thể thay đổi thông tin tài khoản

- ✓ Nếu người dùng quên mật khẩu có thể tạo mới mật khẩu thông qua bước xác minh danh tính
- ✓ Người dùng có thể đăng xuất tài khoản
- Phân chủ cửa hàng:
 - ✓ Chủ cửa hàng có thể thêm, sửa thông tin của sản phẩm, xóa sản phẩm trong shop của mình
 - ✓ Chủ cửa hàng xem doanh thu của mình trong 1 khoảng thời gian
 - ✓ Chủ cửa hàng có thể tạo ra các voucher giảm giá
- Nhân viên bán hàng:
 - ✓ Cập nhật trạng thái đơn hàng cho khách (Hủy đơn)
 - ✓ Cập nhật trạng thái đơn hàng
- Nhân viên kho:
 - ✓ Cập nhật số lượng hàng hóa
 - ✓ Cập nhật trạng thái đơn hàng khi đóng hàng thành công
- Phân cho khách hàng:
 - ✓ Khách hàng có thể thêm sản phẩm vào giỏ hàng.
 - ✓ Khách hàng có thể đặt hàng
 - ✓ Khách hàng có thể tìm kiếm sản phẩm
 - ✓ Khách hàng có thể đánh giá sản phẩm
 - ✓ Khách hàng có thể kiểm tra trạng thái đơn hàng

Yêu cầu phi chức năng:

- ✓ Hệ thống phải chạy được trên các máy có Android hệ hệ 7 trở lên.
- ✓ Hệ thống phải đảm bảo tính dễ sử dụng cho người dùng.

1.4 Phân chia nhiệm vụ

STT	Thành viên	Nhiệm vụ
1	Lê Tuấn Hùng	Thiết kế CSDL, quản lý giỏ hàng, checkout đơn hàng, thêm, sửa xóa sản phẩm trong giỏ hàng, vẽ figma, Website
2	Hồ Văn Quang	Thiết kế CSDL, Product: hiển thị sản phẩm, tìm kiếm, lọc theo yêu cầu. Hiển thị category, collection và các sản phẩm tương ứng. Hiển thị thông tin thanh Navigation

3	Đỗ Minh Quân	Hiển thị chi tiết sản phẩm, hiển thị lịch sử đơn hàng, chi tiết đơn hàng, đánh giá sản phẩm
4	Phan Quang Tùng Dương	Đăng nhập xác thực, đăng nhập bằng tài khoản Email, Google. Quản lý khuyến mãi, quản lý Admin Mobile: Hiển thị thông số đơn hàng, số lượng user đăng ký, thống kê doanh thu.

Chương 2. KIẾN TRÚC VÀ CÔNG NGHỆ

2.1. Kiến trúc hệ thống

Hệ thống ứng dụng bán quần áo trên Android được xây dựng theo mô hình **Client - Server**, trong đó:

1. Thiết bị người dùng (Client)

- Là các điện thoại Android cài đặt ứng dụng.
- Giao tiếp với Firebase thông qua Internet để xác thực, truy xuất và cập nhật dữ liệu.
- Giao diện người dùng được thiết kế đơn giản, thân thiện, tập trung vào trải nghiệm mua sắm.

2. Server (Bên thứ ba - Firebase)

Hệ thống sử dụng nền tảng **Firebase** của Google để thay thế cho một backend truyền thống:

- **Firebase Authentication**: Xác thực người dùng (đăng ký, đăng nhập, phân quyền).
- **Firebase Firestore**: Cơ sở dữ liệu chính, lưu trữ thông tin sản phẩm, đơn hàng, người dùng, v.v.

3.

Cloudinary:

Hệ thống dùng dịch vụ lưu trữ và quản lý hình ảnh. Ứng dụng tải ảnh sản phẩm lên Cloudinary và lấy về đường dẫn URL để hiển thị trong ứng dụng.

2.2. Giới thiệu về Công nghệ phát triển

Để xây dựng ứng dụng bán quần áo trên nền tảng Android, nhóm đã sử dụng kết hợp nhiều công nghệ hiện đại, phù hợp với nhu cầu phát triển linh hoạt, hiệu quả:

1. Ngôn ngữ lập trình và nền tảng

- **Java & Kotlin**:

Ứng dụng được phát triển kết hợp cả hai ngôn ngữ:

- **Java** được sử dụng chủ yếu trong phần xử lý logic, quản lý dữ liệu và điều hướng màn hình.
- **Kotlin** được sử dụng cho một số module mới, nơi cần tận dụng cú pháp ngắn gọn, an toàn null và tương thích tốt với các thư viện hiện đại.
- **Android Studio**: Môi trường phát triển chính thức do Google cung cấp, hỗ trợ đầy đủ các công cụ thiết kế giao diện, kiểm thử và biên dịch ứng dụng Android.

2. Kiến trúc phần mềm

- Ứng dụng được tổ chức theo mô hình **MVVM (Model - View - ViewModel)** giúp tách biệt rõ ràng giao diện người dùng (View), logic xử lý (ViewModel) và dữ liệu (Model).
- MVVM giúp tăng khả năng tái sử dụng, dễ bảo trì và mở rộng ứng dụng trong tương lai.

3. Cơ sở dữ liệu và xác thực

- **Firebase Firestore**: Cơ sở dữ liệu NoSQL thời gian thực, được sử dụng để lưu trữ dữ liệu người dùng, sản phẩm, đơn hàng, mã giảm giá...
- **Firebase Authentication**: Cung cấp tính năng đăng ký, đăng nhập, phân quyền và xác thực email người dùng.

4. Quản lý ảnh

- **Cloudinary**: Dịch vụ lưu trữ và xử lý ảnh trên nền tảng đám mây. Ứng dụng sử dụng Cloudinary để tải lên và hiển thị ảnh sản phẩm, giúp tối ưu hiệu suất và dễ dàng thao tác với hình ảnh (resize, crop, lấy URL...).

5. Thư viện bên thứ ba

Trong quá trình phát triển ứng dụng, nhóm đã sử dụng nhiều thư viện mã nguồn mở nhằm tăng hiệu quả lập trình và cải thiện trải nghiệm người dùng, cụ thể như sau:

- **Glide** (com.github.bumptech.glide:glide)
Hỗ trợ tải ảnh từ Internet nhanh chóng, hiệu quả, có cache và xử lý ảnh nền tốt.

- **Firebase SDK**
Bao gồm:
 - firebase-auth: Xác thực người dùng.
 - firebase-firebase: Cơ sở dữ liệu NoSQL.
 - firebase-analytics: Theo dõi hành vi người dùng.
- **OkHttp** (com.squareup.okhttp3:okhttp)
Dùng để gửi HTTP requests (ví dụ: upload ảnh lên Cloudinary qua API trung gian).
- **MPAndroidChart** (com.github.PhilJay:MPAndroidChart)
Hiển thị biểu đồ thống kê (doanh thu, số đơn hàng, v.v.) theo thời gian.
- **DotsIndicator & CircleIndicator**
Hỗ trợ hiển thị thanh trượt (ViewPager) dạng chấm tròn đẹp mắt.
- **Facebook Shimmer**
Hiệu ứng loading mờ mờ lung linh (skeleton loading), tăng tính chuyên nghiệp khi tải dữ liệu.
- **AndroidX Libraries**
Bao gồm:
 - lifecycle-viewmodel-ktx, lifecycle-livedata-ktx: Hỗ trợ ViewModel và LiveData trong mô hình MVVM.
 - appcompat, constraintlayout, gridlayout, core-ktx: Các thành phần cơ bản để xây dựng giao diện và hoạt động ứng dụng.
 - navigation.fragment: Hỗ trợ điều hướng giữa các Fragment.
- **Google Play Services Auth**
Dùng để hỗ trợ đăng nhập bằng Google.

Chương 3. XÂY DỰNG ỨNG DỤNG

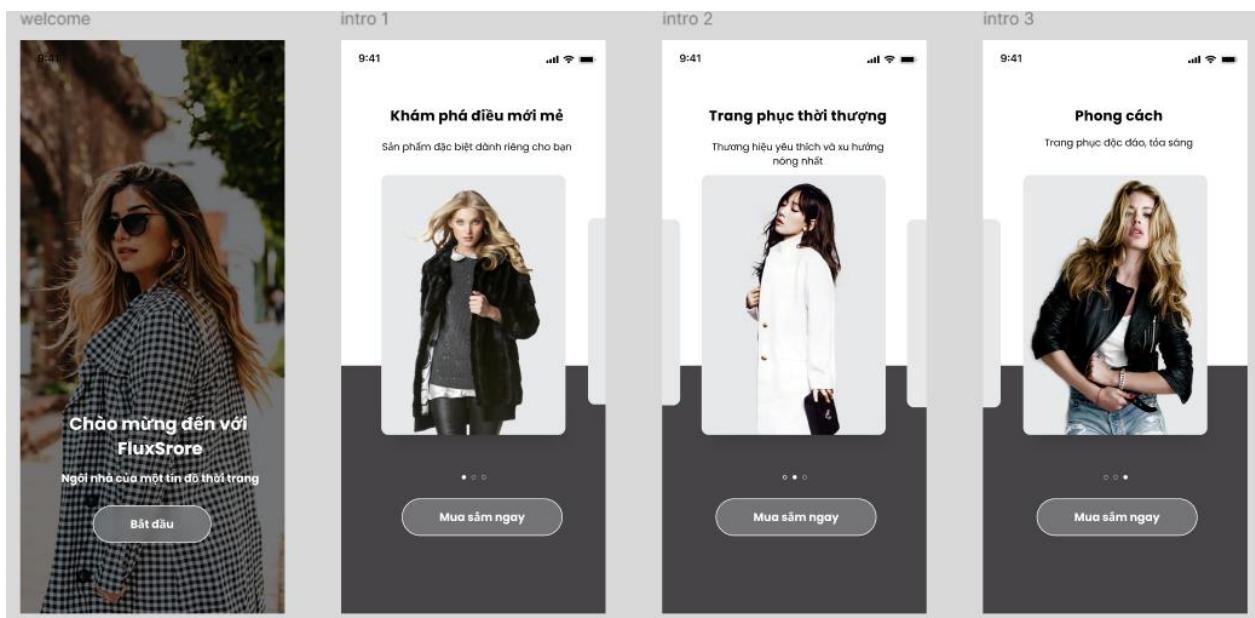
3.1. Thiết kế Figma

- Link GitHub

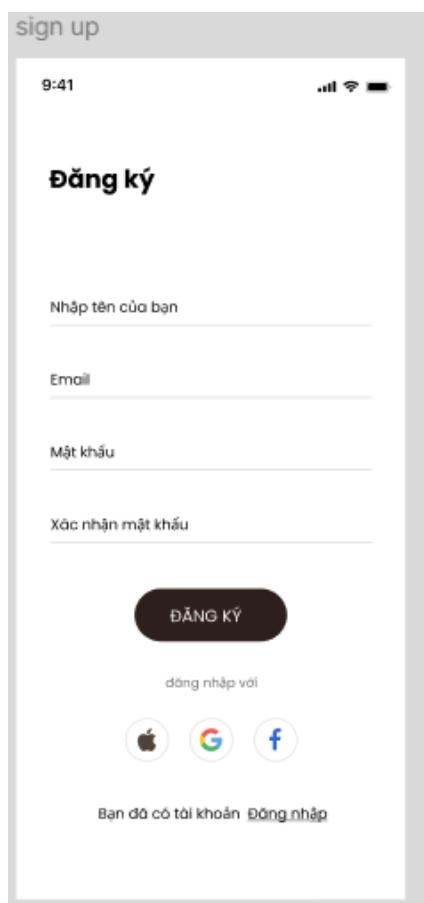
https://github.com/tuanhungg13/clothing_store_app.git

- Ảnh kết xuất Figma các màn hình

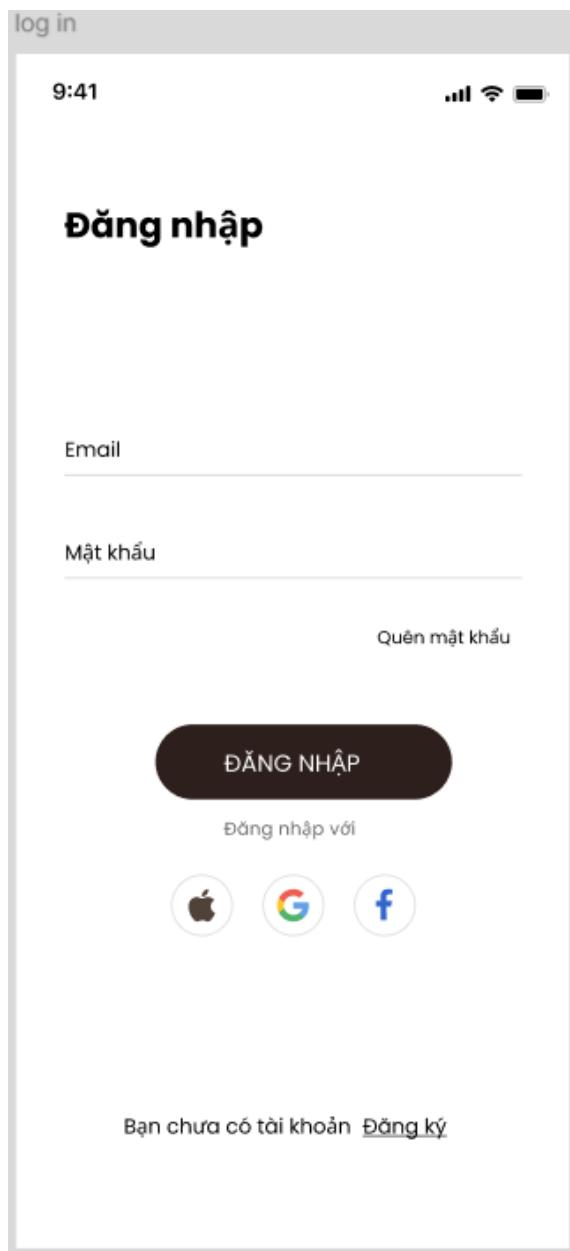
- **Màn hình intro.**



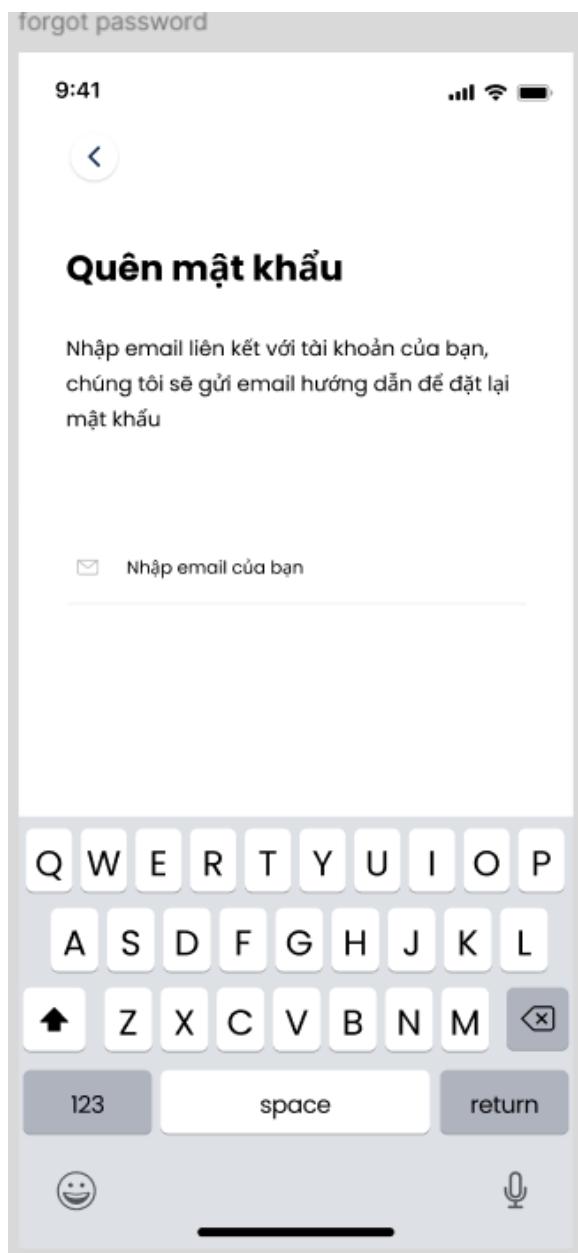
- **Màn hình Đăng ký.**



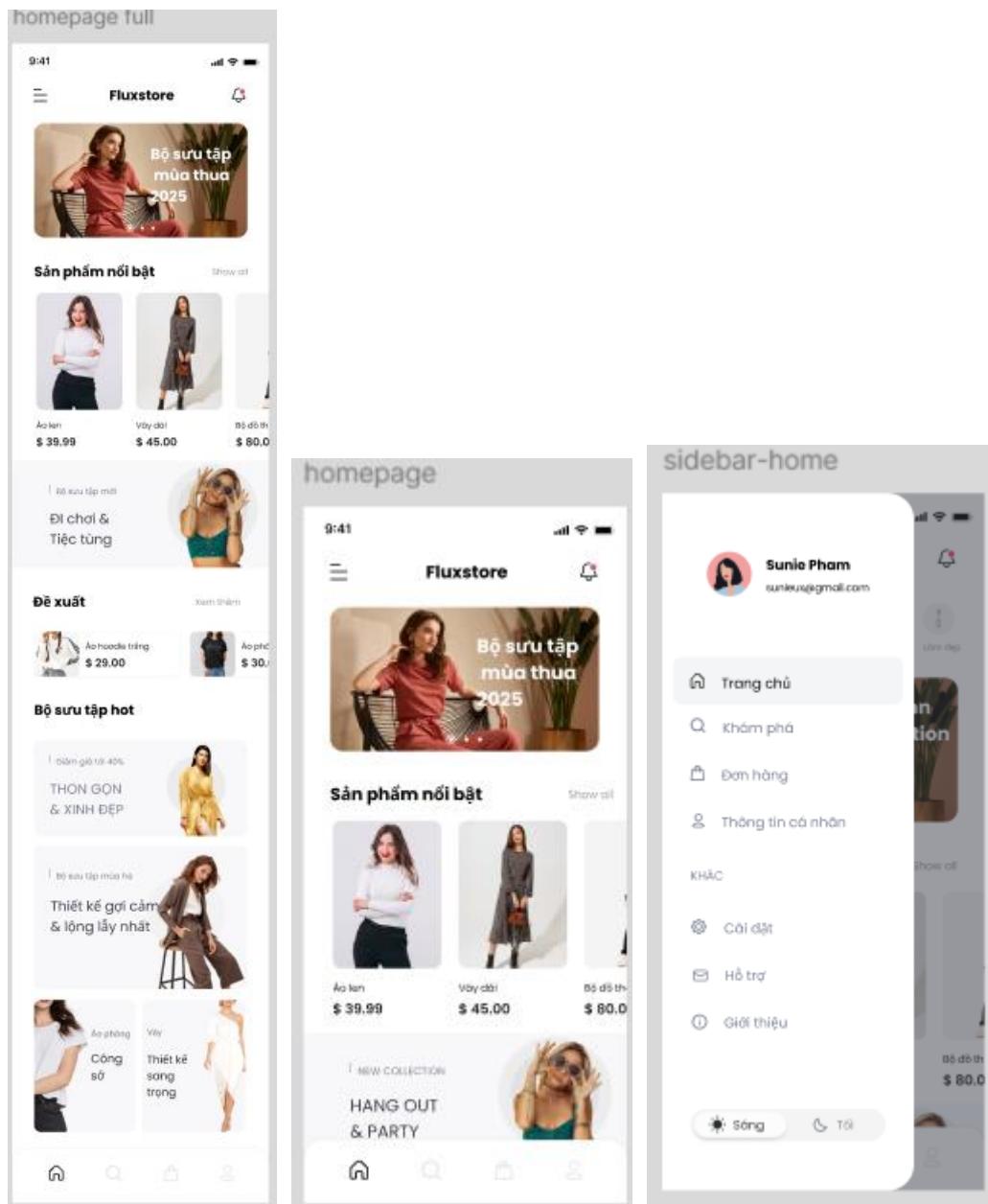
- **Màn hình đăng nhập.**



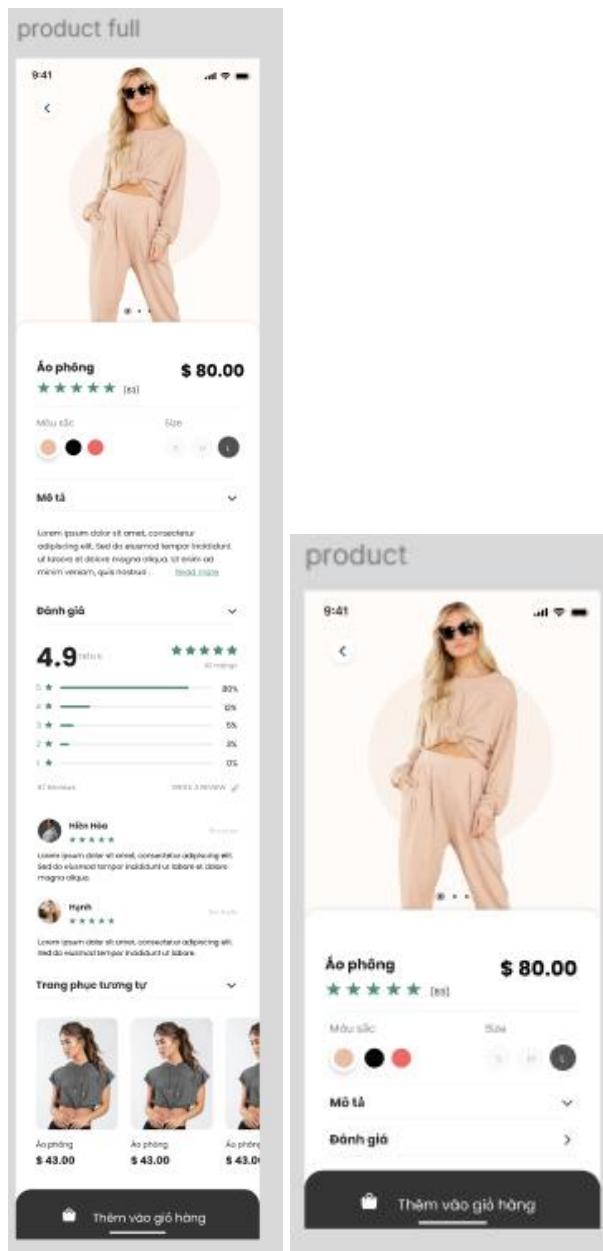
- Màn hình quên mật khẩu.



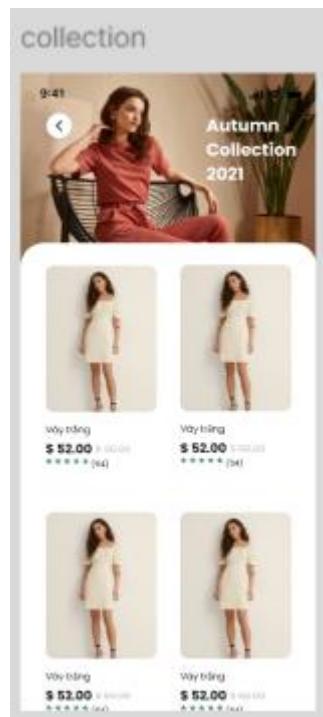
- Màn hình chính.



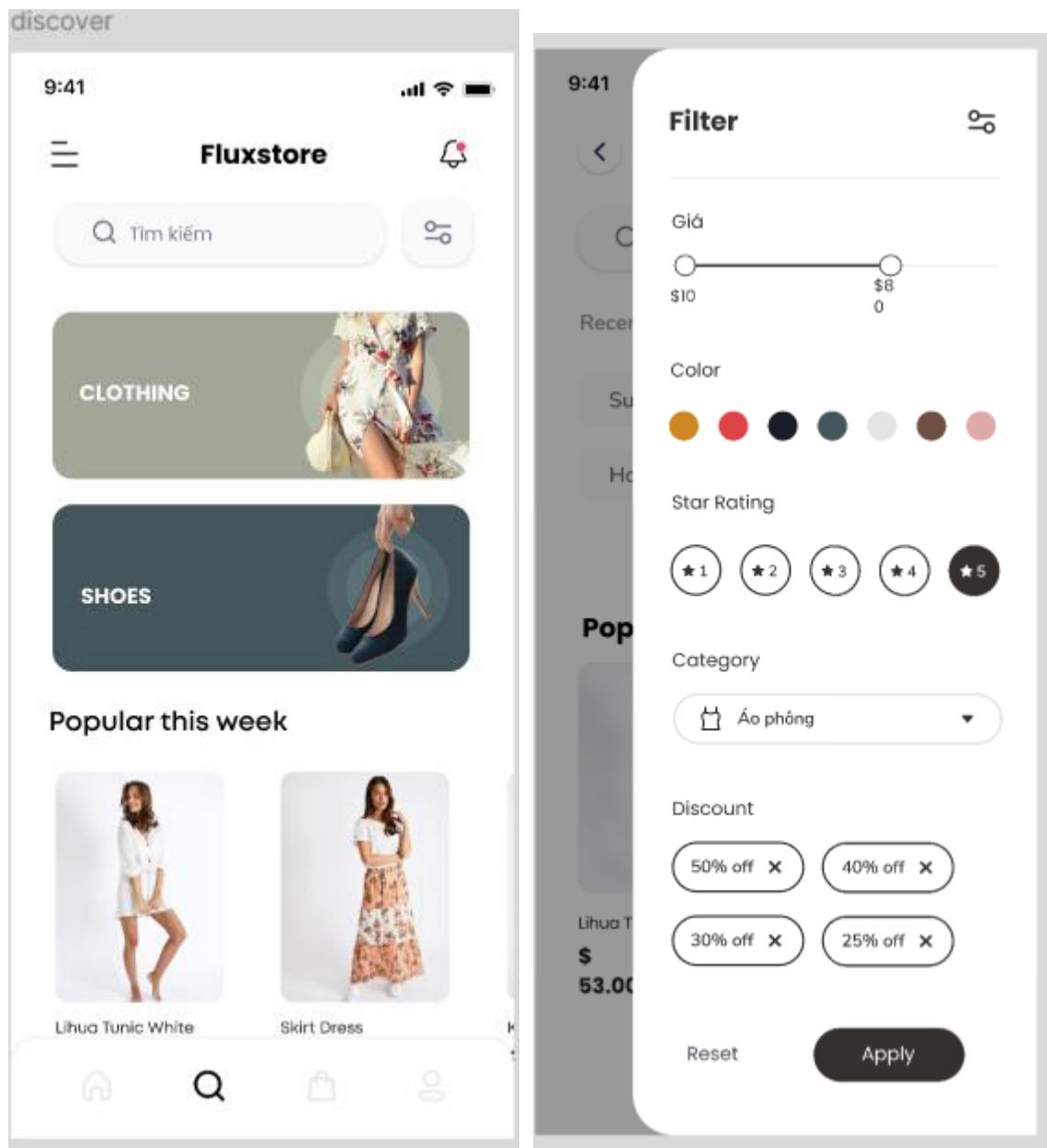
- Màn hình chi tiết sản phẩm.



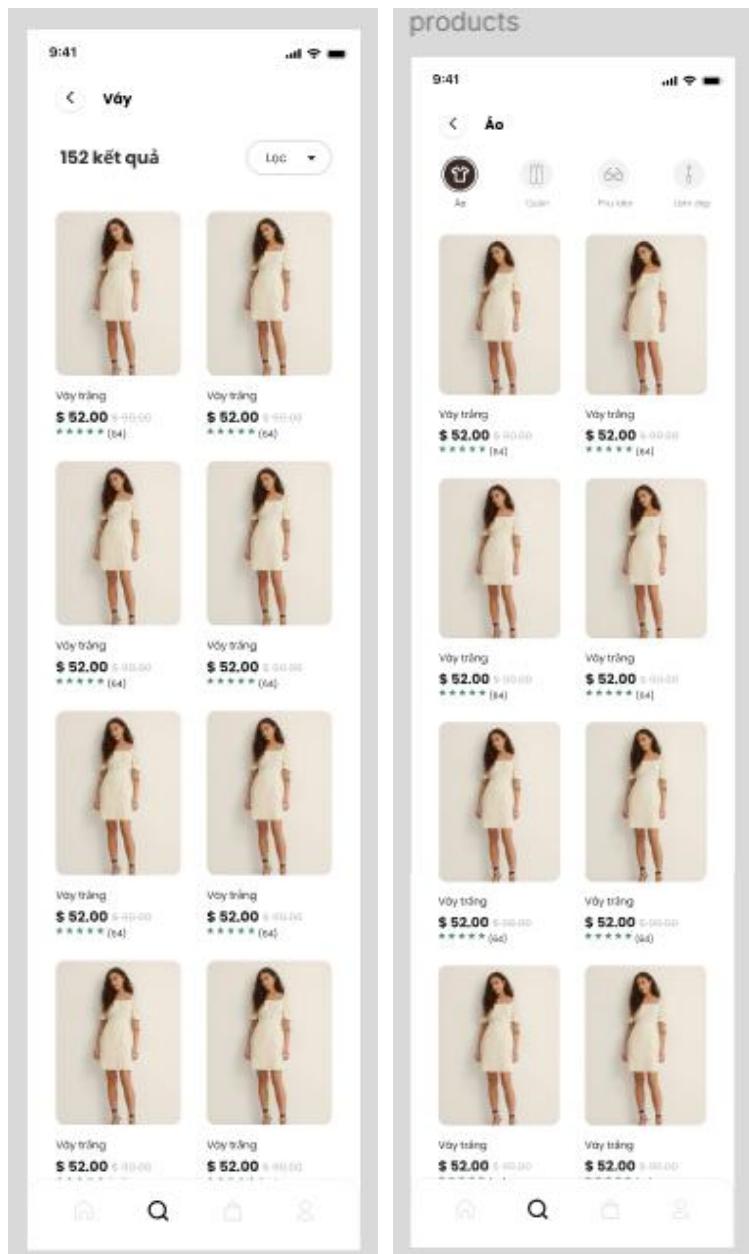
- Màn hình bộ sưu tập.



- **Màn hình tìm kiếm.**



- Màn hình kết quả tìm kiếm.



○ Màn hình đơn hàng.

my orders-1

9:41

Đơn hàng

Chưa xử lý Hoàn tất Hủy

Đơn hàng #123 03/02/2025

 Áo phông
Size: L | Màu sắc: Trắng
\$ 80.00 x1

Quantity: 2 Subtotal: \$110

PENDING Details

Đơn hàng #123 03/02/2025

 Áo phông
Size: L | Màu sắc: Trắng
\$ 80.00 x1

Quantity: 2 Subtotal: \$110

PENDING Details

Đơn hàng #123 03/02/2025

 Áo phông
Size: L | Màu sắc: Trắng
\$ 80.00 x1

Quantity: 2 Subtotal: \$110

SUCCESS Details

Đơn hàng #123 03/02/2025

 Áo phông
Size: L | Màu sắc: Trắng
\$ 80.00 x1

Quantity: 2 Subtotal: \$110

SUCCESS Details

Đơn hàng #123 03/02/2025

 Áo phông
Size: L | Màu sắc: Trắng
\$ 80.00 x1

Quantity: 2 Subtotal: \$110

SUCCESS Details

Đơn hàng #123 03/02/2025

 Áo phông
Size: L | Màu sắc: Trắng
\$ 80.00 x1

Quantity: 2 Subtotal: \$110

SUCCESS Details

Home Search Bag Profile

Left Screen: my orders-3

9:41

Đơn hàng

Chưa xử lý Hoàn tất Hủy

Đơn hàng #123 03/02/2025

Áo phông
Size: L | Mẫu sắc: Trắng
\$ 80.00 x1

Quantity: 2 Subtotal: \$110

CANCEL Details

Đơn hàng #123 03/02/2025

Áo phông
Size: L | Mẫu sắc: Trắng
\$ 80.00 x1

Quantity: 2 Subtotal: \$110

CANCEL Details

Home Search Bag Profile

Right Screen: order info-1

9:41

Đơn hàng #1541

Đơn hàng của bạn đã được giao 

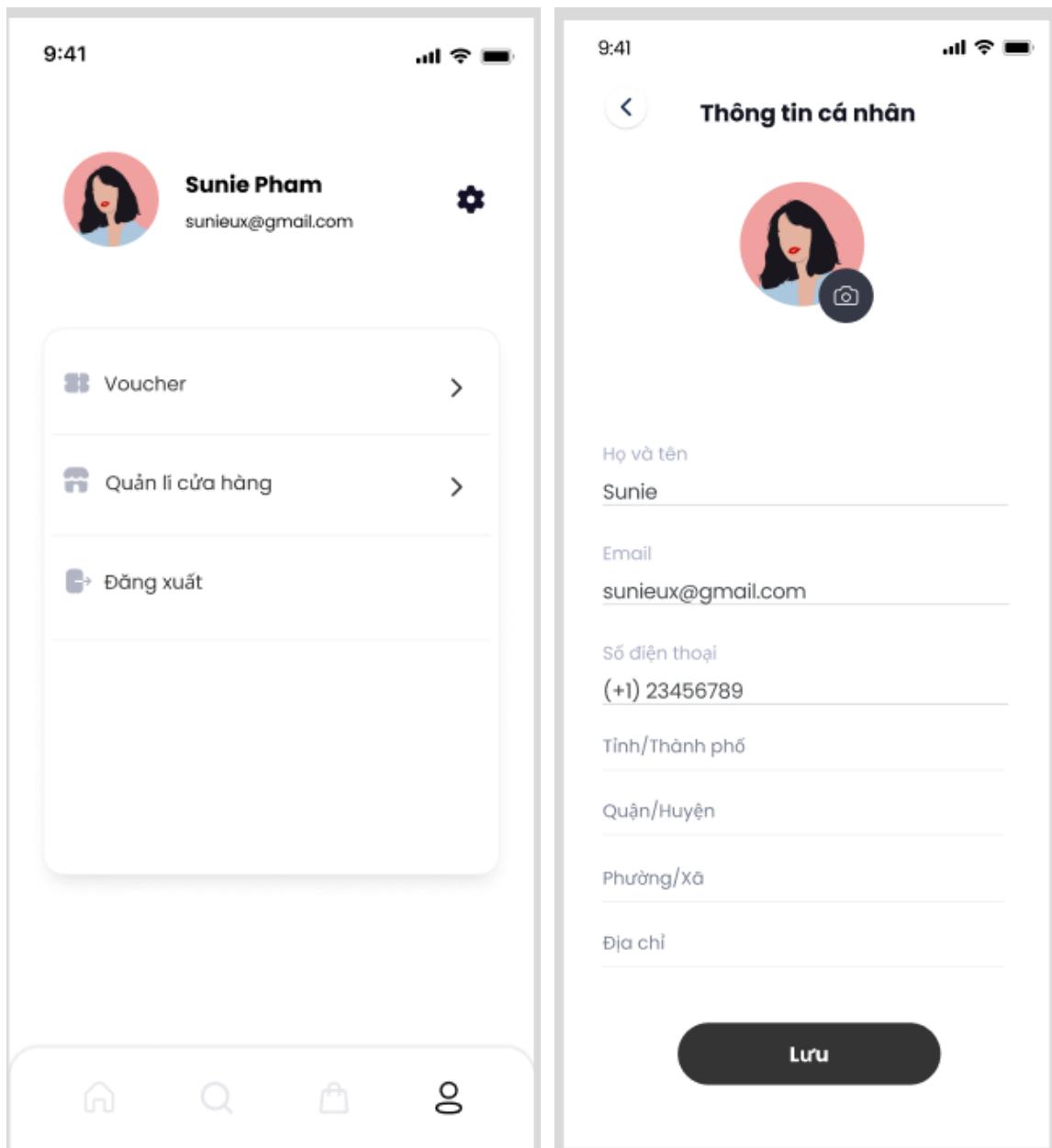
Đừng quên đánh giá sản phẩm để có thể nhận quà

Mã đơn hàng #1514
Mã vận chuyển IK987362341
Địa chỉ giao hàng 175 Tây Sơn, Hà Nội

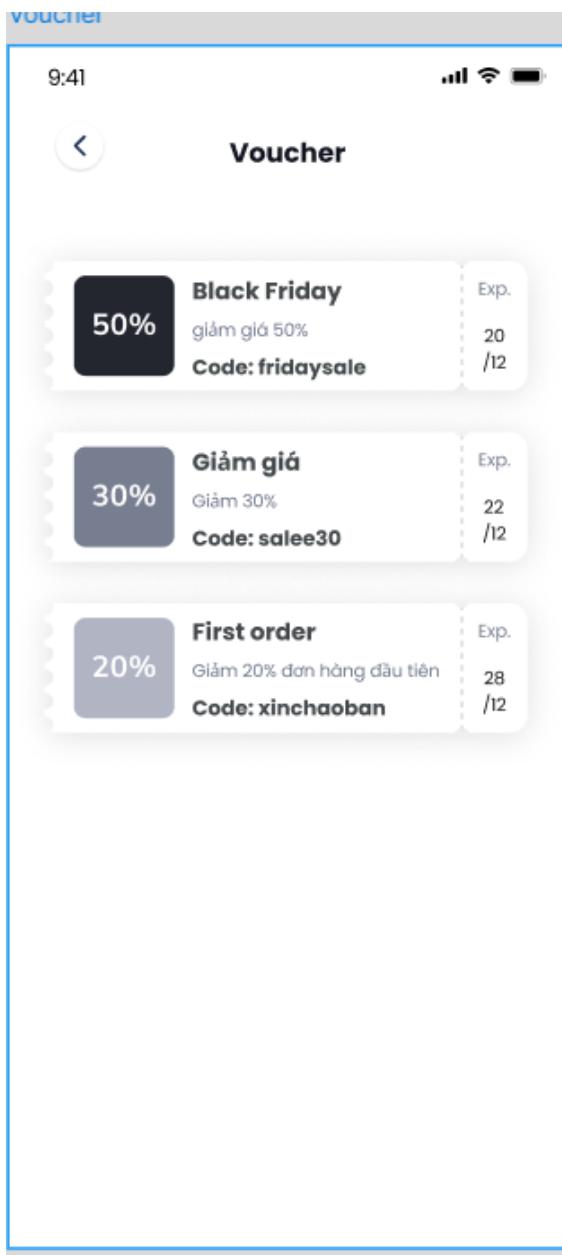
Áo phông	\$68.00	x1
Áo phông	\$68.00	x1
Giá	120.00	
Phí vận chuyển	0.00	
Tổng tiền	\$120.00	

Trang chủ Đánh giá

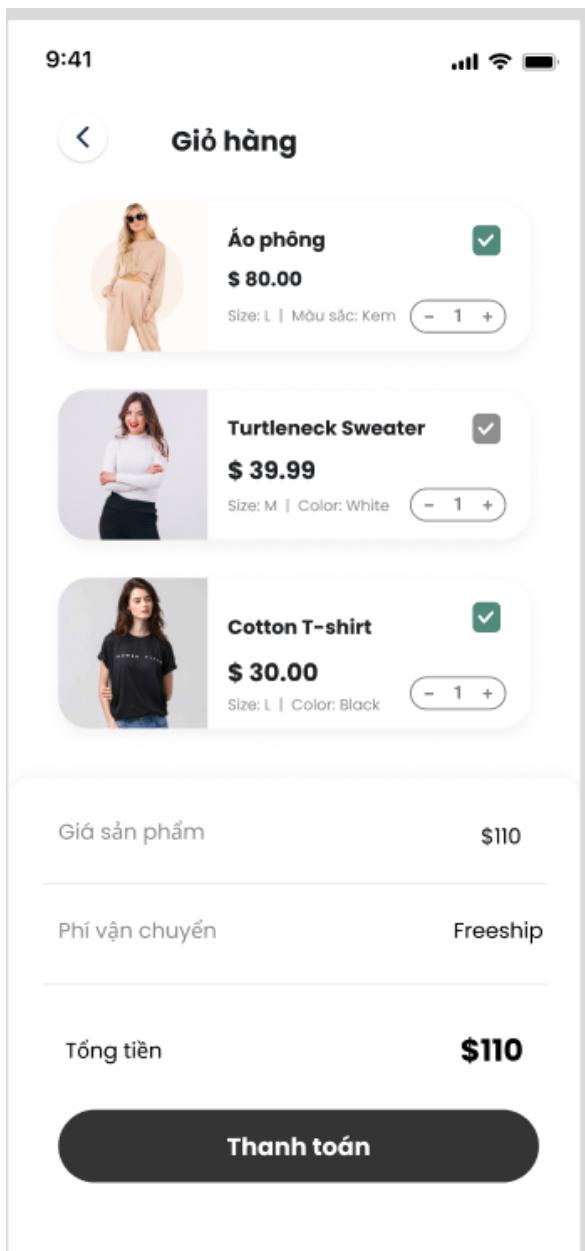
- **Màn hình thông tin cá nhân.**



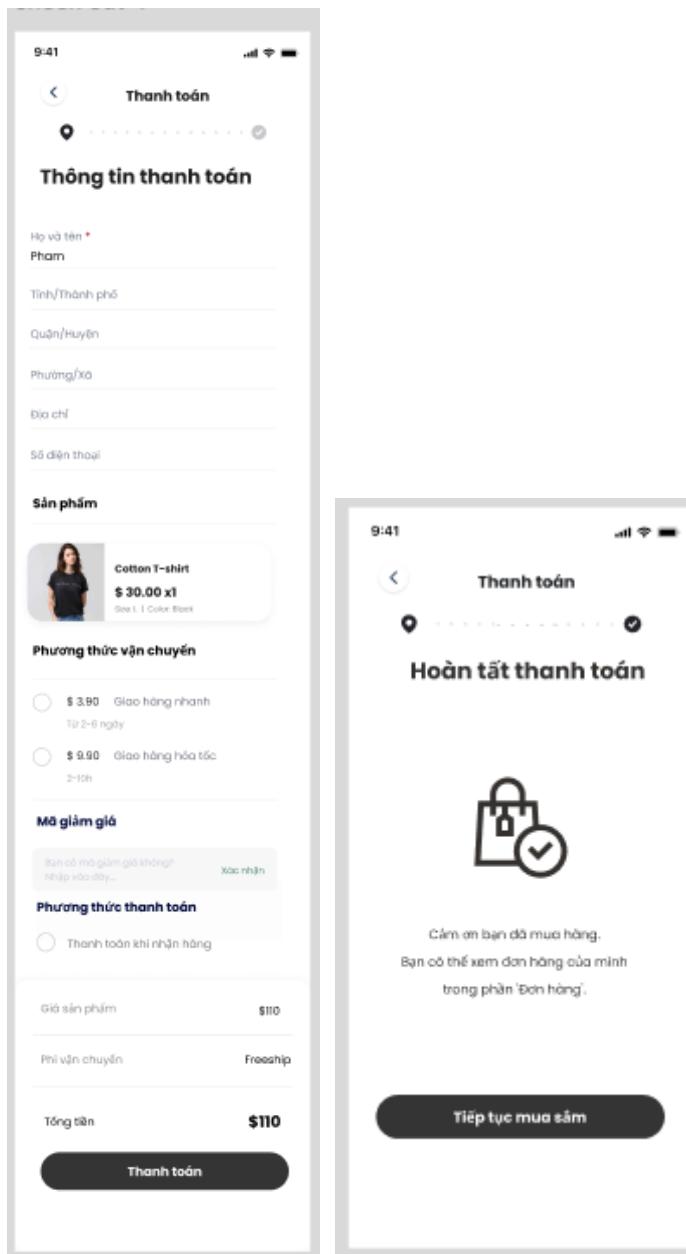
- Màn hình mã giảm giá.



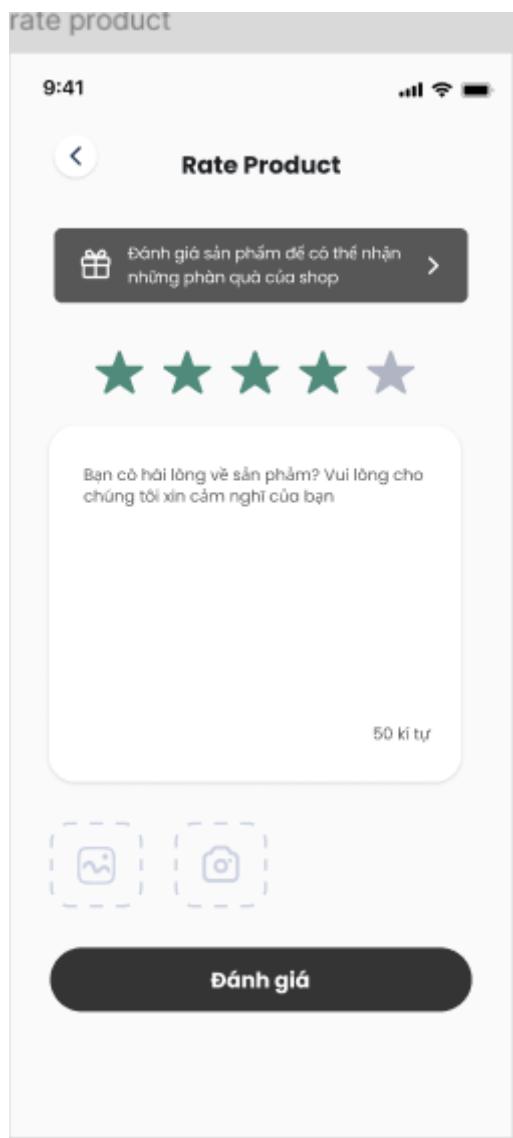
- Màn hình giỏ hàng.



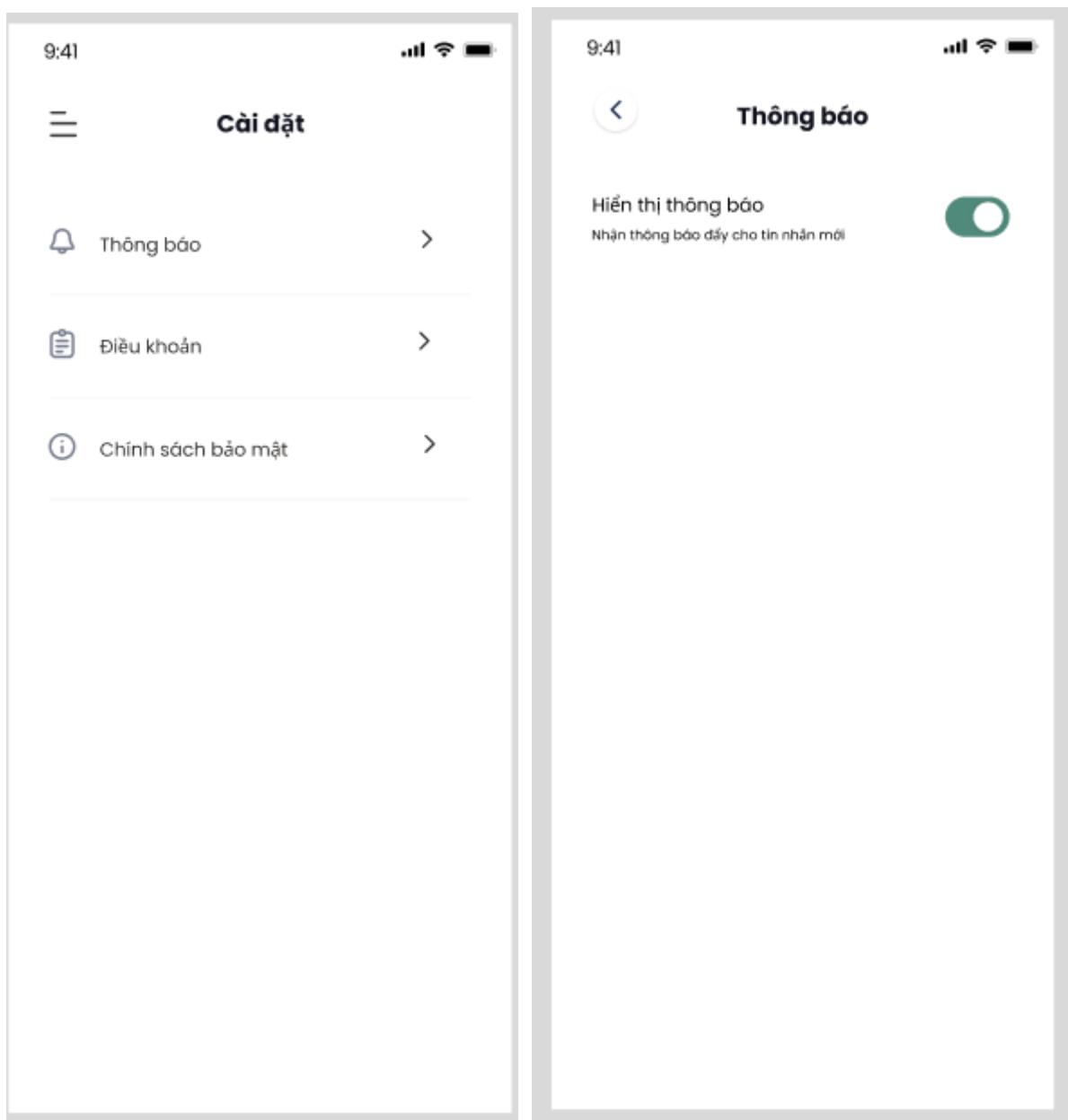
○ **Màn hình thanh toán.**



○ Màn hình đánh giá.



- Màn hình cài đặt.



- Màn hình quản lý cửa hàng.

9:41



Quản lý cửa hàng

Hôm nay

Đơn hàng

30

Người dùng mới

30

Đã vận chuyển

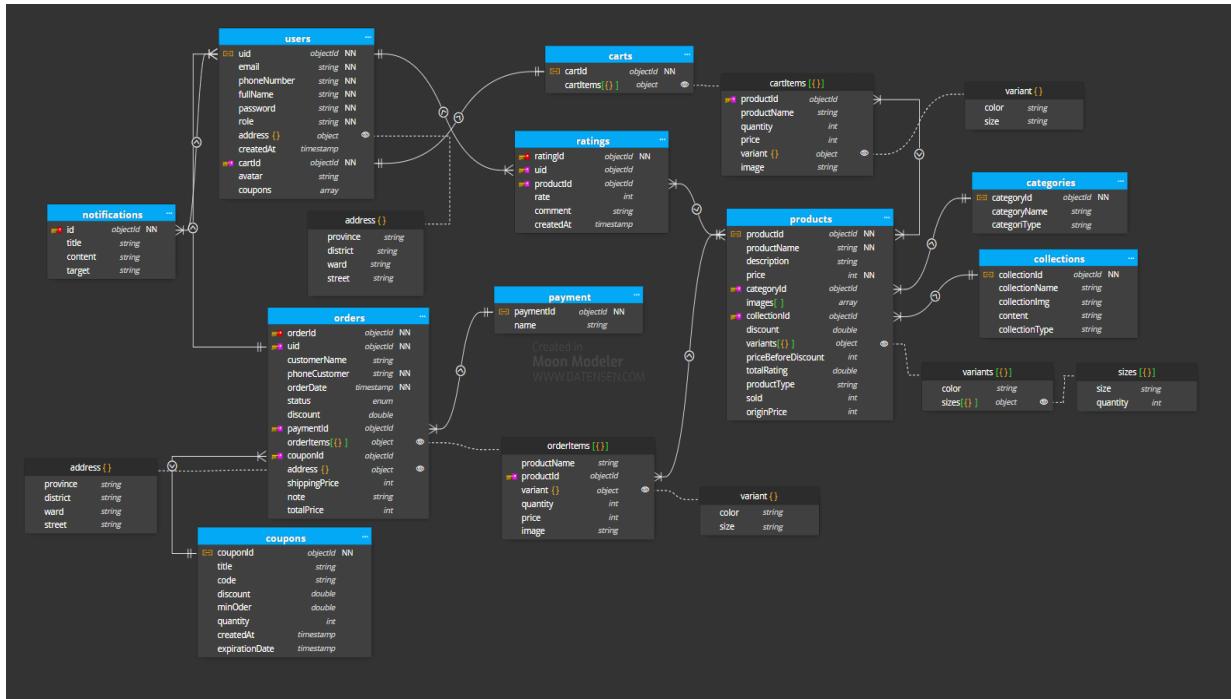
30

Đã đánh giá

30

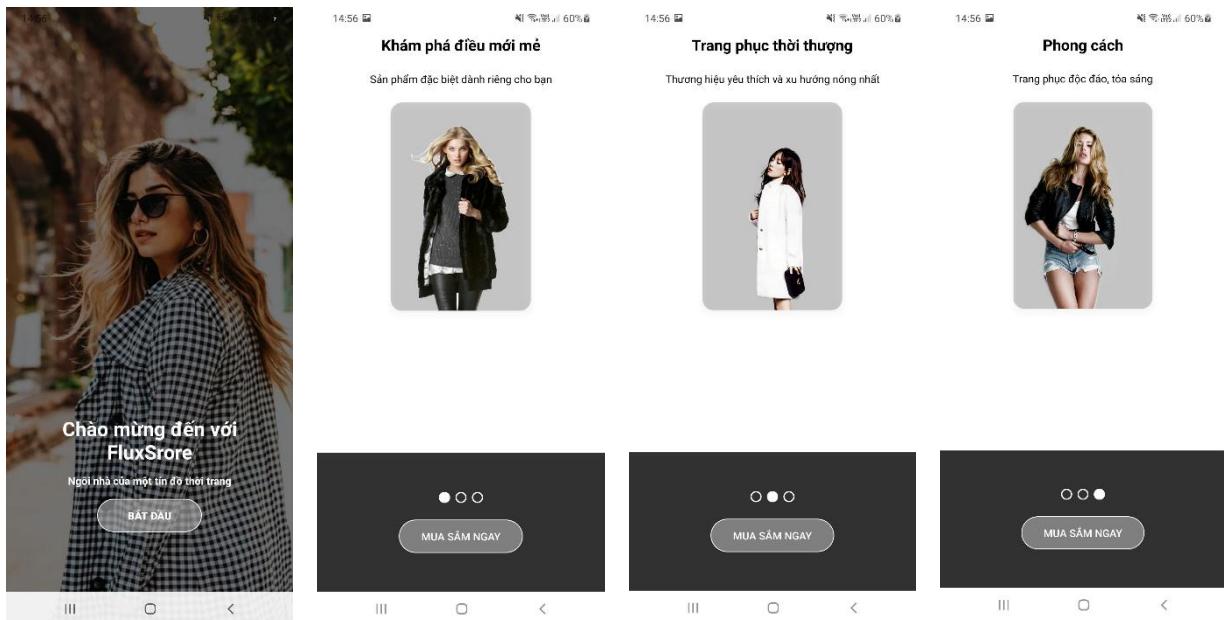


3.2. Thiết kế CSDL

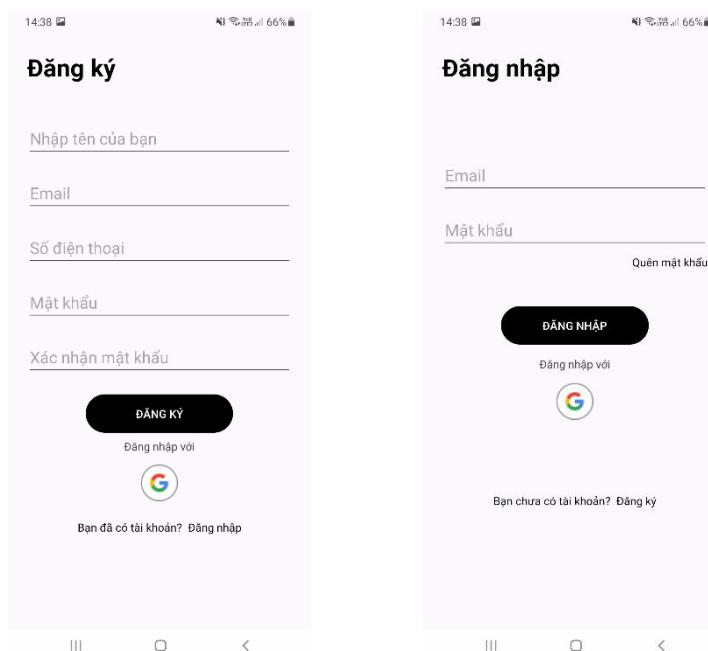


3.3. Giao diện ứng dụng

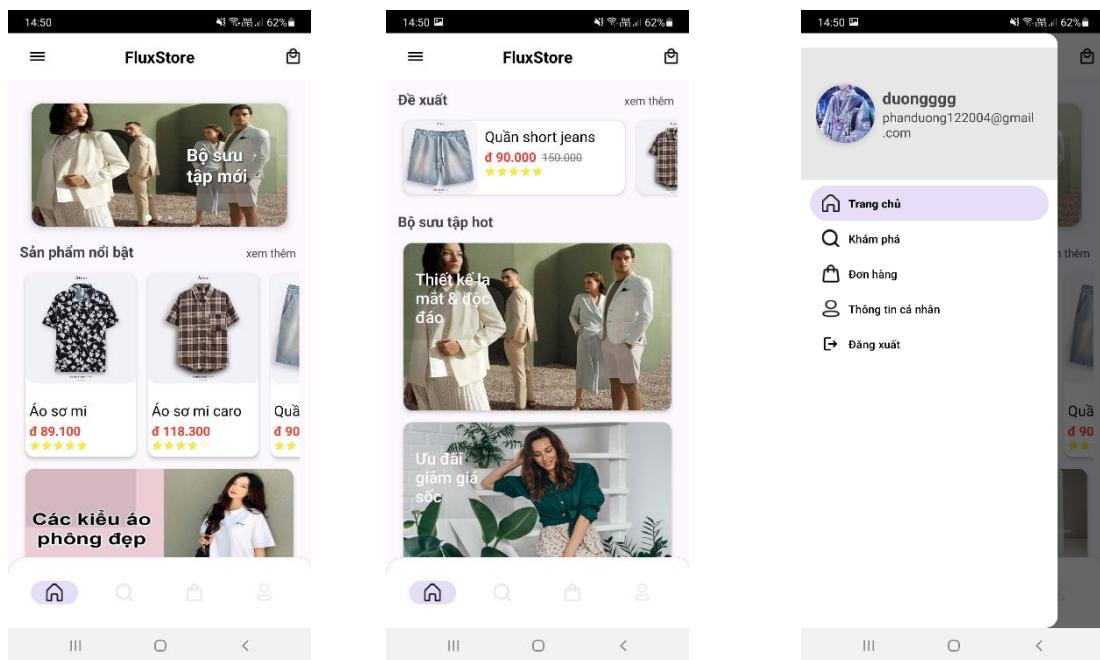
3.3.1. Màn hình giới thiệu



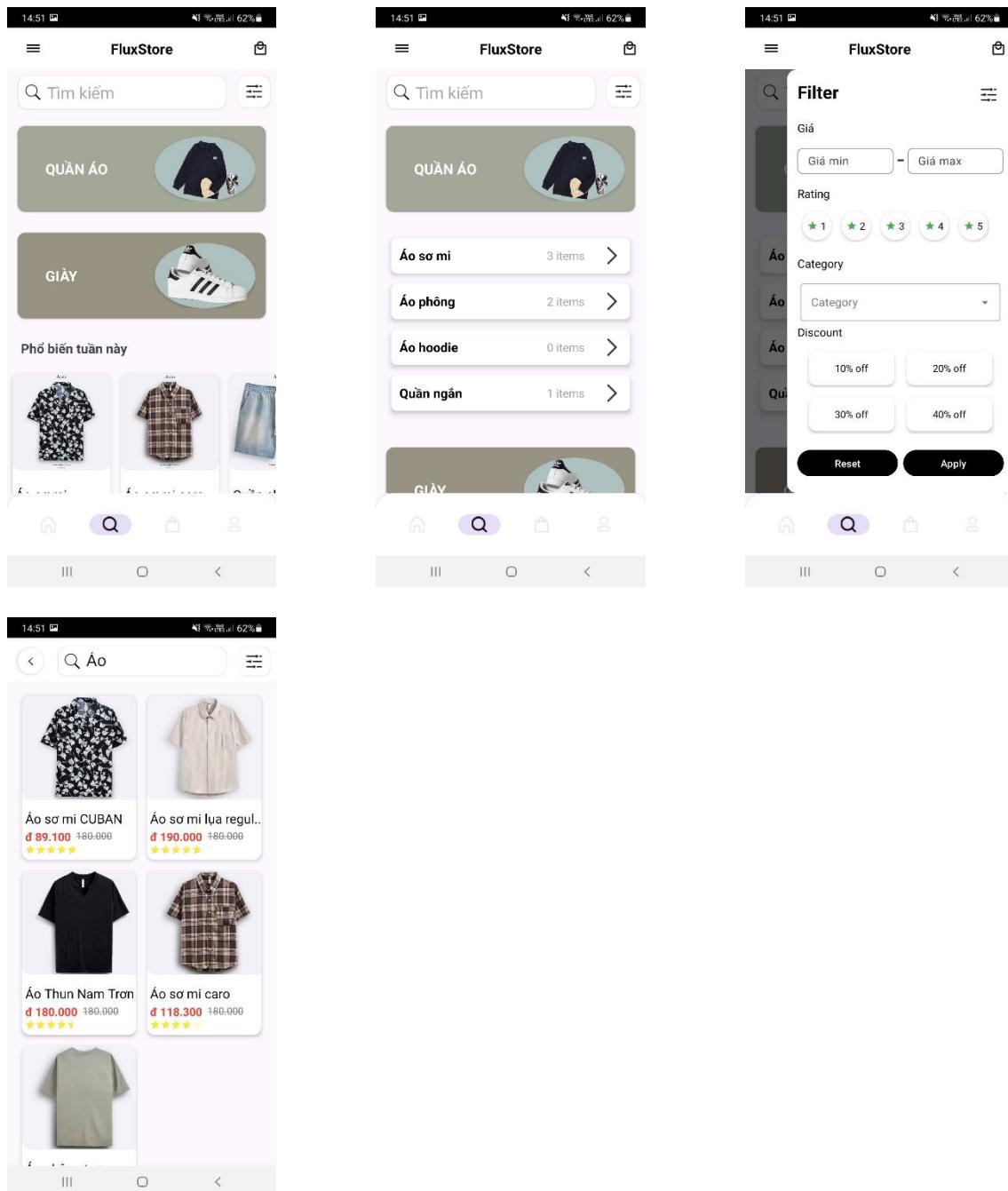
3.3.2. Màn hình đăng nhập & đăng ký



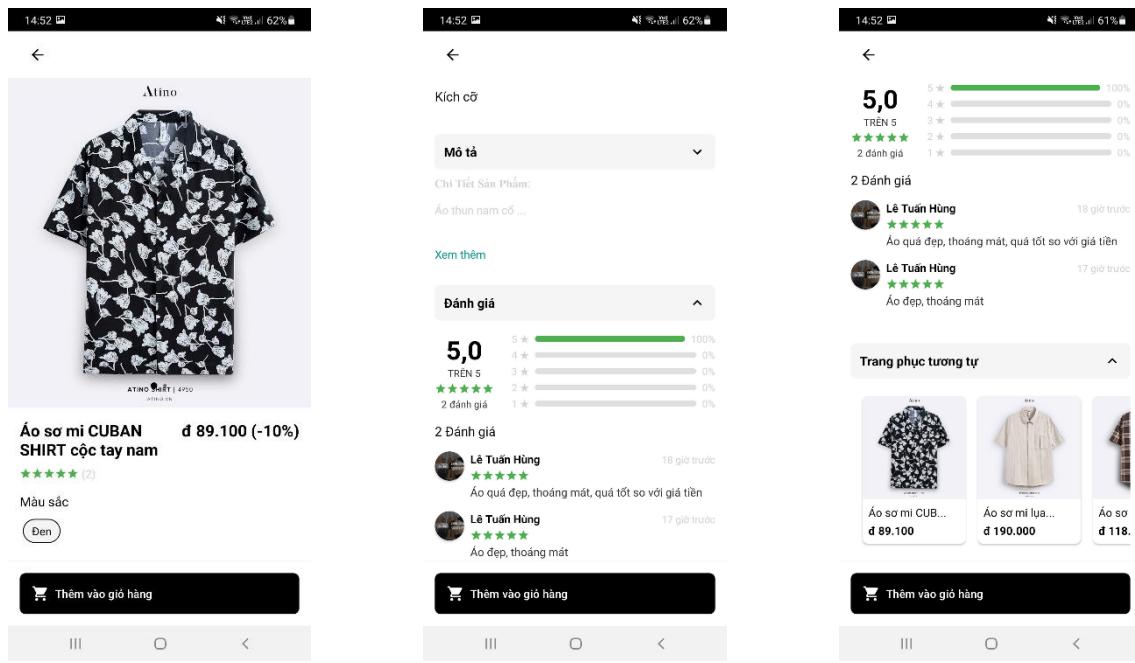
3.3.3. Màn hình trang chủ



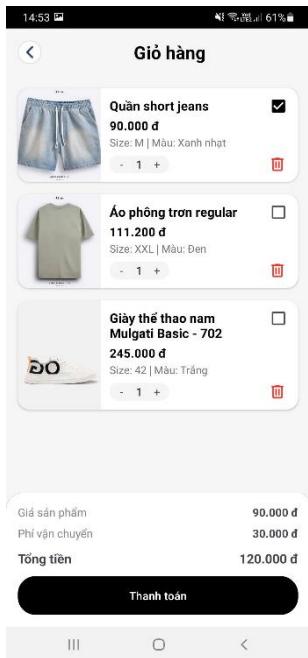
3.3.4. Màn hình tìm kiếm



3.3.5. Màn hình chi tiết sản phẩm



3.3.6. Màn hình giỏ hàng



3.3.7. Màn hình thanh toán



3.3.8. Màn hình lịch sử mua hàng

Đơn hàng # 1619562b-57b7-420e-a0e0-2bc1d35d97ca

Quần short jeans
Kích cỡ: XL | Màu: Xanh nhạt
90.000 đ x5

Số lượng: 1 Tổng tiền: 480.000 đ

Chưa xử lý **Chi tiết**

Đơn hàng # 1830fa90-6a8e-4426-8b27-a674d61cc51e

Áo sơ mi lụa regular
Kích cỡ: L | Màu: Trắng
190.000 đ x1

Số lượng: 1 Tổng tiền: 220.000 đ

Chưa xử lý **Chi tiết**

Đơn hàng # 231ea35e-240d-41de-a70f-b750cb025335

Áo sơ mi lụa regular
Kích cỡ: M | Màu: Rêu
190.000 đ x1

Số lượng: 2 Tổng tiền: 220.000 đ

Hoàn tất **Chi tiết**

Đơn hàng # 761947e-1a87-4264-9a63-c20ca62133d5

Giày thể thao nam Mulgati Basic - 702
Kích cỡ: 39 | Màu: Trắng
245.000 đ x1

Số lượng: 1 Tổng tiền: 245.000 đ

Hoàn tất **Chi tiết**

Đơn hàng # 701db5f-f58e-4a42-8289-3bdcc6b125117

Giày thể thao nam Mulgati Basic - 702
Kích cỡ: 41 | Màu: Trắng
245.000 đ x1

Số lượng: 1 Tổng tiền: 275.000 đ

Hủy **Chi tiết**

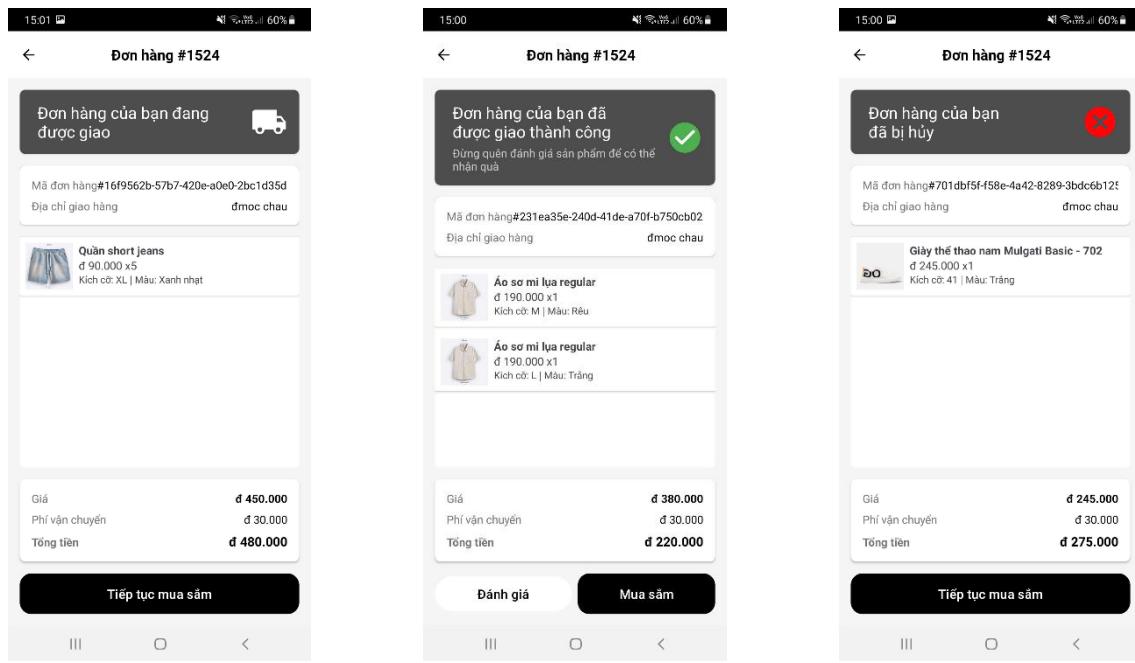
Đơn hàng # b7950fd1-2be5-4ace-be50-0073935b9691

Áo phông trơn regular
Kích cỡ: L | Màu: Rêu
111.200 đ x1

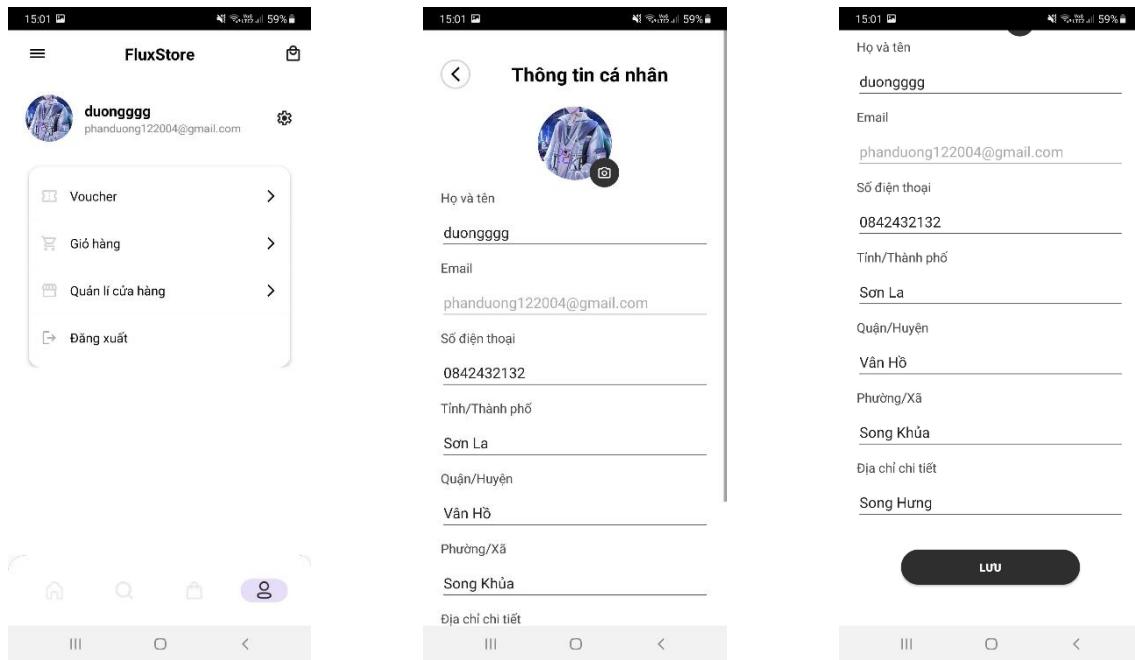
Số lượng: 1 Tổng tiền: 141.200 đ

Hủy **Chi tiết**

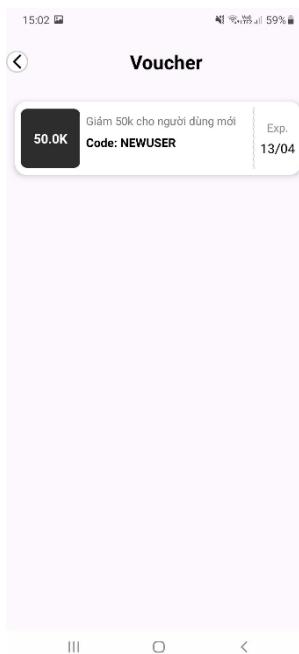
3.3.9. Chi tiết đơn hàng



3.3.10. Màn thông tin người dùng



3.3.11. Màn hình danh sách voucher



3.3.12. Màn hình thống kê



3.4. Code minh họa các chức năng cốt lõi

3.4.1. Chức năng đăng ký.

```

    ⚑ RegisterFragment.java × ⚑ AuthViewModel.java ⚑ AuthService.java ⚑ UserService.java
32  public class RegisterFragment extends Fragment {
    | usage
123  private void handleRegister() {
124      String name = editRegisterName.getText().toString().trim();
125      String email = editRegisterEmail.getText().toString().trim();
126      String phone = editRegisterPhone.getText().toString().trim();
127      String password = editRegisterPassword.getText().toString().trim();
128      String confirmPassword = editConfirmPassword.getText().toString().trim();
129      // validate dữ liệu
130      if (TextUtils.isEmpty(name) || TextUtils.isEmpty(email) ||
131          TextUtils.isEmpty(phone) || TextUtils.isEmpty(password)) {
132          Toast.makeText(getContext(), text: "Vui lòng nhập đầy đủ thông tin", Toast.LENGTH_SHORT).show();
133          return;
134      }
135      if (password.length() < 6) {
136          Toast.makeText(getContext(), text: "Mật khẩu phải từ 6 ký tự trở lên", Toast.LENGTH_SHORT).show();
137          return;
138      }
139      if (!phone.matches(regex: "^(0|\+\d{1,2}\d{9})$")) {
140          editRegisterPhone.setError("Số điện thoại không hợp lệ");
141          editRegisterPhone.requestFocus();
142      }
143      if (!password.equals(confirmPassword)) {
144          Toast.makeText(getContext(), text: "Mật khẩu xác nhận không khớp", Toast.LENGTH_SHORT).show();
145          return;
146      }
147
148      // Lưu lại để truyền sau khi đăng ký thành công
149      registeredEmail = email;
150      registeredPassword = password;
151      //Gửi dữ liệu đến viewmodel để thực hiện đăng ký
152      authViewModel.register(email, password, name, phone);
153  }

```

```

    ⚑ RegisterFragment.java ⚑ AuthViewModel.java × ⚑ AuthService.java ⚑ UserService.java
21  public class AuthViewModel extends ViewModel {
58
59      // ----- ĐĂNG KÝ -----
60      public void register(String email, String password, String fullName, String phoneNumber) {
61          isLoading.setValue(true);
62          authError.setValue(null);
63          //Gọi đến service để lưu dữ liệu
64          AuthService.register(email, password, fullName, phoneNumber, Task<AuthResult> task -> {
65              isLoading.setValue(false);
66              if (task.isSuccessful()) {
67                  currentUid.setValue(AuthService.getCurrentUid());
68              } else {
69                  authError.setValue(new Event<>( content: "Đăng ký thất bại: " + task.getException().getMessage()));
70              }
71          });
72      }
73  }

```

```

    @ RegisterFragment.java      @ AuthViewModel.java      @ AuthService.java      @ UserService.java
21   public class AuthService {
22       1 usage
23       public static void register(String email, String password, String fullName, String phone, OnCompleteListener<AuthResult> callback) {
24           auth.createUserWithEmailAndPassword(email, password)
25               .addOnCompleteListener( Task<AuthResult> task -> {
26                   // Gọi callback để trả kết quả về ViewModel
27                   callback.onComplete(task);
28               }
29               if (task.isSuccessful()) {
30                   String uid = task.getResult().getUser().getUid();
31                   List<String> coupons = new ArrayList<>();
32                   coupons.add("Fr99pt491uAyiyZ4DZjJ");
33                   // Tạo đối tượng User với role "user", address rỗng và cartId null
34                   User newUser = new User(
35                       uid,
36                       email,
37                       phone,
38                       fullName,
39                       role: "user", // role mặc định
40                       null, // Tự động lấy thời gian ở server
41                       new Address(), // address rỗng
42                       null, // cartId sẽ gắn sau khi tạo
43                       coupons
44                   );
45                   // Gọi hàm tạo cart và lưu user
46                   UserService.createUserWithCart(newUser, Task<Void> userTask -> {
47                       if (userTask.isSuccessful()) {
48                           Log.d( tag: "AuthService", msg: "User profile + cart created successfully");
49                       } else {
50                           Log.e( tag: "AuthService", msg: "Failed to create user + cart", userTask.getException());
51                       }
52                   });
53               }
54           );
55       }
56   }

```

```

    @ RegisterFragment.java      @ AuthViewModel.java      @ AuthService.java      @ UserService.java
41   public class UserService {
42       }
43
44       //Tạo cartid và lưu dữ liệu vào firestore
45       2 usages
46       public static void createUserWithCart(User user, OnCompleteListener<Void> onCompleteListener) {
47           cartRef.add(new HashMap<>())
48               .addOnSuccessListener( DocumentReference cartDocRef -> {
49                   String cartId = cartDocRef.getId();
50                   user.setCartId(cartId);
51                   userRef.document(user.getUid()).set(user)
52                       .addOnCompleteListener(onCompleteListener);
53               }
54               .addOnFailureListener( Exception e -> {
55                   onCompleteListener.onComplete(Tasks.forException(e));
56               });
57           }
58       }
59   }

```

3.4.2. Chức năng đăng nhập.

```

    @ LoginFragment.java      @ AuthViewModel.java      @ AuthService.java      @ UserService.java
34   public class LoginFragment extends Fragment {
35       public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
36           }
37
38           // ===== Xử lý đăng nhập =====
39           btnLogin.setOnClickListener( View v -> {
40               String email = editLoginEmail.getText().toString().trim();
41               String password = editLoginPassword.getText().toString().trim();
42
43               if (email.isEmpty() || password.isEmpty()) {
44                   Toast.makeText(getContext(), text: "Vui lòng nhập đầy đủ thông tin", Toast.LENGTH_SHORT).show();
45                   return;
46               }
47               //Gửi dữ liệu đến viewmodel để xử lý
48               authViewModel.login(email, password);
49           });
50       }
51   }

```

```
© LoginFragment.java © AuthViewModel.java x © AuthService.java © UserService.java
21 public class AuthViewModel extends ViewModel {
22     1 usage
40     >     public LiveData<Boolean> getPasswordReset() { return isPasswordReset; }
43
44     // ----- ĐĂNG NHẬP -----
45     1 usage
46     public void login(String email, String password) {
47         isLoading.setValue(true);
48         authError.setValue(null);
49
50         AuthService.login(email, password, Task<AuthResult> task -> {
51             isLoading.setValue(false);
52             if (task.isSuccessful()) {
53                 currentUid.setValue(AuthService.getCurrentUid());
54             } else {
55                 authError.setValue(new Event<>( content: "Đăng nhập thất bại: " + task.getException().getMessage()));
56             }
57         });
58     }
59 }
```

```
© LoginFragment.java © AuthViewModel.java © AuthService.java x © UserService.java
21 public class AuthService {
22
63
64
65     // Đăng nhập
66     1 usage
67     public static void login(String email, String password, OnCompleteListener<AuthResult> callback) {
68         auth.signInWithEmailAndPassword(email, password)
69             .addOnCompleteListener(callback);
70     }
71 }
```

3.4.3. Chức năng đăng nhập bằng Google.

```

    @ LoginFragment.java  @ AuthViewModel.java  @ AuthService.java  @ UserService.java
34     public class LoginFragment extends Fragment {
48         public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
50             btnLogin.setOnClickListener(v -> {
51             });
52
53             // ===== Xử lý đăng nhập bằng Google =====
54             GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
55                 .requestIdToken("641681479947-dtfro9jbcopdu4981hp1je018fr7b3fd.apps.googleusercontent.com") // ID client trong strings.xml từ Firebase
56                 .requestEmail()
57                 .build();
58             GoogleSignInClient googleSignInClient = GoogleSignIn.getClient(requireActivity(), gso);
59
60             // Luôn hiện cửa sổ chọn tài khoản
61             imbLoginGoogle.setOnClickListener(v -> {
62                 googleSignInClient.signIn().addOnCompleteListener(task -> {
63                     Intent signInIntent = googleSignInClient.getSignInIntent();
64                     startActivityForResult(signInIntent, RC_GOOGLE_SIGN_IN);
65                 });
66             });
67
68             // ===== Quan sát UID đăng nhập thành công =====
69             authViewModel.getCurrentUid().observe(getViewLifecycleOwner(), String uid -> {
70                 if (uid != null) {
71                     // Kiểm tra email đã xác thực chưa
72                     FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
73                     if (firebaseUser != null && !firebaseUser.isEmailVerified()) {
74                         Toast.makeText(getApplicationContext(), "Vui lòng xác thực email trước khi đăng nhập", Toast.LENGTH_LONG).show();
75                         FirebaseAuth.getInstance().signOut(); // Đăng xuất nếu email chưa xác thực
76                     } else {
77                         userViewModel.fetchUserInfo(uid); // Tiếp tục lấy thông tin người dùng nếu email đã xác thực
78                     }
79                 }
80             });
81         }
82     }

```

```

    @ LoginFragment.java  @ AuthViewModel.java  @ AuthService.java  @ UserService.java
21     public class AuthViewModel extends ViewModel {
22
23         // ----- ĐĂNG NHẬP GOOGLE -----
24         2 usages
25         public void loginWithGoogle(String idToken) {
26             isLoading.setValue(true);
27             authError.setValue(null);
28
29             AuthCredential credential = GoogleAuthProvider.getCredential(idToken, null);
30             AuthService.loginWithCredential(credential, Task<AuthResult> task -> {
31                 isLoading.setValue(false);
32                 if (task.isSuccessful()) {
33                     FirebaseUser firebaseUser = task.getResult().getUser();
34                     if (firebaseUser != null) {
35                         currentUid.setValue(firebaseUser.getUid());
36
37                         // Nếu user chưa có trong Firestore thì tạo mới
38                         UserService.getUserProfile(firebaseUser.getUid(), Task<DocumentSnapshot> userTask -> {
39                             if (!userTask.isSuccessful() || userTask.getResult() == null || !userTask.getResult().exists()) {
40                                 User newUser = getUser(firebaseUser);
41                                 UserService.createUserWithCart(newUser, Task<Void> createTask -> {
42                                     if (!createTask.isSuccessful()) {
43                                         authError.setValue(new Event<>(content: "Không thể tạo tài khoản mới từ Google: " + createTask.getException().getMessage()));
44                                     }
45                                 });
46                             }
47                         });
48                     }
49                 }
50             } else {
51                 authError.setValue(new Event<>(content: "Đăng nhập Google thất bại: " + task.getException().getMessage()));
52             }
53         });
54     }

```

```
© LoginFragment.java     © AuthViewModel.java     © AuthService.java ×     © UserService.java
21  public class AuthService {
22 >    public static void logout() { auth.signOut(); }
23
24    // Lấy user hiện tại
25 >    public static FirebaseUser getCurrentUser() { return auth.getCurrentUser(); }
26
27    // Kiểm tra đã đăng nhập chưa
28    no usages
29 >    public static boolean isLoggedIn() { return auth.getCurrentUser() != null; }
30
31    // Lấy UID hiện tại
32    2 usages
33 @    public static String getCurrentUid() {
34        FirebaseUser user = auth.getCurrentUser();
35        return user != null ? user.getUid() : null;
36    }
37
38    // Đăng nhập bằng Credential (Google, Facebook...)
39    1 usage
40 >    public static void loginWithCredential(AuthCredential credential, OnCompleteListener<AuthResult> callback) {
41        auth.signInWithCredential(credential)
42            .addOnCompleteListener(callback);
43    }
44
```

```
© LoginFragment.java     © AuthViewModel.java     © AuthService.java ×     © UserService.java
21  public class AuthService {
22 >    public static void logout() { auth.signOut(); }
23
24    // Lấy user hiện tại
25 >    public static FirebaseUser getCurrentUser() { return auth.getCurrentUser(); }
26
27    // Kiểm tra đã đăng nhập chưa
28    no usages
29 >    public static boolean isLoggedIn() { return auth.getCurrentUser() != null; }
30
31    // Lấy UID hiện tại
32    2 usages
33 @    public static String getCurrentUid() {
34        FirebaseUser user = auth.getCurrentUser();
35        return user != null ? user.getUid() : null;
36    }
37
38    // Đăng nhập bằng Credential (Google, Facebook...)
39    1 usage
40 >    public static void loginWithCredential(AuthCredential credential, OnCompleteListener<AuthResult> callback) {
41        auth.signInWithCredential(credential)
42            .addOnCompleteListener(callback);
43    }
44
```

3.4.4. Chức năng giỏ hàng.

+ File: CartActivity:

```
CartActivity.kt x CartService.kt CartViewModel.kt ▲1 ✓18 ⌂ ⌂
24
25 class CartActivity : AppCompatActivity() {
26
27     private lateinit var cartViewModel: CartViewModel
28     private lateinit var cartAdapter: CartAdapter
29     private lateinit var rvCartItems: RecyclerView
30     private lateinit var tvTotal: TextView
31     private lateinit var btnCheckout: TextView
32     private lateinit var tvProductPrice: TextView
33     private lateinit var tvShippingFee: TextView
34     private lateinit var btnBack: ImageView
35     private lateinit var shimmerLayout: View
36     private lateinit var ivEmptyCart: ImageView
37
38     private var cartId: String? = null
39
40     @Suppress("MissingInflatedId")
41     override fun onCreate(savedInstanceState: Bundle?) {
42         super.onCreate(savedInstanceState)
43
44         // 🔒 Kiểm tra đăng nhập bằng FirebaseAuth
45         val currentUser = FirebaseAuth.getInstance().currentUser
46         if (currentUser == null) {
47             val intent = Intent(this, AuthActivity::class.java)
48             intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
49             startActivity(intent)
50             return
51         }
52
53         setContentView(R.layout.activity_cart)
```

```
CartActivity.kt x CartService.kt CartViewModel.kt ▲1 ✓18 ⌂ ⌂
25 class CartActivity : AppCompatActivity() {
26     override fun onCreate(savedInstanceState: Bundle?) {
27
28         val cartService = CartService()
29         val factory = CartViewModelFactory(cartService)
30         cartViewModel = ViewModelProvider(this, factory).get(CartViewModel::class.java)
31         val userViewModel = ViewModelProvider(this)[UserViewModel::class.java]
32         val uid = FirebaseAuth.getInstance().currentUser?.uid
33
34
35         uid?.let {
36             userViewModel.fetchUserInfo(it)
37         }
38
39         userViewModel.currentUser.observe(this) { user ->
40             user?.let {
41                 cartId = it.cartId
42                 cartViewModel.fetchCartItems(it.cartId)
43                 // ↪ Bạn có thể dùng cartId ở đây, ví dụ:
44                 Log.d("CHECKOUT", "Cart ID là: $cartId")
45             }
46         }
47
48     }
49 }
```

```
CartActivity.kt x CartService.kt CartViewModel.kt
25     class CartActivity : AppCompatActivity() {
40         override fun onCreate(savedInstanceState: Bundle?) {
58             ...
88             cartViewModel.cartItems.observe(this, Observer { items ->
89                 shimmerLayout.visibility = View.GONE
90                 rvCartItems.visibility = View.VISIBLE
91
92                 cartId?.let { cid -> // ✅ Đảm bảo cartId non-null
93                     if (items.isEmpty()) {
94                         // Nếu giỏ hàng rỗng, hiển thị thông báo giỏ hàng trống
95                         ivEmptyCart.visibility = View.VISIBLE
96                         rvCartItems.visibility = View.GONE
97                     } else {
98                         ivEmptyCart.visibility = View.GONE
99                         rvCartItems.visibility = View.VISIBLE
100                        cartAdapter = CartAdapter(
101                            items.toMutableList(),
102                            onQuantityChanged = { cartItem, newQuantity ->
103                                cartViewModel.updateItemQuantity(cartItem, newQuantity, cid)
104                            },
105                            onItemRemoved = { cartItem ->
106                                cartViewModel.deleteItemFromCart(cartItem, cid)
107                            },
108                            onSelectionChanged = { cartItem, isSelected ->
109                                cartViewModel.toggleItemSelection(cartItem, isSelected)
110                            }
111                        )
112                        rvCartItems.layoutManager = LinearLayoutManager(this)
113                        rvCartItems.adapter = cartAdapter
114                    }
115                } ?: run {
116                    Toast.makeText(this, "Không tìm thấy Cart ID", Toast.LENGTH_SHORT).show()
117                }
118            }
119        }
120    }
```

```
CartActivity.kt x CartService.kt CartViewModel.kt
25     class CartActivity : AppCompatActivity() {
40         override fun onCreate(savedInstanceState: Bundle?) {
58             ...
121             cartViewModel.totalPrice.observe(this, Observer { total ->
122                 // Cập nhật giá tiền khi có sản phẩm được chọn
123                 tvTotal.text = formatPrice(total)
124             })
125
126             cartViewModel.totalPriceProduct.observe(this) { total ->
127                 // Cập nhật giá tiền khi có sản phẩm được chọn
128                 tvProductPrice.text = formatPrice(total)
129             }
130
131             cartViewModel.shippingFee.observe(this) { shippingFee ->
132                 // Cập nhật phí vận chuyển
133                 tvShippingFee.text = formatPrice(shippingFee)
134             }
135
136             btnCheckout.setOnClickListener {
137                 val selectedItems =
138                     cartViewModel.cartItems.value?.filter { it.isSelected } ?: emptyList()
139
140                 if (selectedItems.isNotEmpty()) {
141                     val intent = Intent(this, CheckoutActivity::class.java)
142                     intent.putParcelableArrayListExtra("selected_items", ArrayList(selectedItems))
143                     startActivity(intent)
144                 } else {
145                     Toast.makeText(this, "Vui lòng chọn sản phẩm để thanh toán", Toast.LENGTH_SHORT)
146                         .show()
147                 }
148             }
149         }
150     }
```

```

class CartViewModelFactory(private val cartService: CartService) : ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(CartViewModel::class.java)) {
            @Suppress("UNCHECKED_CAST")
            return CartViewModel(cartService) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}

```

+ File: CartViewModel:

```

CartActivity.kt   CartService.kt   CartViewModel.kt ×
15  class CartViewModel(private val cartService: CartService) : ViewModel() {
33  // ...
34
35      fun fetchCartItems(cartId: String) {
36          viewModelScope.launch {
37              // ...
38
39              // Giữ lại trạng thái isSelected nếu đã được chọn trước đó
40              val updatedItems = items.map { newItem ->
41                  val isSelected = selectedItems.any {
42                      newItem.productId == it.productId && it.variant == newItem.variant
43                  }
44                  newItem.copy(isSelected = isSelected)
45              }
46
47              _cartItems.value = updatedItems
48
49              // Cập nhật lại selectedItems (dựa theo danh sách mới)
50              selectedItems.clear()
51              selectedItems.addAll(updatedItems.filter { it.isSelected })
52
53              // Tính lại tổng tiền và phí ship
54              _totalPriceProduct.value = calculateSelectedItemsTotalPrice()
55              _totalPrice.value = calculateTotalPrice()
56              _shippingFee.value = if (selectedItems.isNotEmpty()) 30000 else 0
57
58              Log.d("CartViewModel", "Fetched items: $updatedItems")
59          }
60      }
61

```

```

fun calculateSelectedItemsTotalPrice(): Int {
    val selectedItemsList = selectedItems.toList()
    return selectedItemsList.sumBy { it.price * it.quantity }
}

// Tính tổng tiền giỏ hàng
private fun calculateTotalPrice(): Int {
    val total = calculateSelectedItemsTotalPrice()
    return if (selectedItems.isNotEmpty()) total + 30000 else total
}

```

```
fun toggleItemSelection(cartItem: CartItem, isSelected: Boolean) {
    cartItem.isSelected = isSelected // Thêm dòng này

    if (isSelected) {
        if (!selectedItems.any { it.productId == cartItem.productId && it.variant == cartItem.variant }) {
            selectedItems.add(cartItem)
        }
    } else {
        selectedItems.removeIf {
            it.productId == cartItem.productId && it.variant == cartItem.variant
        }
    }
    Log.d("CartViewModel", "Selected items: $selectedItems")
    val totalProduct = calculateSelectedItemsTotalPrice()
    val totalPrice = calculateTotalPrice()
    _totalPriceProduct.value = totalProduct
    _totalPrice.value = totalPrice
    _shippingFee.value = if (selectedItems.isNotEmpty()) 30000 else 0
}
```

```
fun deleteItemFromCart(cartItem: CartItem, cartId: String) {
    viewModelScope.launch {
        val success = cartService.removeItemFromCart(cartItem, cartId)
        if (success) {
            fetchCartItems(cartId) // cập nhật lại danh sách
        } else {
            Toast.makeText(
                null,
                "Xóa sản phẩm thất bại!",
                Toast.LENGTH_SHORT
            ).show()
        }
    }
}

fun deleteMultipleItemsFromCart(cartItems: List<CartItem>, cartId: String) {
    viewModelScope.launch {
        val success = cartService.removeMultipleItemsFromCart(cartItems, cartId)
        if (success) {
            fetchCartItems(cartId)
        }
    }
}
```

+ File: CartService:

```
CartActivity.kt CartService.kt x CartViewModel.kt
9  class CartService {
12
13      // Lấy dữ liệu giỏ hàng từ Firebase
14      suspend fun fetchCartItems(cartId: String): List<CartItem> {
15          return try {
16              val documentSnapshot = firestore.collection("carts")
17                  .document(cartId)
18                  .get()
19                  .await()
20
21              if (!documentSnapshot.exists()) {
22                  Log.w("Firestore", "Cart document not found with ID: $cartId")
23                  return emptyList()
24              }
25
26              val cart = documentSnapshot.toObject(Cart::class.java)
27              val cartItems = cart?.cartItems ?: return emptyList()
28
29              // Duyệt từng item để lấy stock tương ứng
30              return cartItems.map { item ->
31                  val productSnapshot = firestore.collection("products")
32                      .document(item.productId)
33                      .get()
34                      .await()
35
36                  val variants = productSnapshot["variants"] as? List<Map<String, Any>> ?: emptyList()
37
38                  val matchedVariant = variants.find {
39                      (it["color"] as? String)?.trim()?.lowercase() == item.variant.color.trim().lowercase()
40                  }
41
42          }
43
44      }
45
46      // Cập nhật số lượng sản phẩm
47      suspend fun updateItemQuantity(cartItem: CartItem, cartId: String): Boolean = runCatching {
48          val cartRef = firestore.collection("carts").document(cartId)
49          val currentItems =
50              cartRef.get().await().get("cartItems") as? List<Map<String, Any>> ?: return false
51          val updatedItems = currentItems.map { item ->
52              if (
53                  item["productId"] == cartItem.productId &&
54                  (item["variant"] as? Map<*, *>)?.>.get("color") == cartItem.variant.color &&
55                  (item["variant"] as? Map<*, *>)?.>.get("size") == cartItem.variant.size
56              ) {
57                  item.toMutableMap().apply {
58                      this["quantity"] = cartItem.quantity
59                      this.remove("stock") // ✅ Xoá thuộc tính stock nếu có
60                  }
61              } else {
62                  item.toMutableMap().apply {
63                      this.remove("stock") // ✅ Cần thận xoá luôn ở các item còn lại nếu lỡ có thêm
64                  }
65              }
66          }
67
68          cartRef.update("cartItems", updatedItems).await()
69          true
70      }.getOrDefault {
71          it.printStackTrace()
72          false
73      }
74
75  }
```

```
CartActivity.kt CartService.kt x CartViewModel.kt
9  class CartService {
12
13      // Cập nhật số lượng sản phẩm
14      suspend fun updateItemQuantity(cartItem: CartItem, cartId: String): Boolean = runCatching {
15          val cartRef = firestore.collection("carts").document(cartId)
16          val currentItems =
17              cartRef.get().await().get("cartItems") as? List<Map<String, Any>> ?: return false
18          val updatedItems = currentItems.map { item ->
19              if (
20                  item["productId"] == cartItem.productId &&
21                  (item["variant"] as? Map<*, *>)?.>.get("color") == cartItem.variant.color &&
22                  (item["variant"] as? Map<*, *>)?.>.get("size") == cartItem.variant.size
23              ) {
24                  item.toMutableMap().apply {
25                      this["quantity"] = cartItem.quantity
26                      this.remove("stock") // ✅ Xoá thuộc tính stock nếu có
27                  }
28              } else {
29                  item.toMutableMap().apply {
30                      this.remove("stock") // ✅ Cần thận xoá luôn ở các item còn lại nếu lỡ có thêm
31                  }
32              }
33          }
34
35          cartRef.update("cartItems", updatedItems).await()
36          true
37      }.getOrDefault {
38          it.printStackTrace()
39          false
40      }
41
42  }
```

```
CartActivity.kt    CartService.kt    CartViewModel.kt
9  class CartService {
64    suspend fun updateItemQuantity(cartItem: CartItem, cartId: String): Boolean = runCatching {
90      false
91    }
92
93    suspend fun removeItemFromCart(cartItem: CartItem, cartId: String): Boolean = runCatching {
94      val cartRef = firestore.collection("carts").document(cartId)
95      val currentItems =
96        cartRef.get().await().get("cartItems") as? List<Map<String, Any>> ?: return false
97
98      val updatedItems = currentItems.filterNot { item ->
99        item["productId"] == cartItem.productId &&
100       (item["variant"] as? Map<*, *>)?.>.get("color") == cartItem.variant.color &&
101       (item["variant"] as? Map<*, *>)?.>.get("size") == cartItem.variant.size
102    }
103
104    cartRef.update("cartItems", updatedItems).await()
105    true
106  }.getOrDefault {
107    it.printStackTrace()
108    false
109  }
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
159
160
161
162
163
164
165
166
167
168
169
169
170
171
172
173
174
175
176
177
178
179
179
180
181
182
183
184
185
186
187
188
189
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
209
210
211
212
213
214
215
216
217
218
219
219
220
221
222
223
224
225
226
227
228
229
229
230
231
232
233
234
235
236
237
238
239
239
240
241
242
243
244
245
246
247
248
249
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1290
1291
1292
1293
1294
1295
1295
1296
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1395
1396
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1490
1491
1492
1493
1494
1495
1495
1496
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1590
1591
1592
1593
1594
1595
1595
1596
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1690
1691
1692
1693
1694
1695
1695
1696
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1790
1791
1792
1793
1794
1795
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1890
1891
1892
1893
1894
1895
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1990
1991
1992
1993
1994
1995
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2090
2091
2092
2093
2094
2095
2095
2096
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2188
2189
2190
2191
2192
2193
2194
2195
2195
2196
2197
2198
2199
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2238
2239
2240
```

```

// 1 Kiểm tra tồn kho và cập nhật variants
order.orderItems.forEach { item ->
    val productRef = db.collection("products").document(item.productId)
    val snapshot = transaction.get(productRef)
    Log.d("TAG", "Đang kiểm tra tồn kho cho sản phẩm: ${item.variant}")
    val variants = snapshot.get("variants") as? List<Map<String, Any>>
        ?: throw FirebaseFirestoreException(
            "Sản phẩm không có biến thể!",
            FirebaseFirestoreException.Code.ABORTED
        )
}

```

```

object CheckoutService {
    fun submitOrder(
        db: FirebaseFirestore
    ) {
        db.runTransaction { transaction ->
            order.orderItems.forEach { item ->
                val updatedVariants = variants.map { variant ->
                    val color = variant["color"] as? String
                    val sizes = variant["sizes"] as? List<Map<String, Any>> ?: emptyList()

                    if (color == item.variant.color) {
                        val updatedSizes = sizes.map { sizeMap ->
                            val sizeName = sizeMap["size"] as? String
                            if (sizeName == item.variant.size) {
                                val currentQty = (sizeMap["quantity"] as? Long)? .toInt() ?: 0

                                if (currentQty < item.quantity) {
                                    throw FirebaseFirestoreException(
                                        "${item.productName} (${item.variant.color} - ${item.variant.size}) không đủ hàng",
                                        FirebaseFirestoreException.Code.ABORTED
                                    )
                                }
                            }
                            sizeMap.toMutableMap().apply {
                                this["quantity"] = (currentQty - item.quantity).toLong()
                            }
                        } else sizeMap
                    }
                    variant.toMutableMap().apply {
                        this["sizes"] = updatedSizes
                    }
                } else variant
            }
        }
    }
}

```

```

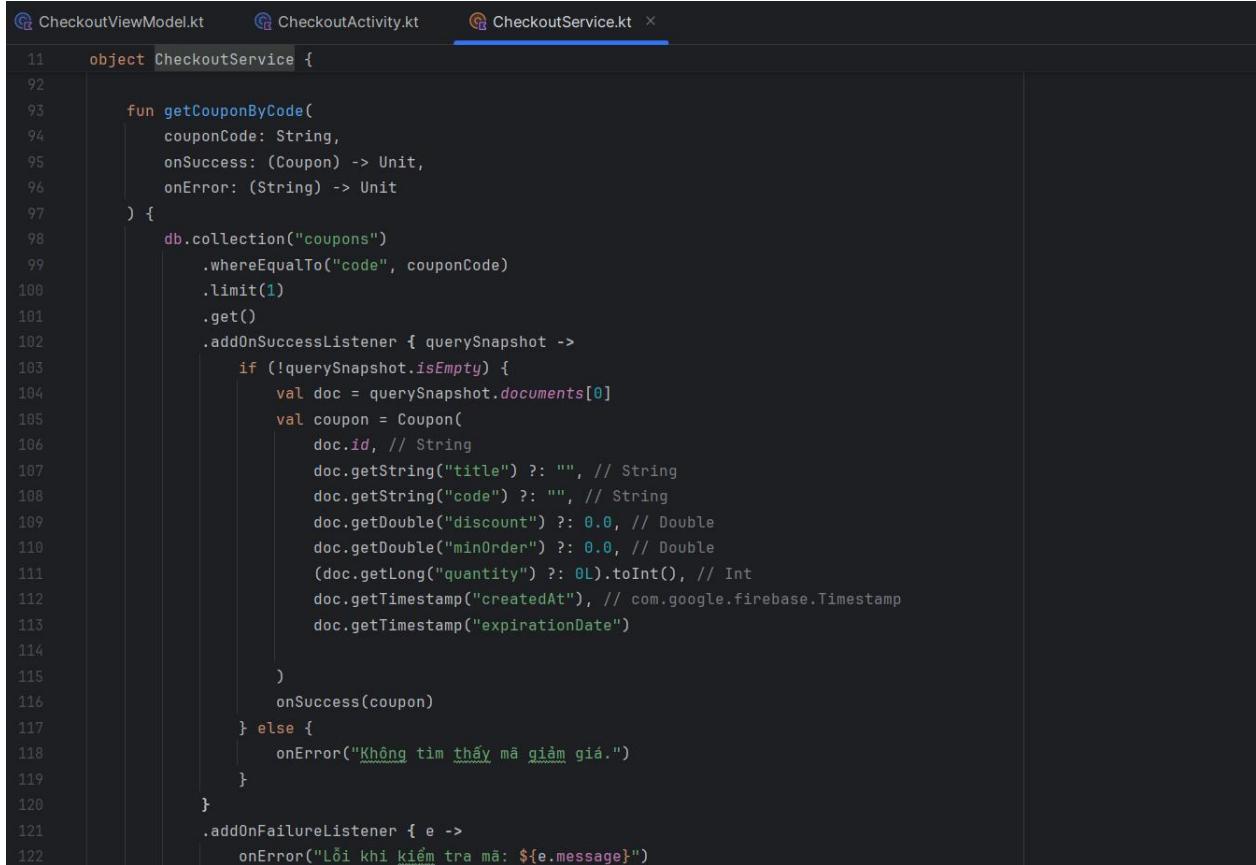
// ❷ Ghi các thay đổi vào Firestore
productRefs.forEach { (productId, updatedVariants) ->
    val productRef = db.collection("products").document(productId)
    transaction.update(productRef, "variants", updatedVariants)
}

// ❸ Tạo đơn hàng
val orderRef = db.collection("orders").document(order.orderId)
val cleanedOrder = order.copy(orderItems = cleanedOrderItems)
transaction.set(orderRef, cleanedOrder)

if (!order.couponId.isNullOrEmpty()) {
    val userRef = db.collection("users").document(order.uid)
    transaction.update(userRef, "coupons", FieldValue.arrayRemove(order.couponId))
}

}.addOnSuccessListener {
    onSuccess()
}.addOnFailureListener { e ->
    Log.e("TAG", "Lỗi khi đặt đơn hàng: ${e.message}")
    onError("Lỗi khi đặt đơn hàng: ${e.message} ?: \"Lỗi không xác định\"")
}
}
}

```



```

object CheckoutService {
    fun get_coupon_by_code(
        couponCode: String,
        onSuccess: (Coupon) -> Unit,
        onError: (String) -> Unit
    ) {
        db.collection("coupons")
            .whereEqualTo("code", couponCode)
            .limit(1)
            .get()
            .addOnSuccessListener { querySnapshot ->
                if (!querySnapshot.isEmpty) {
                    val doc = querySnapshot.documents[0]
                    val coupon = Coupon(
                        doc.id, // String
                        doc.getString("title") ?: "", // String
                        doc.getString("code") ?: "", // String
                        doc.getDouble("discount") ?: 0.0, // Double
                        doc.getDouble("minOrder") ?: 0.0, // Double
                        (doc.getLong("quantity") ?: 0L).toInt(), // Int
                        doc.getTimestamp("createdAt"), // com.google.firebaseio.Timestamp
                        doc.getTimestamp("expirationDate")
                    )
                    onSuccess(coupon)
                } else {
                    onError("Không tìm thấy mã giảm giá.")
                }
            }
            .addOnFailureListener { e ->
                onError("Lỗi khi kiểm tra mã: ${e.message}")
            }
    }
}

```

```
fun getCouponIdsOfUser(
    userId: String,
    onResult: (List<String>) -> Unit,
    onError: (Exception) -> Unit
) {
    FirebaseFirestore.getInstance()
        .collection("users")
        .document(userId)
        .get()
        .addOnSuccessListener { document ->
            val coupons = document.get("coupons") as? List<String> ?: emptyList()
            onResult(coupons)
        }
        .addOnFailureListener { e ->
            onError(e)
        }
}
```

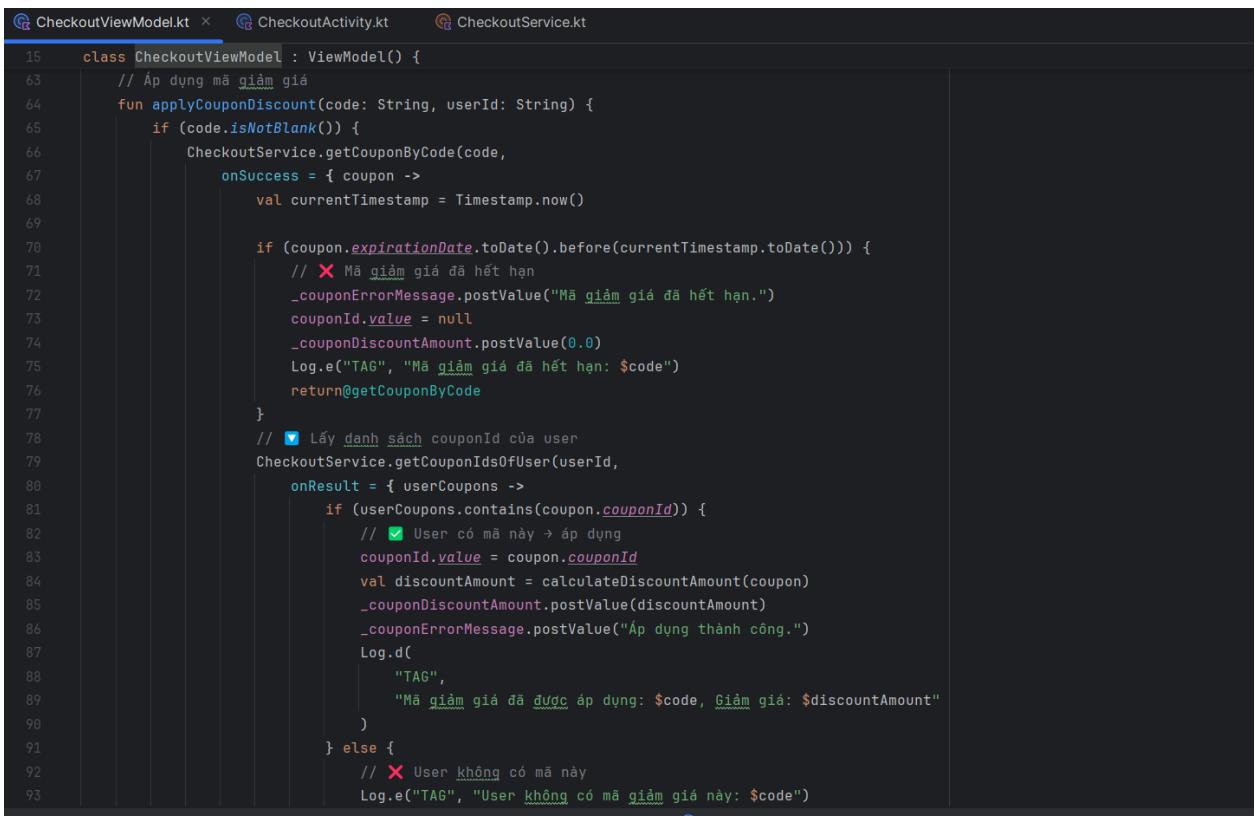
+ File: CheckoutViewModel

```

// Tính toán tổng tiền
fun calculateTotalPrice() {
    val itemsPrice = orderItems.sumOf { it.price * it.quantity }
    val discountAmount = _couponDiscountAmount.value ?: 0.0
    Log.d("TAG", "Tổng tiền sản phẩm: $itemsPrice, Giảm giá: $discountAmount")
    val shipPrice = calculateShippingPrice()
    shippingPrice.value = shipPrice
    totalPrice.value = itemsPrice - discountAmount + shipPrice
}

fun calculateTotalPriceOfProducts() {
    val itemsPrice = orderItems.sumOf { it.price * it.quantity }
    totalPriceProduct.value = itemsPrice.toDouble()
}

```



```

15 class CheckoutViewModel : ViewModel() {
16     // Áp dụng mã giảm giá
17     fun applyCouponDiscount(code: String, userId: String) {
18         if (code.isNotBlank()) {
19             CheckoutService.getCouponByCode(code,
20                 onSuccess = { coupon ->
21                     val currentTimestamp = Timestamp.now()
22
23                     if (coupon.expirationDate.toDate().before(currentTimestamp.toDate())) {
24                         // ✗ Mã giảm giá đã hết hạn
25                         _couponErrorMessage.postValue("Mã giảm giá đã hết hạn.")
26                         couponId.value = null
27                         _couponDiscountAmount.postValue(0.0)
28                         Log.e("TAG", "Mã giảm giá đã hết hạn: $code")
29                         return@getCouponByCode
30                     }
31                     // ✅ Lấy danh sách couponId của user
32                     CheckoutService.getCouponIdsOfUser(userId,
33                         onResult = { userCoupons ->
34                             if (userCoupons.contains(coupon.couponId)) {
35                                 // ✅ User có mã này → áp dụng
36                                 couponId.value = coupon.couponId
37                                 val discountAmount = calculateDiscountAmount(coupon)
38                                 _couponDiscountAmount.postValue(discountAmount)
39                                 _couponErrorMessage.postValue("Áp dụng thành công.")
40                                 Log.d(
41                                     "TAG",
42                                     "Mã giảm giá đã được áp dụng: $code, Giảm giá: $discountAmount"
43                                 )
44                             } else {
45                                 // ✗ User không có mã này
46                                 Log.e("TAG", "User không có mã giảm giá này: $code")
47                             }
48                         }
49                     )
50                 }
51             )
52         }
53     }
54 }

```

```

// Tính toán giảm giá từ coupon
private fun calculateDiscountAmount(coupon: Coupon): Double {
    val itemsPrice = orderItems.sumOf { it.price * it.quantity } + (shippingPrice.value ?: 0.0)
    return if (itemsPrice >= coupon.minOrder) {
        coupon.discount
    } else {
        couponId.value = null
        0.0
    }
}

// Tính toán phí vận chuyển
private fun calculateShippingPrice(): Double {
    return if (shippingMethod.value == "express") {
        100000.0
    } else {
        30000.0
    }
}

```

```

// Tạo đối tượng Order từ thông tin hiện tại
fun createOrder(uid: String): Order {
    val cleanedOrderItems = orderItems.map { cartItem ->
        OrderItem(
            productId = cartItem.productId,
            productName = cartItem.productName,
            variant = cartItem.variant,
            image = cartItem.image,
            quantity = cartItem.quantity,
            price = cartItem.price
        )
    }

    return Order(
        orderId = UUID.randomUUID().toString(),
        uid = uid,
        customerName = customerName.value ?: "",
        phoneCustomer = phoneCustomer.value ?: "",
        orderItems = cleanedOrderItems,
        shippingPrice = shippingPrice.value ?: 0.0,
        discount = _couponDiscountAmount.value ?: 0.0,
        couponId = couponId.value?.takeIf { it.isNotBlank() },
        address = address.value ?: "",
        totalPrice = totalPrice.value ?: 0.0,
        province = province.value ?: "",
        district = district.value ?: "",
        ward = ward.value ?: ""
    )
}

```

```
// Gửi đơn hàng lên server
fun submitOrder(uid: String) {
    val order = createOrder(uid)

    CheckoutService.submitOrder(order,
        onSuccess = { _orderStatus.postValue(true) },
        onError = { _orderStatus.postValue(false) }
    )
}
```

+ File: CheckoutService

```

11  object CheckoutService {
14      fun submitOrder(
15          order: Order,
16          onSuccess: () -> Unit,
17          onError: (String) -> Unit
18      ) {
19          Log.d("TAG", "Đang xử lý đơn hàng: ${order}")
20          db.runTransaction { transaction ->
21
22              val productRefs = mutableMapOf<String, List<Map<String, Any>>>()
23              val cleanedOrderItems = mutableListOf<OrderItem>()
24
25              // ❶ Kiểm tra tồn kho và cập nhật variants
26              order.orderItems.forEach { item ->
27                  val productRef = db.collection("products").document(item.productId)
28                  val snapshot = transaction.get(productRef)
29                  Log.d("TAG", "Đang kiểm tra tồn kho cho sản phẩm: ${item.variant}")
30                  val variants = snapshot.get("variants") as? List<Map<String, Any>?
31                  ?: throw FirebaseFirestoreException(
32                      "Sản phẩm không có biến thể",
33                      FirebaseFirestoreException.Code.ABORTED
34                  )
35
36                  val updatedVariants = variants.map { variant ->
37                      val color = variant["color"] as? String
38                      val sizes = variant["sizes"] as? List<Map<String, Any>? ?: emptyList()
39
40                      if (color == item.variant.color) {
41                          val updatedSizes = sizes.map { sizeMap ->
42                              val sizeName = sizeMap["size"] as? String
43                              if (sizeName == item.variant.size) {
44                                  val currentQty = (sizeMap["quantity"] as? Long)? .toInt() ?: 0

```



```

11  object CheckoutService {
14      fun submitOrder(
15          db: FirebaseFirestore,
16          order: Order,
17          onSuccess: () -> Unit,
18          onError: (String) -> Unit
19      ) {
20          db.runTransaction { transaction ->
21
22              order.orderItems.forEach { item ->
23                  val updatedVariants = item.variants.map { variant ->
24
25                      if (currentQty < item.quantity) {
26                          throw FirebaseFirestoreException(
27                              "${item.productName} (${item.variant.color} - ${item.variant.size}) không đủ hàng!",
28                              FirebaseFirestoreException.Code.ABORTED
29                          )
30                      }
31
32                      sizeMap.toMutableMap().apply {
33                          this["quantity"] = (currentQty - item.quantity).toLong()
34                      }
35
36                      } else sizeMap
37                  }
38
39                  variant.toMutableMap().apply {
40                      this["sizes"] = updatedSizes
41                  }
42
43                  } else variant
44
45
46                  productRefs[item.productId] = updatedVariants
47                  cleanedOrderItems.add(item)
48
49              }
50
51          }
52
53      }
54
55  }

```

```

    // ❷ Ghi các thay đổi vào Firestore
    productRefs.forEach { (productId, updatedVariants) ->
        val productRef = db.collection("products").document(productId)
        transaction.update(productRef, "variants", updatedVariants)
    }

    // ❸ Tạo đơn hàng
    val orderRef = db.collection("orders").document(order.orderId)
    val cleanedOrder = order.copy(orderItems = cleanedOrderItems)
    transaction.set(orderRef, cleanedOrder)

    if (!order.couponId.isNullOrEmpty()) {
        val userRef = db.collection("users").document(order.uid)
        transaction.update(userRef, "coupons", FieldValue.arrayRemove(order.couponId))
    }

}.addOnSuccessListener {
    onSuccess()
}.addOnFailureListener { e ->
    Log.e("TAG", "Lỗi khi đặt đơn hàng: ${e.message}")
    onError("Lỗi khi đặt đơn hàng: ${e.message} ?: \"Lỗi không xác định\"")
}
}
}

```

```

object CheckoutService {
    fun get_coupon_by_code(
        couponCode: String,
        onSuccess: (Coupon) -> Unit,
        onError: (String) -> Unit
    ) {
        db.collection("coupons")
            .whereEqualTo("code", couponCode)
            .limit(1)
            .get()
            .addOnSuccessListener { querySnapshot ->
                if (!querySnapshot.isEmpty) {
                    val doc = querySnapshot.documents[0]
                    val coupon = Coupon(
                        doc.id, // String
                        doc.getString("title") ?: "", // String
                        doc.getString("code") ?: "", // String
                        doc.getDouble("discount") ?: 0.0, // Double
                        doc.getDouble("minOrder") ?: 0.0, // Double
                        (doc.getLong("quantity") ?: 0L).toInt(), // Int
                        doc.getTimestamp("createdAt"), // com.google.firebaseio.Timestamp
                        doc.getTimestamp("expirationDate")
                    )
                    onSuccess(coupon)
                } else {
                    onError("Không tìm thấy mã giảm giá.")
                }
            }
            .addOnFailureListener { e ->
                onError("Lỗi khi kiểm tra mã: ${e.message}")
            }
    }
}

```

```

fun getCouponIdsOfUser(
    userId: String,
    onResult: (List<String>) -> Unit,
    onError: (Exception) -> Unit
) {
    FirebaseFirestore.getInstance()
        .collection("users")
        .document(userId)
        .get()
        .addOnSuccessListener { document ->
            val coupons = document.get("coupons") as? List<String> ?: emptyList()
            onResult(coupons)
        }
        .addOnFailureListener { e ->
            onError(e)
        }
}

```

3.4.6. Chức năng hiển thị Collection

+ File: ProductCollectionService

```

package com.project.clothingstore.service;

import ...

public class ProductCollectionService {

    private final CollectionReference collectionsRef;

    private FirebaseFirestore db = FirebaseFirestore.getInstance();

    public ProductCollectionService() {
        collectionsRef = FirebaseHelper.getProductCollection();
    }

    public ProductCollectionService(CollectionReference collectionsRef) {
        this.collectionsRef = collectionsRef;
    }

    public void getCollectionList(MutableLiveData<List<ProductCollections>> liveData, @Nullable String collectionType) {
        CollectionReference collectionsRef = db.collection(collectionPath: "collections");

        Query query = collectionsRef;

        // Nếu collectionType khác null hoặc không rỗng thì lọc
        if (collectionType != null && !collectionType.isEmpty()) {
            // Chuyển đổi collectionType thành số nguyên
            query = query.whereEqualTo(field: "collectionType", collectionType);
        }
    }
}

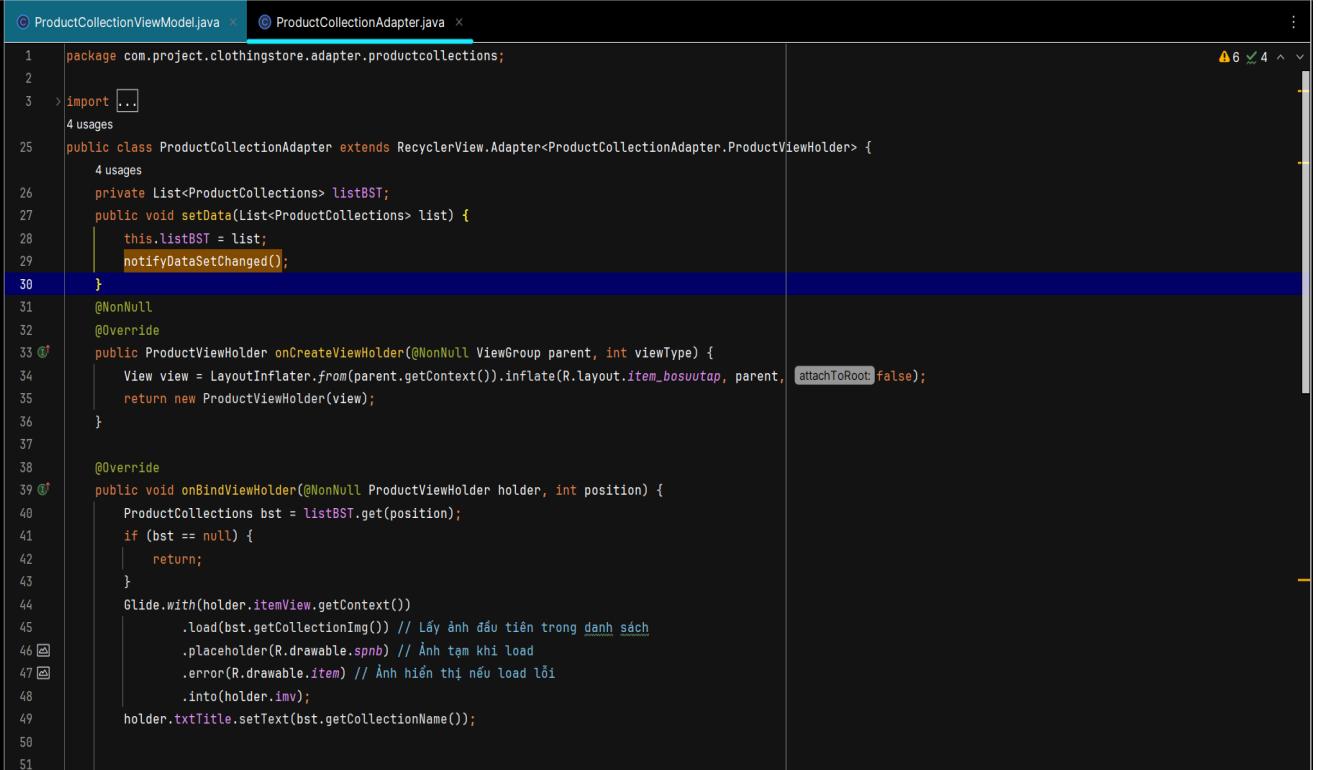
```

```
gle-services.json   RecommendProductViewModel.java   MainActivity.java   SearchBarFragment.java   ProductCollectionService.java   ProductAdapter.java   Product.java   ...
18 public class ProductCollectionService {
19     3 usages
20     public void getCollectionList(MutableLiveData<List<ProductCollections>> liveData, @Nullable String collectionType) {
21         CollectionReference collectionsRef = db.collection(collectionPath: "collections");
22
23         Query query = collectionsRef;
24
25         // Nếu collectionType khác null hoặc không rỗng thì lọc
26         if (collectionType != null && !collectionType.isEmpty()) {
27             // Chuyển đổi collectionType thành số nguyên
28             query = query.whereEqualTo(field: "collectionType", collectionType);
29         }
30
31         // Giới hạn 5 kết quả
32         query.limit(5).get()
33             .addOnCompleteListener(task -> {
34                 if (task.isSuccessful()) {
35                     List<ProductCollections> productCollectionsList = new ArrayList<>();
36                     for (QueryDocumentSnapshot doc : task.getResult()) {
37                         ProductCollections productCollections = doc.toObject(ProductCollections.class);
38                         productCollections.setCollectionId(doc.getId());
39                         productCollectionsList.add(productCollections);
40                     }
41
42                     liveData.setValue(productCollectionsList);
43                 } else {
44                     liveData.setValue(new ArrayList<>()); // Tránh null
45                 }
46             });
47     }
48
49
50
51
52
53
54
55
56
57
58
59
60 }
```

+ File: ProductCollectionViewModel

```
ProductCollectionViewModel.java ...
1 package com.project.clothingstore.viewmodel.productcollections;
2
3 > import ...
4
5     3 usages
6     public class ProductCollectionViewModel extends ViewModel {
7         2 usages
8         private MutableLiveData<List<ProductCollections>> listProductCategory = new MutableLiveData<>();
9
10        1 usage
11        private ProductCollectionService productCollectionService = new ProductCollectionService();
12
13        no usages
14        > public ProductCollectionViewModel() { LoadProductCategory(); }
15
16
17        1 usage
18        public void LoadProductCategory() {
19            productCollectionService.getCollectionList(listProductCategory, collectionType: "1");
20        }
21
22
23        1 usage
24        public LiveData<List<ProductCollections>> getListProductCategory() {
25            return listProductCategory;
26        }
27    }
```

+ File: ProductCollectionAdapter



```
ProductCollectionViewModel.java x ProductCollectionAdapter.java x
1 package com.project.clothingstore.adapter.productcollections;
2
3 > import ...
4 usages
5
6 public class ProductCollectionAdapter extends RecyclerView.Adapter<ProductCollectionAdapter.ProductViewHolder> {
7     4 usages
8     private List<ProductCollections> listBST;
9     public void setData(List<ProductCollections> list) {
10         this.listBST = list;
11         notifyDataSetChanged();
12     }
13
14     @NonNull
15     @Override
16     public ProductViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
17         View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_bosuutap, parent, false);
18         return new ProductViewHolder(view);
19     }
20
21     @Override
22     public void onBindViewHolder(@NonNull ProductViewHolder holder, int position) {
23         ProductCollections bst = listBST.get(position);
24         if (bst == null) {
25             return;
26         }
27         Glide.with(holder.itemView.getContext())
28             .load(bst.getCollectionImg()) // Lấy ảnh đầu tiên trong danh sách
29             .placeholder(R.drawable.spnb) // Ảnh tạm khi load
30             .error(R.drawable.item) // Ảnh hiển thị nếu load lỗi
31             .into(holder.img);
32         holder.txtTitle.setText(bst.getCollectionName());
33     }
34
35     public class ProductViewHolder extends RecyclerView.ViewHolder {
36         ...
37     }
38 }
39
40
41
42
43
44
45
46
47
48
49
50
51
```

```
ProductCollectionViewModel.java ProductCollectionAdapter.java x
25 public class ProductCollectionAdapter extends RecyclerView.Adapter<ProductCollectionAdapter.ProductViewHolder> {
26
27     public void onBindViewHolder(@NotNull ProductViewHolder holder, int position) {
28
29         holder.itemView.setOnClickListener(v -> {
30             // Chuyển đến CollectionActivity và truyền dữ liệu
31             Intent intent = new Intent(holder.itemView.getContext(), CollectionActivity.class);
32             intent.putExtra("name: "collectionImg", bst.getCollectionImg());
33             intent.putExtra("name: "collectionId", bst.getCollectionId());
34             holder.itemView.getContext().startActivity(intent);
35         });
36     }
37
38     @Override
39     public int getItemCount() {
40         if (listBST != null){
41             return listBST.size();
42         }
43         return 0;
44     }
45
46     4 usages
47     public class ProductViewHolder extends RecyclerView.ViewHolder {
48
49         2 usages
50         ImageView imv;
51
52         2 usages
53         TextView txtTitle;
54
55         1 usage
56         public ProductViewHolder(@NotNull View itemView) {
57             super(itemView);
58             imv = itemView.findViewById(R.id.img_BST);
59             txtTitle = itemView.findViewById(R.id.txt_img_BST);
60         }
61     }
62 }
```

+File: ProductCollectionFragment

```

ProductCollectionFragment.java ProductCollectionAdapter.java ProductCollectionFragment.java
21 public class ProductCollectionFragment extends Fragment {
38
39     @Override
40     public View onCreateView(LayoutInflater inflater, ViewGroup container,
41                             Bundle savedInstanceState) {
42         View view = inflater.inflate(R.layout.fragment_product_collection, container, false);
43         mviewPager2 = view.findViewById(R.id.vp2_product_category);
44         mcircleIndicator3 = view.findViewById(R.id.cicle_BST);
45
46         // Khởi tạo ViewModel
47         boSuuTapViewModel = new ViewModelProvider(owner, this).get(ProductCollectionViewModel.class);
48
49         // Khởi tạo Adapter với danh sách trống
50         adapterBST = new ProductCollectionAdapter();
51         mviewPager2.setAdapter(adapterBST);
52         mcircleIndicator3.setViewPager(mviewPager2);
53
54         // Lắng nghe dữ liệu từ ViewModel
55         boSuuTapViewModel.getListProductCategory().observe(getViewLifecycleOwner(), list -> {
56             adapterBST.setData(list);
57             mcircleIndicator3.createIndicators(list.size(), currentPosition: 0);
58         });
59
60     >     mviewPager2.registerOnPageChangeCallback(onPageSelected(position) -> {
61             super.onPageSelected(position);
62             handler.removeCallbacks(runnable);
63             handler.postDelayed(runnable, delayMillis: 5000);
64         });
65
66     >
67
68         return view;
69     }
70 }

```

```

21 public class ProductCollectionFragment extends Fragment {
40     public View onCreateView(LayoutInflater inflater, ViewGroup container,
56         adapterBST.setData(list);
57         mcircleIndicator3.createIndicators(list.size(), currentPosition: 0);
58     });
59
60     >     mviewPager2.registerOnPageChangeCallback(onPageSelected(position) -> {
61         super.onPageSelected(position);
62         handler.removeCallbacks(runnable);
63         handler.postDelayed(runnable, delayMillis: 5000);
64     });
65
66     >
67
68         return view;
69     }
70
72     @Override
73     public void onPause() {
74         super.onPause();
75         handler.removeCallbacks(runnable);
76     }
77
78     @Override
79     public void onResume() {
80         super.onResume();
81         handler.postDelayed(runnable, delayMillis: 5000);
82     }
83 }

```

3.4.7. Chức năng hiển thị Product

+ File ProductService:

ProductService.java

```
4 > import ...  
23  
12 usages  
24 public class ProductService {  
    1 usage  
    private final CollectionReference productRef;  
    6 usages  
    private FirebaseFirestore db = FirebaseFirestore.getInstance();  
27  
    4 usages  
28 >     public ProductService() { productRef = FirebaseHelper.getProductCollection(); }  
31  
32  
    2 usages  
33     public void getSanPhamList(MutableLiveData<List<Product>> liveData, String field, int limitt) {  
34         CollectionReference productsRef = db.collection(collectionPath).  
35  
36         // Lấy 5 sản phẩm có lượt bán cao nhất  
37         productsRef.orderBy(field, Query.Direction.DESCENDING).  
38             .limit(limitt)  
39             .get().Task<QuerySnapshot>  
40             .addOnCompleteListener(task -> {  
41                 if (task.isSuccessful()) {  
42                     List<Product> productList = new ArrayList<>();  
43                     for (QueryDocumentSnapshot doc : task.getResult()) {  
44                         Product product = doc.toObject(Product.class);  
45                         product.setProductId(doc.getId());  
46                         productList.add(product);  
47                     }  
48                     liveData.setValue(productList);  
49                 } else {  
50                     liveData.setValue(new ArrayList<>()); // Tránh null
```

```
ProductService.java x
14 40
24 public class ProductService {
25     public void getSanPhamList(MutableLiveData<List<Product>> liveData, String field, int limit) {
26         .addOnCompleteListener(task -> {
27             if (task.isSuccessful()) {
28                 List<Product> productList = task.getResult();
29                 liveData.setValue(productList);
30             } else {
31                 liveData.setValue(new ArrayList<>()); // Tránh null
32             }
33         });
34     }
35
36     // Lay san pham theo categoryId
37     no usages
38     public void getProductByCategoryIdList(MutableLiveData<List<Product>> liveData, String categoryId) {
39         if (categoryId == null || categoryId.isEmpty()) {
40             liveData.setValue(new ArrayList<>()); // Tránh trường hợp categoryId không hợp lệ
41             return;
42         }
43
44         CollectionReference productsRef = db.collection(collectionPath: "products");
45
46         productsRef.whereEqualTo(field: "categoryId", categoryId).query
47             .get() Task<QuerySnapshot>
48             .addOnCompleteListener(task -> {
49                 if (task.isSuccessful()) {
50                     List<Product> productList = new ArrayList<>();
51                     for (QueryDocumentSnapshot doc : task.getResult()) {
52                         Product product = doc.toObject(Product.class);
53                         product.setProductId(doc.getId());
54                         productList.add(product);
55                     }
56                     liveData.setValue(productList);
57                 } else {
58                     liveData.setValue(new ArrayList<>()); // Tránh null nếu truy vấn thất bại
59                 }
60             });
61     }
62 }
```

```
ProductService.java x
24 public class ProductService {
79
80     // Lọc sản phẩm theo categoryId, categoryType, minPrice, maxPrice, rating, discount
81
82     1usage
83     public void getFilteredProducts(MutableLiveData<List<Product>> liveData, int productType, int minPrice, int maxPrice, double rating, List<Integer> discounts, String key
84         CollectionReference productsRef = db.collection(collectionPath:"products");
85         Query query = productsRef;
86
87         // Lọc theo categoryId
88         if (categoryId != null && !categoryId.isEmpty()) {
89             query = query.whereEqualTo(field: "categoryId", categoryId);
90
91             // Thêm điều kiện nếu có
92             if (productType != -1) {
93                 query = query.whereEqualTo(field: "productType", productType);
94             }
95
96             if (minPrice != -1) {
97                 query = query.whereGreaterThanOrEqualTo(field: "price", minPrice);
98             }
99
100            if (maxPrice != -1) {
101                query = query.whereLessThanOrEqualTo(field: "price", maxPrice);
102            }
103
104            // Lọc theo đánh giá
105            if (rating != -1) {
106                query = query.whereGreaterThanOrEqualTo(field: "totalRating", rating);
107            }
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
```

```
ProductService.java x
24 public class ProductService {
82     public void getFilteredProducts(MutableLiveData<List<Product>> liveData, int productType, int minPrice, int maxPrice, double rating, List<Integer> discounts, String key
109     // Lọc số bộ theo discount
110     if (discounts != null && !discounts.isEmpty()) {
111         double minDiscountPercent = (discounts.get(0) + 1) / 10.0;
112         for (int d : discounts) {
113             double percent = (d + 1) / 10.0;
114             if (percent < minDiscountPercent) {
115                 minDiscountPercent = percent;
116             }
117         }
118         query = query.whereGreaterThanOrEqualTo(field: "discount", minDiscountPercent);
119     }
120
121     query.get().addOnCompleteListener(task -> {
122         if (task.isSuccessful()) {
123             List<Product> productList = new ArrayList<>();
124             for (QueryDocumentSnapshot doc : task.getResult()) {
125                 Product product = doc.toObject(Product.class);
126                 product.setProductId(doc.getId());
127
128                 if (discounts != null && !discounts.isEmpty()) {
129                     double productDiscount = product.getDiscount(); // ví dụ: 0.3
130                     boolean isMatched = false;
131
132                     for (int level : discounts) {
133                         double requiredDiscount = (level + 1) / 10.0;
134                         if (productDiscount == requiredDiscount) {
135                             isMatched = true;
136                             break;
137                         }
138                     }
139                 }
140             }
141             productList.add(product);
142             liveData.postValue(productList);
143         }
144     });
145 }
```

```

24  public class ProductService {
82      public void getFilteredProducts(MutableLiveData<List<Product>> liveData, int productType, int minPrice, int maxPrice, double rating, List<Integer> discounts, String keyword) {
121          query.get().addOnCompleteListener(task -> {
137              if (!task.isSuccessful()) {
138                  liveData.setValue(null);
139              }
140              if (!isMatched) continue;
141              String productName = product.getProductName();
142              if (keyword != null && !keyword.isEmpty()) {
143                  // Só sánh gần đúng, không phân biệt hoa thường
144                  if (productName != null && productName.toLowerCase().contains(keyword.toLowerCase().trim())) {
145                      productList.add(product);
146                  }
147                  } else {
148                      productList.add(product);
149                  }
150                  }
151              }
152          }
153          liveData.setValue(productList);
154      } else {
155          liveData.setValue(new ArrayList<>()); // Tránh null nếu truy vấn thất bại
156      }
157  });
158 }
159
160
161
162 // Lay san pham theo CollectionId
163 1usage
164  public void getProductByCollectionIdList(MutableLiveData<List<Product>> liveData, String collectionId) {
165      if (collectionId == null || collectionId.isEmpty()) {

```

```

24  public class ProductService {
159
160
161
162 // Lay san pham theo CollectionId
163 1usage
164  public void getProductByCollectionIdList(MutableLiveData<List<Product>> liveData, String collectionId) {
165      if (collectionId == null || collectionId.isEmpty()) {
166          liveData.setValue(new ArrayList<>()); // Tránh trường hợp categoryId không hợp lệ
167          return;
168      }
169
170      CollectionReference productsRef = db.collection("products");
171
172      productsRef.whereEqualTo("collectionId", collectionId).get()
173          .addOnCompleteListener(task -> {
174              if (task.isSuccessful()) {
175                  List<Product> productList = new ArrayList<>();
176                  for (QueryDocumentSnapshot doc : task.getResult()) {
177                      Product product = doc.toObject(Product.class);
178                      product.setProductId(doc.getId());
179                      productList.add(product);
180                  }
181                  liveData.setValue(productList);
182              } else {
183                  liveData.setValue(new ArrayList<>()); // Tránh null nếu truy vấn thất bại
184              }
185          });
186
187
188

```

+File: ProductViewModal

```
ProductService.java ProductViewModel.java

1 package com.project.clothingstore.viewmodel.Product;
2
3 > import ...
4 usages
5 public class ProductViewModel extends ViewModel {
6     2 usages
7     private MutableLiveData<List<Product>> listProduct = new MutableLiveData<>();
8     1 usage
9     private ProductService productService = new ProductService();
10
11     no usages
12     public ProductViewModel() {
13         // Default load with initial limit
14     }
15
16     1 usage
17     public void loadFilteredProduct(int categoriType, int minPrice, int maxPrice,
18                                     double rating, List<Integer> discounts, String productName, String categoriId) {
19         productService.getFilteredProducts(listProduct, categoriType, minPrice, maxPrice, rating, discounts, productName, categoriId);
20     }
21
22
23     1 usage
24     public LiveData<List<Product>> getListProduct() { return listProduct; }
25
26 }
```

+ File Product Adapter:

```
ProductAdapter.java ProductViewModel.java
Project Alt+1 ce.java

1 package com.project.clothingstore.adapter.product;
2
3 > import ...
4 usages
5 public class ProductAdapter extends RecyclerView.Adapter<ProductAdapter.ProductViewHolder> {
6
7     4 usages
8     List<Product> listSP;
9     public void setData(List<Product> list) {
10         this.listSP = list;
11         notifyDataSetChanged();
12     }
13
14     @NonNull
15     @Override
16     public ProductViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
17         View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_products, parent, attachToRoot:false);
18         return new ProductViewHolder(view);
19     }
20
21     @Override
22     public void onBindViewHolder(@NonNull ProductViewHolder holder, int position) {
23         Product sp = listSP.get(position);
24         if (sp == null || sp.getImages() == null || sp.getImages().isEmpty()) {
25             return;
26         }
27
28         // Load ảnh từ URL bằng Glide
29         Glide.with(holder.itemView.getContext())
30             .load(sp.getImages().get(0)) // Lấy ảnh đầu tiên trong danh sách
31             .placeholder(R.drawable.item) // Ảnh tạm khi load
32     }
33 }
```

```
© ProductService.java © ProductViewModel.java © ProductAdapter.java × : 4 5 ^ v
24 public class ProductAdapter extends RecyclerView.Adapter<ProductAdapter.ProductViewHolder> {
40     public void onBindViewHolder(@NonNull ProductViewHolder holder, int position) {
53         String productName = sp.getProductName();
54         final int MAX_PRODUCT_NAME_LENGTH = 18; // Độ dài tối đa của tên sản phẩm
55         if (productName.length() > MAX_PRODUCT_NAME_LENGTH) {
56             productName = productName.substring(0, 18) + "...";
57         }
58         holder.txtName.setText(productName);
59
60         DecimalFormat formatter = new DecimalFormat("###,###");
61         String formatted = formatter.format(sp.getPrice()); // "7,000,000,000"
62         formatted = formatted.replace(",","");
63         holder.txtPrice.setText(String.valueOf(obj.d + formatted));
64         holder.txtOldPrice.setText(String.valueOf(obj.d + formatter.format(sp.getPriceBeforeDiscount())));
65         holder.txtOldPrice.setPaintFlags(holder.txtOldPrice.getPaintFlags() | Paint.STRIKE_THRU_TEXT_FLAG);
66         holder.ratingBar.setRating(sp.getTotalRating());
67         holder.itemView.setOnClickListener(v -> {
68             Context context = v.getContext(); // Lấy context từ View
69             Intent intent = new Intent(context, ProductDetailActivity.class);
70             intent.putExtra("productId", sp.getProductId()); // Truyền ID của sản phẩm
71             context.startActivity(intent);
72         });
73     }
74
75     @Override
76     public int getItemCount() {
77         if (listSP != null) {
78             return listSP.size();
79         }
80         return 0;
81     }
82 }
```

```
© ProductService.java © ProductViewModel.java © ProductAdapter.java × : 4 5 ^ v
24 public class ProductAdapter extends RecyclerView.Adapter<ProductAdapter.ProductViewHolder> {
76     public int getItemCount() {
77         return listSP.size();
78     }
79     return 0;
80 }
81
82
4 usages
83     public class ProductViewHolder extends RecyclerView.ViewHolder {
84         2 usages
85         ImageView img;
86         2 usages
87         TextView txtName, txtPrice, txtOldPrice;
88
89         2 usages
90         RatingBar ratingBar;
91         1 usage
92         public ProductViewHolder(@NonNull View itemView) {
93             super(itemView);
94             img = itemView.findViewById(R.id.img_item_product);
95             txtName = itemView.findViewById(R.id.txt_name_product);
96             txtPrice = itemView.findViewById(R.id.txt_price_product);
97             txtOldPrice = itemView.findViewById(R.id.txt_oldprice_product);
98             ratingBar = itemView.findViewById(R.id.rating_product);
99         }
100     }
101 }
```

+ File ProductFragment:

```

1 package com.project.clothingstore.view.fragment.product;
2
3 > import ...
4
5 </> public class ProductFragment extends Fragment {
6     3 usages
7     private ProductViewModel productViewModel;
8     3 usages
9     private ProductAdapter productAdapter;
10    3 usages
11    private RecyclerView recyclerView;
12    2 usages
13    String categoryId, productName;
14    2 usages
15    int categoriType, minPrice, maxPrice;
16    2 usages
17    double rating;
18    2 usages
19    ArrayList<Integer> discountList;
20
21    @Override
22    @Override
23    public View onCreateView(LayoutInflater inflater, ViewGroup container,
24                            Bundle savedInstanceState) {
25        View view = inflater.inflate(R.layout.fragment_product, container, false);
26
27        if (getArguments() != null) {
28            categoryId = getArguments().getString(key: "categoryId");
29            productName = getArguments().getString(key: "productName");
30
31            categoriType = getArguments().getInt(key: "categoriType", defaultValue: -1);
32            minPrice = getArguments().getInt(key: "minPrice", defaultValue: -1);
33            maxPrice = getArguments().getInt(key: "maxPrice", defaultValue: -1);
34        }
35    }
36
37
38
39
40
41
42
43
44

```

3.4.8. Chức năng hiển thị Category

+ File CaterogyService:

```

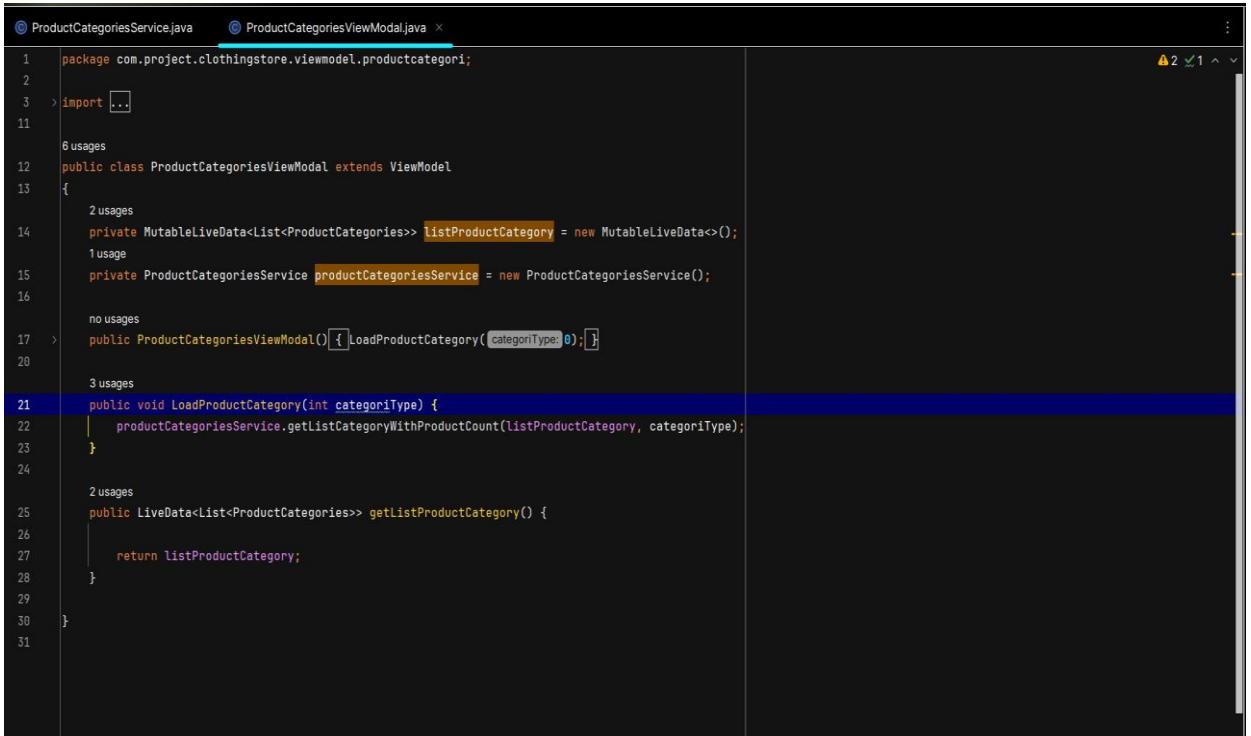
1 package com.project.clothingstore.service;
2
3 > import ...
4
5
6 </> public class ProductCategoriesService {
7
8     2 usages
9     private final CollectionReference productRef;
10
11     private FirebaseFirestore db = FirebaseFirestore.getInstance();
12
13     1 usage
14     public ProductCategoriesService(CollectionReference productRef) {
15         this.productRef = productRef;
16     }
17
18
19     // Lay category theo type khong co so luong
20     no usages
21     public ProductCategoriesService(CollectionReference productRef) {
22         this.productRef = productRef;
23     }
24
25
26     // Lay category theo type khong co so luong
27     no usages
28     public void getListCategoryByType(MutableLiveData<List<ProductCategories>> liveData, int categoriType) {
29         CollectionReference collectionsRef = db.collection(collectionPath: "categories");
30
31         Query query = collectionsRef;
32
33         query = query.whereEqualTo(field: "categoriType", categoriType);
34
35         // Lay tat ca ket qua
36         query.get()
37             .addOnCompleteListener(task -> {
38                 if (task.isSuccessful()) {
39                     liveData.postValue(task.getResult());
40                 } else {
41                     liveData.postValue(null);
42                 }
43             });
44     }
45
46
47
48
49

```

```
ProductCategoriesService.java
16 public class ProductCategoriesService {
17     public void getListCategoryByType(MutableLiveData<List<ProductCategories>> liveData, int categoriType) {
18         .addOnCompleteListener(task -> {
19             for (QueryDocumentSnapshot doc : task.getResult()) {
20                 ProductCategories productCategories = doc.toObject(ProductCategories.class);
21                 productCategories.setCategoryId(doc.getId());
22                 productCategorilist.add(productCategories);
23             }
24             liveData.setValue(productCategorilist);
25         } else {
26             liveData.setValue(new ArrayList<>()); // Tránh null
27         }
28     });
29 }
30
31     // Lay category theo type co so luong
32     1 usage
33     public void getListCategoryWithProductCount(MutableLiveData<List<ProductCategories>> liveData, int categoriType) {
34         CollectionReference categoriesRef = db.collection(collectionPath:"categories");
35         CollectionReference productsRef = db.collection(collectionPath:"products");
36
37         Query query = categoriesRef.whereEqualTo(field: "categoriType", categoriType);
38
39         query.get().addOnCompleteListener(task -> {
40             if (task.isSuccessful()) {
41                 List<ProductCategories> categoryList = new ArrayList<>();
42
43                 for (QueryDocumentSnapshot doc : task.getResult()) {
44                     ProductCategories category = doc.toObject(ProductCategories.class);
45                     String categoryId = doc.getId();
46                     category.setCategoryId(categoryId);
47                 }
48             }
49         });
50     }
51 }
```

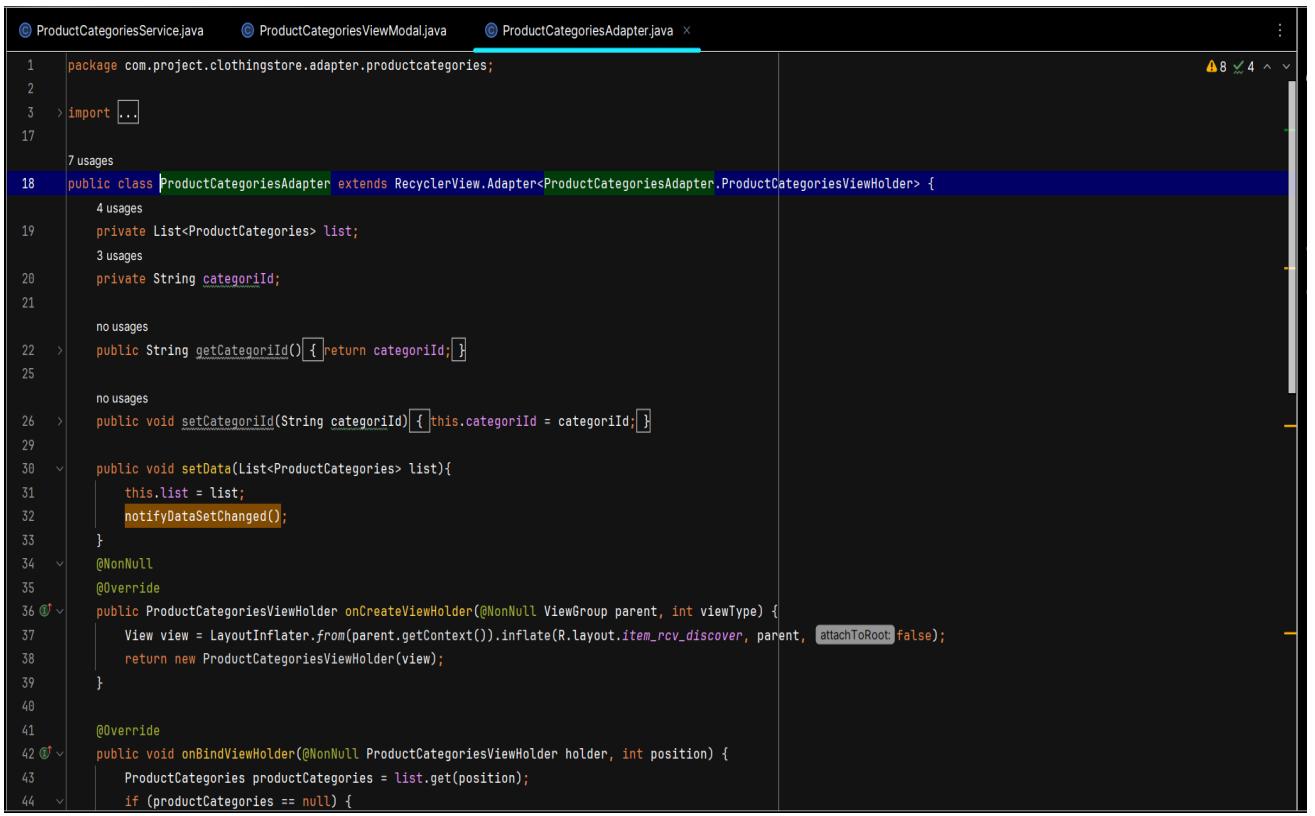
```
ProductCategoriesService.java
16 public class ProductCategoriesService {
17     1 usage
18     public void getListCategoryWithProductCount(MutableLiveData<List<ProductCategories>> liveData, int categoriType) {
19         CollectionReference categoriesRef = db.collection(collectionPath: "categories");
20         CollectionReference productsRef = db.collection(collectionPath: "products");
21
22         Query query = categoriesRef.whereEqualTo(field: "categoriType", categoriType);
23
24         query.get().addOnCompleteListener(task -> {
25             if (task.isSuccessful()) {
26                 List<ProductCategories> categoryList = new ArrayList<>();
27
28                 for (QueryDocumentSnapshot doc : task.getResult()) {
29                     ProductCategories category = doc.toObject(ProductCategories.class);
30                     String categoryId = doc.getId();
31                     category.setCategoryId(categoryId);
32
33                     productsRef.whereEqualTo(field: "categoryId", categoryId).get().addOnSuccessListener(productTask -> {
34                         int count = productTask.size();
35                         category.setQuantity(count); // < gán số lượng
36                         categoryList.add(category);
37
38                         // Nếu đủ số lượng category thì cập nhật LiveData
39                         if (categoryList.size() == task.getResult().size()) {
40                             liveData.setValue(categoryList);
41                         }
42                     });
43                 }
44             }
45         });
46     }
47 }
```

+ File CategoryViewModal



```
ProductCategoriesService.java  ProductCategoriesViewModel.java  x
1 package com.project.clothingstore.viewmodel.productcategories;
2
3 > import ...
11
12 6 usages
13 public class ProductCategoriesViewModel extends ViewModel
14 {
15     2 usages
16     private MutableLiveData<List<ProductCategories>> listProductCategory = new MutableLiveData<>();
17     1 usage
18     private ProductCategoriesService productCategoriesService = new ProductCategoriesService();
19
20     no usages
21     > public ProductCategoriesViewModel() { LoadProductCategory(categoriType); }
22
23     3 usages
24     public void LoadProductCategory(int categoriType) {
25         productCategoriesService.getListCategoryWithProductCount(listProductCategory, categoriType);
26     }
27
28     2 usages
29     public LiveData<List<ProductCategories>> getListProductCategory() {
30         return listProductCategory;
31     }
32 }
```

+ File CategoriAdapter:

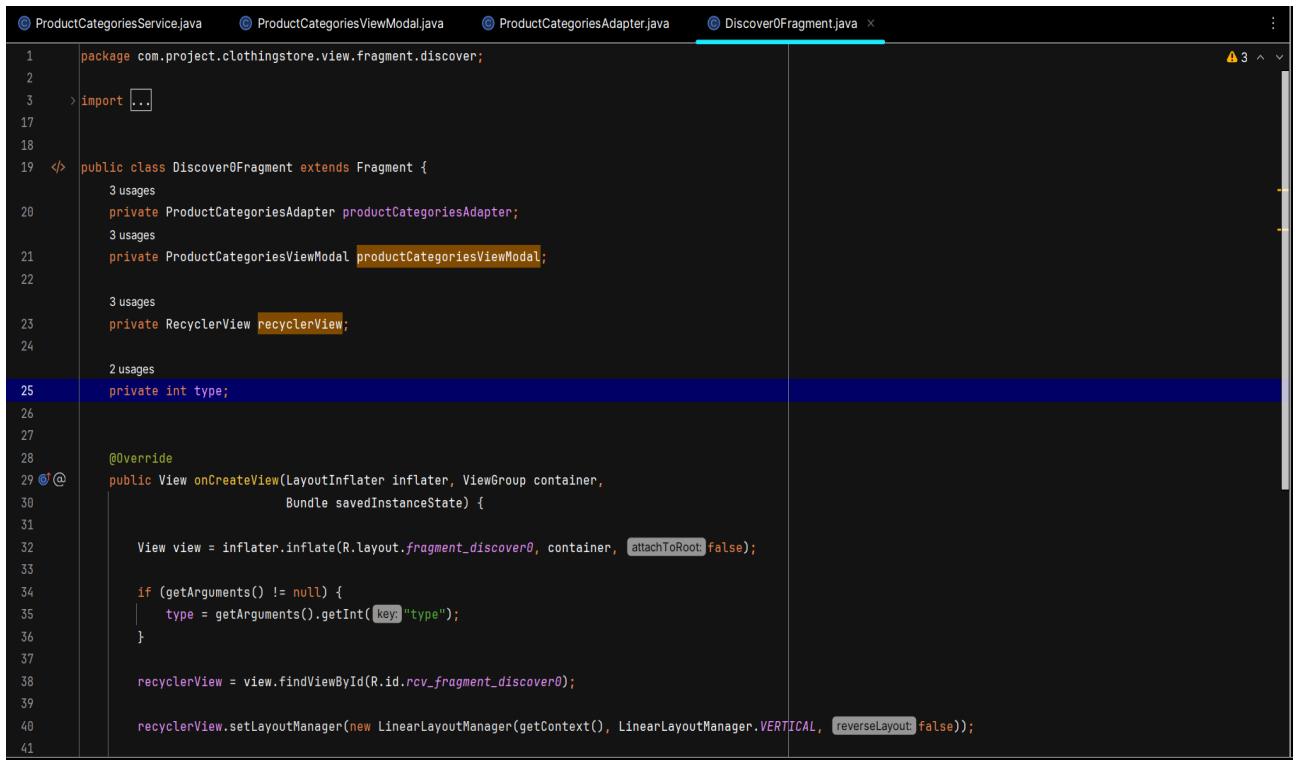


```
ProductCategoriesService.java  ProductCategoriesViewModel.java  ProductCategoriesAdapter.java  x
1 package com.project.clothingstore.adapter.productcategories;
2
3 > import ...
17
18 7 usages
19 public class ProductCategoriesAdapter extends RecyclerView.Adapter<ProductCategoriesAdapter.ProductCategoriesViewHolder> {
20     4 usages
21     private List<ProductCategories> list;
22     3 usages
23     private String categoriId;
24
25     no usages
26     > public String getCategoryID() { return categoriId; }
27
28     no usages
29     > public void setCategoryID(String categoriId) { this.categoriId = categoriId; }
30
31     public void setData(List<ProductCategories> list) {
32         this.list = list;
33         notifyDataSetChanged();
34     }
35     @NonNull
36     @Override
37     public ProductCategoriesViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
38         View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_rcv_discover, parent, false);
39         return new ProductCategoriesViewHolder(view);
40     }
41     @Override
42     public void onBindViewHolder(@NonNull ProductCategoriesViewHolder holder, int position) {
43         ProductCategories productCategories = list.get(position);
44         if (productCategories == null) {
```

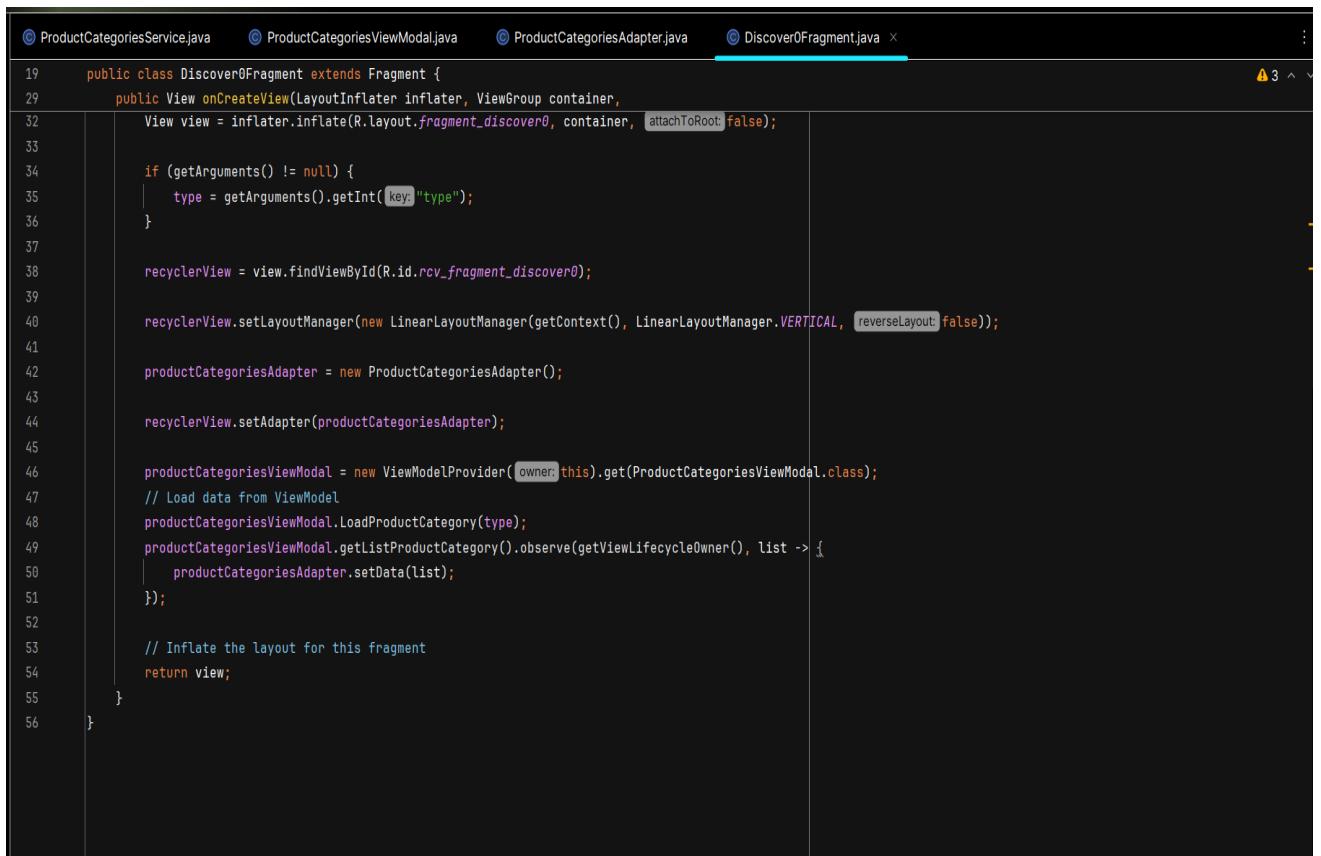
```
ProductCategoriesService.java ProductCategoriesViewModal.java ProductCategoriesAdapter.java
18 public class ProductCategoriesAdapter extends RecyclerView.Adapter<ProductCategoriesAdapter.ProductCategoriesViewHolder> {
38     return new ProductCategoriesViewHolder(view);
39 }
40
41     @Override
42     public void onBindViewHolder(@NonNull ProductCategoriesViewHolder holder, int position) {
43         ProductCategories productCategories = list.get(position);
44         if (productCategories == null) {
45             return;
46         }
47         holder.txtName_Discover.setText(productCategories.getCategoryName());
48         holder.txt_sl_Discover.setText(Integer.toString(productCategories.getQuantity()) + " items");
49         this.categoryId = productCategories.getCategoryId();
50
51         holder.itemView.setOnClickListener(v -> {
52             Intent intent = new Intent(holder.itemView.getContext(), ProductsActivity.class);
53             intent.putExtra("categoryId", productCategories.getCategoryId());
54             intent.putExtra("categoryName", productCategories.getCategoryName());
55             holder.itemView.getContext().startActivity(intent);
56         });
57     }
58
59     @Override
60     public int getItemCount() {
61         if (list != null) {
62             return list.size();
63         }
64         return 0;
65     }
66
67
68     4 usages
```

```
ProductCategoriesService.java ProductCategoriesViewModal.java ProductCategoriesAdapter.java
18 public class ProductCategoriesAdapter extends RecyclerView.Adapter<ProductCategoriesAdapter.ProductCategoriesViewHolder> {
42     public void onBindViewHolder(@NonNull ProductCategoriesViewHolder holder, int position) {
57
58     }
59
60     @Override
61     public int getItemCount() {
62         if (list != null) {
63             return list.size();
64         }
65         return 0;
66     }
67
68     4 usages
69     public class ProductCategoriesViewHolder extends RecyclerView.ViewHolder {
70         2 usages
71         TextView txtName_Discover, txt_sl_Discover;
72         1 usage
73         public ProductCategoriesViewHolder(@NonNull View itemView) {
74             super(itemView);
75             txtName_Discover = itemView.findViewById(R.id.txt_Nameitem_Discover);
76             txt_sl_Discover = itemView.findViewById(R.id.txt_sl_item_Discover);
77         }
78     }
79 }
```

+ File CategoriFragment:



```
1 package com.project.clothingstore.view.fragment.discover;
2
3 > import ...;
17
18
19 </> public class DiscoverOfFragment extends Fragment {
20     3 usages
21     private ProductCategoriesAdapter productCategoriesAdapter;
22     3 usages
23     private ProductCategoriesViewModal productCategoriesViewModal;
24
25     3 usages
26     private RecyclerView recyclerView;
27
28     2 usages
29     private int type;
30
31
32     @Override
33     @Override
34     public View onCreateView(LayoutInflater inflater, ViewGroup container,
35                             Bundle savedInstanceState) {
36
37         View view = inflater.inflate(R.layout.fragment_discover0, container, false);
38
39         if (getArguments() != null) {
40             type = getArguments().getInt("key: "type");
41         }
42
43         recyclerView = view.findViewById(R.id.rcv_fragment_discover0);
44
45         recyclerView.setLayoutManager(new LinearLayoutManager(getContext(), LinearLayoutManager.VERTICAL, false));
46
47         productCategoriesAdapter = new ProductCategoriesAdapter();
48
49         recyclerView.setAdapter(productCategoriesAdapter);
50
51         productCategoriesViewModal = new ViewModelProvider(owner: this).get(ProductCategoriesViewModal.class);
52         // Load data from ViewModel
53         productCategoriesViewModal.LoadProductCategory(type);
54         productCategoriesViewModal.getListProductCategory().observe(getViewLifecycleOwner(), list -> {
55             productCategoriesAdapter.setData(list);
56         });
57
58         // Inflate the layout for this fragment
59         return view;
60     }
61 }
```



```
19     public class DiscoverOfFragment extends Fragment {
20
21         public View onCreateView(LayoutInflater inflater, ViewGroup container,
22
23             View view = inflater.inflate(R.layout.fragment_discover0, container, false);
24
25             if (getArguments() != null) {
26                 type = getArguments().getInt("key: "type");
27             }
28
29             recyclerView = view.findViewById(R.id.rcv_fragment_discover0);
30
31             recyclerView.setLayoutManager(new LinearLayoutManager(getContext(), LinearLayoutManager.VERTICAL, false));
32
33             productCategoriesAdapter = new ProductCategoriesAdapter();
34
35             recyclerView.setAdapter(productCategoriesAdapter);
36
37             productCategoriesViewModal = new ViewModelProvider(owner: this).get(ProductCategoriesViewModal.class);
38             // Load data from ViewModel
39             productCategoriesViewModal.LoadProductCategory(type);
40             productCategoriesViewModal.getListProductCategory().observe(getViewLifecycleOwner(), list -> {
41                 productCategoriesAdapter.setData(list);
42             });
43
44             // Inflate the layout for this fragment
45             return view;
46         }
47     }
48 }
```

3.4.9. Chức năng tìm kiếm

+ File SearchbarFragment

```
⑤ SearchBarFragment.java x
1 package com.project.clothingstore.view.fragment.Filter;
2
3 >import ...
22
23
24 <> public class SearchBarFragment extends Fragment {
25     2 usages
26     ImageButton imgbtn_find, imgbtn_filter;
27     7 usages
28     EditText edt_find;
29     3 usages
30     String productName;
31     4 usages
32     // Interface callback
33     7 usages 2 implementations
34     public interface OnFilterButtonClickListener {
35         1 usage 2 implementations
36         void onFilterButtonClick();
37     }
38
39     4 usages
40     private OnFilterButtonClickListener listener;
41
42     @Override
43     public void onAttach(@NotNull Context context) {
44         super.onAttach(context);
45
46         // Đầu tiên lấy listener từ Fragment cha nếu có
47         if (getParentFragment() instanceof OnFilterButtonClickListener) {
48             listener = (OnFilterButtonClickListener) getParentFragment();
49         }
50
51         // Nếu không có Fragment cha, kiểm tra context chính là activity có implement interface
52     }
53
54     @Override
55     public View onCreateView(LayoutInflater inflater, ViewGroup container,
56                             Bundle savedInstanceState) {
57         View view = inflater.inflate(R.layout.fragment_search_bar, container, false);
58
59         imgbtn_find = view.findViewById(R.id.imgbtn_find_search_bar);
60         imgbtn_filter = view.findViewById(R.id.imgbtn_filter_search_bar);
61         edt_find = view.findViewById(R.id.edt_find_discover_search_bar);
62
63         if (getArguments() != null) {
64             productName = getArguments().getString("productName");
65             if (productName != null) {
66                 edt_find.setText(productName);
67             }
68         }
69     }
70 }
```

```
Project Alt+1 SearchBarFragment.java x
1 package com.project.clothingstore.view.fragment.Filter;
2
3 >import ...
22
23
24 <> public class SearchBarFragment extends Fragment {
25     2 usages
26     ImageButton imgbtn_find, imgbtn_filter;
27     7 usages
28     EditText edt_find;
29     3 usages
30     String productName;
31     4 usages
32     // Interface callback
33     7 usages 2 implementations
34     public interface OnFilterButtonClickListener {
35         1 usage 2 implementations
36         void onFilterButtonClick();
37     }
38
39     4 usages
40     private OnFilterButtonClickListener listener;
41
42     @Override
43     public void onAttach(@NotNull Context context) {
44         super.onAttach(context);
45
46         // Đầu tiên lấy listener từ Fragment cha nếu có
47         if (getParentFragment() instanceof OnFilterButtonClickListener) {
48             listener = (OnFilterButtonClickListener) getParentFragment();
49         }
50
51         // Nếu không có Fragment cha, kiểm tra context chính là activity có implement interface
52         else if (context instanceof OnFilterButtonClickListener) {
53             listener = (OnFilterButtonClickListener) context;
54         }
55     }
56
57     @Override
58     public View onCreateView(LayoutInflater inflater, ViewGroup container,
59                             Bundle savedInstanceState) {
60         View view = inflater.inflate(R.layout.fragment_search_bar, container, false);
61
62         imgbtn_find = view.findViewById(R.id.imgbtn_find_search_bar);
63         imgbtn_filter = view.findViewById(R.id.imgbtn_filter_search_bar);
64         edt_find = view.findViewById(R.id.edt_find_discover_search_bar);
65
66         if (getArguments() != null) {
67             productName = getArguments().getString("productName");
68             if (productName != null) {
69                 edt_find.setText(productName);
70             }
71         }
72     }
73 }
```

```
① SearchBarFragment.java x
24  public class SearchBarFragment extends Fragment {
52      public View onCreateView(LayoutInflater inflater, ViewGroup container,
62          if (productName != null) {
63              edt_find.setText(productName);
64          }
65      }
66
67
68      imgbtn_filter.setOnClickListener(v -> {
69          if (listener != null) {
70              listener.onFilterButtonClick();
71          }
72      });
73      edt_find.setOnEditorActionListener(new TextView.OnEditorActionListener() {
74          @Override
75          public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
76              if (actionId == EditorInfo.IME_ACTION_SEARCH ||
77                  (event != null & event.getKeyCode() == KeyEvent.KEYCODE_ENTER && event.getAction() == KeyEvent.ACTION_DOWN)) {
78                  String searchText = edt_find.getText().toString().trim();
79                  if (!searchText.isEmpty()) {
80                      Intent intent = new Intent(getActivity(), ProductsActivity.class);
81                      intent.putExtra("name", searchText);
82                      startActivity(intent);
83                  } else {
84                      edt_find.setError("Vui lòng nhập từ khóa tìm kiếm");
85                  }
86              }
87              return true;
88          }
89      });
90  });
91
```

```
① SearchBarFragment.java x
24  public class SearchBarFragment extends Fragment {
52      public View onCreateView(LayoutInflater inflater, ViewGroup container,
73          edt_find.setOnEditorActionListener(new TextView.OnEditorActionListener() {
75          public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
85              }
86              return true;
87          }
88          return false;
89      });
90  );
91
92
93      imgbtn_find.setOnClickListener(v -> {
94          String searchText = edt_find.getText().toString();
95          if (!searchText.isEmpty()) {
96              Intent intent = new Intent(getActivity(), ProductsActivity.class);
97              intent.putExtra("name", searchText);
98              startActivity(intent);
99          } else {
100              edt_find.setError("Vui lòng nhập từ khóa tìm kiếm");
101          }
102      });
103
104      return view;
105  }
106 }
```

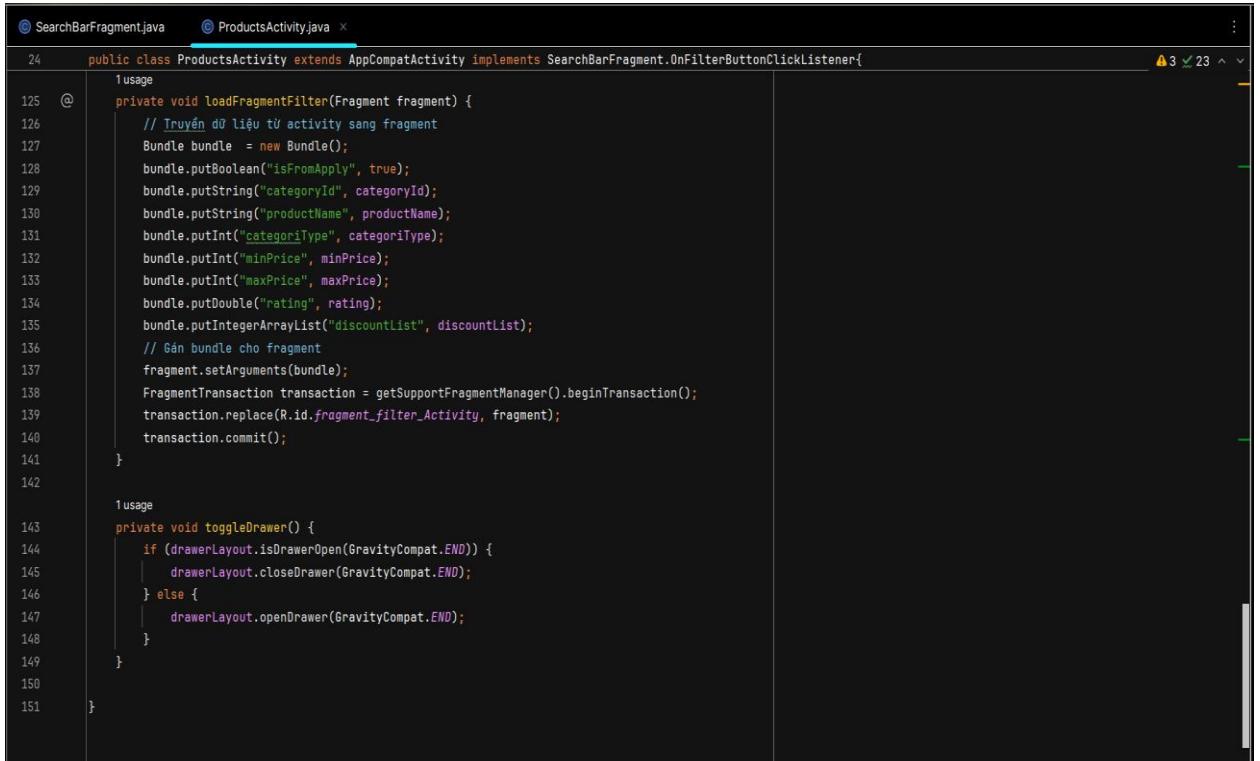
+ File Product Activity:

```
② SearchBarFragment.java ③ ProductsActivity.java ×
1 package com.project.clothingstore.view.activity.product;
2
3 > import ...
23
24 public class ProductsActivity extends AppCompatActivity implements SearchBarFragment.OnFilterButtonClickListener{
25     2 usages
26     ImageButton imgbtn_back;
27     3 usages
28     private String categoryId, productName;
29     3 usages
30     int categoriType;
31     5 usages
32     int minPrice, maxPrice;
33     3 usages
34     double rating;
35     3 usages
36     ArrayList<Integer> discountList;
37
38     4 usages
39     private DrawerLayout drawerLayout;
40     @Override
41     protected void onCreate(Bundle savedInstanceState) {
42         super.onCreate(savedInstanceState);
43         setContentView(R.layout.activity_products);
44         txt_title = findViewById(R.id.txt_title_products);
45         imgbtn_back = findViewById(R.id.btn_back_products);
46         drawerLayout = findViewById(R.id.drawer_layout_activity);
47
48         Intent intent = getIntent();
49         categoryId = intent.getStringExtra(name: "categoryId");
50
51         productName = intent.getStringExtra(name: "productName");
52
53         categoriType = intent.getIntExtra(name: "categoriType", defaultValue: -1);
54         String minPriceStr = intent.getStringExtra(name: "minPrice");
55         String maxPriceStr = intent.getStringExtra(name: "maxPrice");
56         if (minPriceStr != null && !minPriceStr.isEmpty()) {
57             try {
58                 minPrice = Integer.parseInt(minPriceStr);
59             } catch (NumberFormatException e) {
60                 minPrice = -1; // Giá trị mặc định nếu không thể chuyển đổi
61             }
62         } else {
63             minPrice = -1; // Giá trị mặc định nếu không có dữ liệu
64         }
65         if (maxPriceStr != null && !maxPriceStr.isEmpty()) {
66             try {
67                 maxPrice = Integer.parseInt(maxPriceStr);
68             } catch (NumberFormatException e) {
69                 maxPrice = -1; // Giá trị mặc định nếu không thể chuyển đổi
70             }
71         } else {
72             maxPrice = -1; // Giá trị mặc định nếu không có dữ liệu
73         }
74         rating = intent.getIntExtra(name: "rating", defaultValue: -1); // Nếu không có thì là -1
75         discountList = intent.getIntegerArrayListExtra(name: "discountList");
76
77     }
78 }
```

```
② SearchBarFragment.java ③ ProductsActivity.java ×
24 public class ProductsActivity extends AppCompatActivity implements SearchBarFragment.OnFilterButtonClickListener{
25     protected void onCreate(Bundle savedInstanceState) {
26         drawerLayout = findViewById(R.id.drawer_layout_activity);
27
28         Intent intent = getIntent();
29         categoryId = intent.getStringExtra(name: "categoryId");
30
31         productName = intent.getStringExtra(name: "productName");
32
33         categoriType = intent.getIntExtra(name: "categoriType", defaultValue: -1);
34         String minPriceStr = intent.getStringExtra(name: "minPrice");
35         String maxPriceStr = intent.getStringExtra(name: "maxPrice");
36         if (minPriceStr != null && !minPriceStr.isEmpty()) {
37             try {
38                 minPrice = Integer.parseInt(minPriceStr);
39             } catch (NumberFormatException e) {
40                 minPrice = -1; // Giá trị mặc định nếu không thể chuyển đổi
41             }
42         } else {
43             minPrice = -1; // Giá trị mặc định nếu không có dữ liệu
44         }
45         if (maxPriceStr != null && !maxPriceStr.isEmpty()) {
46             try {
47                 maxPrice = Integer.parseInt(maxPriceStr);
48             } catch (NumberFormatException e) {
49                 maxPrice = -1; // Giá trị mặc định nếu không thể chuyển đổi
50             }
51         } else {
52             maxPrice = -1; // Giá trị mặc định nếu không có dữ liệu
53         }
54         rating = intent.getIntExtra(name: "rating", defaultValue: -1); // Nếu không có thì là -1
55         discountList = intent.getIntegerArrayListExtra(name: "discountList");
56
57     }
58 }
```

```
© SearchBarFragment.java   © ProductsActivity.java ×
24     public class ProductsActivity extends AppCompatActivity implements SearchBarFragment.OnFilterButtonClickListener{
34         protected void onCreate(Bundle savedInstanceState) {
69
70
71             if (savedInstanceState == null) {
72                 loadFragmentProduct(new ProductFragment(), categoryId);
73                 loadFragmentSearch(new SearchBarFragment());
74                 loadFragmentFilter(new FilterFragment());
75             }
76
77
78
79             // Xử lý sự kiện nhấn nút quay lại
80             // Xử lý sự kiện nhấn nút quay lại
81             imgbtn_back.setOnClickListener(v -> {
82                 Intent intentT = new Intent(getApplicationContext(), ProductsActivity.this, MainActivity.class);
83                 intentT.putExtra("name", "openSearchFragment", true); // Thêm thông báo mở SearchFragment
84                 startActivity(intentT);
85                 finish(); // Đóng ProductsActivity
86             });
87
88
89         }
90
91         1usage
92         @Override
93         > public void onFilterButtonClick() {
94             toggleDrawer(); // Gọi mở/dòng drawer từ fragment con
95
96         1usage
97         @ private void loadFragmentProduct(Fragment fragment, String cateId) {
98
99             // Truyền dữ liệu từ activity sang fragment
100            Bundle bundle = new Bundle();
101            bundle.putString("categoryId", cateId);
102            bundle.putString("productName", productName);
103            bundle.putInt("categoryType", categoryType);
104            bundle.putInt("minPrice", minPrice);
105            bundle.putInt("maxPrice", maxPrice);
106            bundle.putDouble("rating", rating);
107            bundle.putIntegerArrayList("discountList", discountList);
108            // Gán bundle cho fragment
109            fragment.setArguments(bundle);
110            // Thực hiện giao dịch fragment
111            FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
112            transaction.replace(R.id.fragment_product_items, fragment);
113            transaction.commit();
114        }
115
116         1usage
117         @ private void loadFragmentSearch(Fragment fragment) {
118             Bundle bundle = new Bundle();
119             bundle.putBoolean("isSearch", true);
120             bundle.putString("productName", productName);
121             fragment.setArguments(bundle);
122             FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
123             transaction.replace(R.id.fragment_search_filter_activity_products, fragment);
124             transaction.commit();
125
126         1usage
```

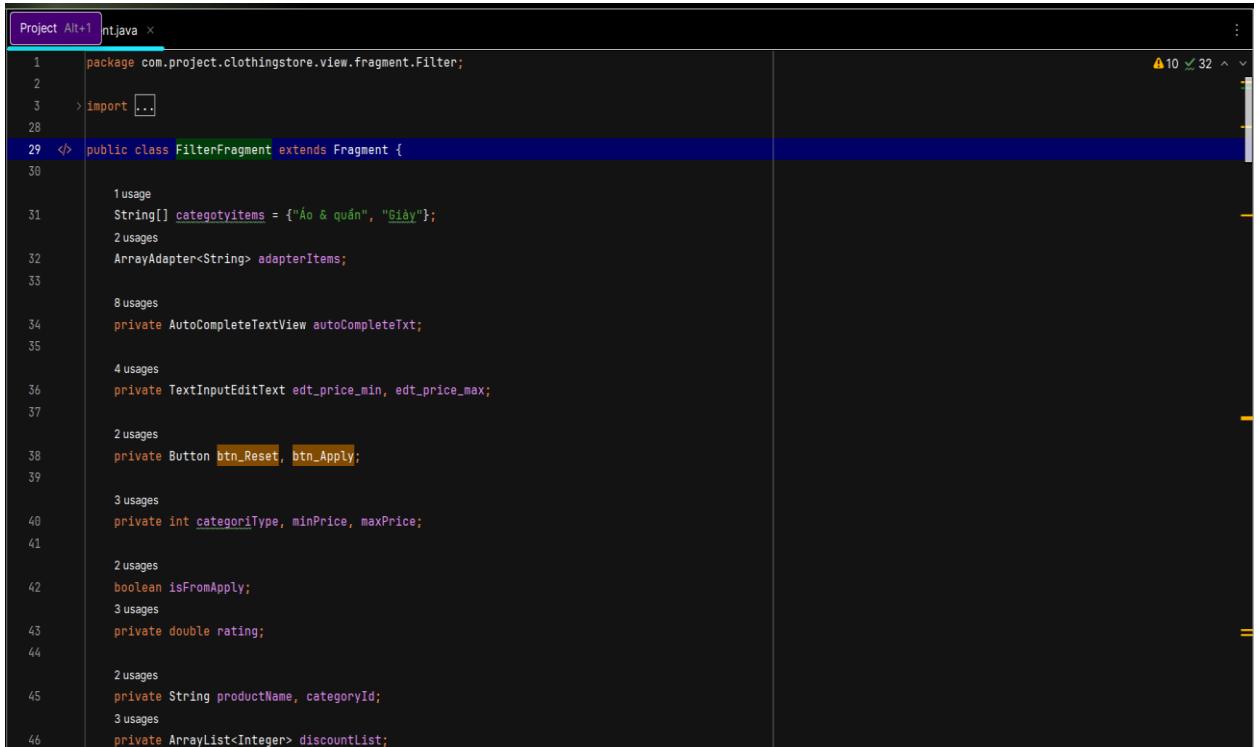
```
Project Alt+1 fragment.java   © ProductsActivity.java ×
24     public class ProductsActivity extends AppCompatActivity implements SearchBarFragment.OnFilterButtonClickListener{
96         private void loadFragmentProduct(Fragment fragment, String cateId) {
97
98             // Truyền dữ liệu từ activity sang fragment
99             Bundle bundle = new Bundle();
100            bundle.putString("categoryId", cateId);
101            bundle.putString("productName", productName);
102            bundle.putInt("categoryType", categoryType);
103            bundle.putInt("minPrice", minPrice);
104            bundle.putInt("maxPrice", maxPrice);
105            bundle.putDouble("rating", rating);
106            bundle.putIntegerArrayList("discountList", discountList);
107            // Gán bundle cho fragment
108            fragment.setArguments(bundle);
109            // Thực hiện giao dịch fragment
110            FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
111            transaction.replace(R.id.fragment_product_items, fragment);
112            transaction.commit();
113        }
114
115         1usage
116         @ private void loadFragmentSearch(Fragment fragment) {
117             Bundle bundle = new Bundle();
118             bundle.putBoolean("isSearch", true);
119             bundle.putString("productName", productName);
120             fragment.setArguments(bundle);
121             FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
122             transaction.replace(R.id.fragment_search_filter_activity_products, fragment);
123             transaction.commit();
124
125         1usage
```



```
24     public class ProductsActivity extends AppCompatActivity implements SearchBarFragment.OnFilterButtonClickListener{
125     @
126     private void loadFragmentFilter(Fragment fragment) {
127         // Truyền dữ liệu từ activity sang fragment
128         Bundle bundle = new Bundle();
129         bundle.putBoolean("isFromApply", true);
130         bundle.putString("categoryId", categoryId);
131         bundle.putString("productName", productName);
132         bundle.putInt("categoryType", categoryType);
133         bundle.putInt("minPrice", minPrice);
134         bundle.putInt("maxPrice", maxPrice);
135         bundle.putDouble("rating", rating);
136         bundle.putIntegerArrayList("discountList", discountList);
137         // Gán bundle cho fragment
138         fragment.setArguments(bundle);
139         FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
140         transaction.replace(R.id.fragment_filter_Activity, fragment);
141         transaction.commit();
142     }
143
144     private void toggleDrawer() {
145         if (drawerLayout.isDrawerOpen(GravityCompat.END)) {
146             drawerLayout.closeDrawer(GravityCompat.END);
147         } else {
148             drawerLayout.openDrawer(GravityCompat.END);
149         }
150     }
151 }
```

3.4.10. Chức năng lọc sản phẩm

+ File Filter Fragment



```
1 package com.project.clothingstore.view.fragment;
2
3 > import ...
28
29 <public class FilterFragment extends Fragment {
30
31     1 usage
32     String[] categoryItems = {"Áo & quần", "Giày"};
33     2 usages
34     ArrayAdapter<String> adapterItems;
35
36     8 usages
37     private AutoCompleteTextView autoCompleteTxt;
38
39     4 usages
40     private TextInputEditText edt_price_min, edt_price_max;
41
42     2 usages
43     private Button btn_Reset, btn_Apply;
44
45     3 usages
46     private int categoryType, minPrice, maxPrice;
```

```
Project Alt+1 ntjava x
29  public class FilterFragment extends Fragment {
30      5 usages
31      Set<Integer> selectedStar = new HashSet<>();
32      4 usages
33      Set<Integer> selectedDiscount = new HashSet<>();
34
35      9 usages
36      List<FrameLayout> framesstar = new ArrayList<>();
37      8 usages
38      List<FrameLayout> framesdiscount = new ArrayList<>();
39      @Override
40      @Q @
41      public View onCreateView(LayoutInflater inflater, ViewGroup container,
42          Bundle savedInstanceState) {
43          View view = inflater.inflate(R.layout.fragment_filter, container, false);
44
45          autoCompleteTxt = view.findViewById(R.id.auto_category_filter);
46          adapterItems = new ArrayAdapter<>(getContext(), R.layout.hvq_list_items, categoryItems);
47          autoCompleteTxt.setAdapter(adapterItems);
48
49          autoCompleteTxt.setOnItemSelectedListener((parent, v, position, id) -> {
50              String item = parent.getItemAtPosition(position).toString();
51              // Có thể lưu lại category đã chọn nếu cần
52          });
53
54          if (getArguments() != null) {
55
56              isFromApply = getArguments().getBoolean("isFromApply", defaultValue: false);
57              categoryId = getArguments().getString("categoryId");
58              productName = getArguments().getString("productName");
59              categoriType = getArguments().getInt("categoriType", defaultValue: -1);
60              minPrice = getArguments().getInt("minPrice", defaultValue: -1);
61
62          }
63
64      }
65
66      if (getArguments() != null) {
67
68          isFromApply = getArguments().getBoolean("isFromApply", defaultValue: false);
69          categoryId = getArguments().getString("categoryId");
70          productName = getArguments().getString("productName");
71          categoriType = getArguments().getInt("categoriType", defaultValue: -1);
72          minPrice = getArguments().getInt("minPrice", defaultValue: -1);
73
74      }
75
76  }
```

```
Project Alt+1 ntjava x
29  public class FilterFragment extends Fragment {
30      5 usages
31      Set<Integer> selectedStar = new HashSet<>();
32      4 usages
33      Set<Integer> selectedDiscount = new HashSet<>();
34
35      9 usages
36      List<FrameLayout> framesstar = new ArrayList<>();
37      8 usages
38      List<FrameLayout> framesdiscount = new ArrayList<>();
39      @Override
40      @Q @
41      public View onCreateView(LayoutInflater inflater, ViewGroup container,
42          Bundle savedInstanceState) {
43          View view = inflater.inflate(R.layout.fragment_filter, container, false);
44
45          autoCompleteTxt = view.findViewById(R.id.auto_category_filter);
46          adapterItems = new ArrayAdapter<>(getContext(), R.layout.hvq_list_items, categoryItems);
47          autoCompleteTxt.setAdapter(adapterItems);
48
49          autoCompleteTxt.setOnItemSelectedListener((parent, v, position, id) -> {
50              String item = parent.getItemAtPosition(position).toString();
51              // Có thể lưu lại category đã chọn nếu cần
52          });
53
54          if (getArguments() != null) {
55
56              isFromApply = getArguments().getBoolean("isFromApply", defaultValue: false);
57              categoryId = getArguments().getString("categoryId");
58              productName = getArguments().getString("productName");
59              categoriType = getArguments().getInt("categoriType", defaultValue: -1);
60              minPrice = getArguments().getInt("minPrice", defaultValue: -1);
61              maxPrice = getArguments().getInt("maxPrice", defaultValue: -1);
62              rating = getArguments().getDouble("rating", defaultValue: -1); // Nếu không có thì là -1
63              discountList = getArguments().getIntegerArrayList("discountList");
64
65          }
66
67          // Khởi tạo frame rating (sao)
68          framesstar.add(view.findViewById(R.id.star_frame1));
69          framesstar.add(view.findViewById(R.id.star_frame2));
70          framesstar.add(view.findViewById(R.id.star_frame3));
71          framesstar.add(view.findViewById(R.id.star_frame4));
72          framesstar.add(view.findViewById(R.id.star_frame5));
73
74          // Khởi tạo frame discount
75          framesdiscount.add(view.findViewById(R.id.discount_frame0));
76          framesdiscount.add(view.findViewById(R.id.discount_frame1));
77          framesdiscount.add(view.findViewById(R.id.discount_frame2));
78          framesdiscount.add(view.findViewById(R.id.discount_frame3));
79
80          edt_price_min = view.findViewById(R.id.txt_PriceMin);
81          edt_price_max = view.findViewById(R.id.txt_PriceMax);
82
83          btn_Reset = view.findViewById(R.id.btn_reset_filter);
84          btn_Apply = view.findViewById(R.id.btn_apply_filter);
85
86          // Gán sự kiện click
87      }
88
89  }
```

```
⑥ FilterFragment.java x
29  public class FilterFragment extends Fragment {
56      public View onCreateView(LayoutInflater inflater, ViewGroup container,
100     btn_Apply = view.findViewById(R.id.btn_apply_filter);
101
102     // Gán sự kiện click
103     setupStarFrameClick(framesstar, selectedStar); // chỉ chọn 1 sao
104     setupFrameClick(framesdiscount, selectedDiscount); // nhiều discount
105
106     // Xử lý nút Reset
107     btn_Reset.setOnClickListener(v -> {
108         autoCompleteTxt.setText("");
109
110         edt_price_min.setText("");
111         edt_price_max.setText("");
112
113         selectedStar.clear();
114         for (FrameLayout f : framesstar) f.setSelected(false);
115
116         selectedDiscount.clear();
117         for (FrameLayout f : framesdiscount) f.setSelected(false);
118
119         Toast.makeText(getContext(), "Đã đặt lại bộ lọc", Toast.LENGTH_SHORT).show();
120     });
121
122     // Xử lý nút Apply
123     btn_Apply.setOnClickListener(v -> {
124         String category = autoCompleteTxt.getText().toString().trim();
125         int categoryType;
126         if (category.equals("Áo & quần")) {
127             categoryType = 0;
128         } else if (category.equals("Giày")) {
129             categoryType = 1;
130         }
131
132
133
134         String minPrice = edt_price_min.getText().toString().trim();
135         String maxPrice = edt_price_max.getText().toString().trim();
136
137         Integer selectedIndex = selectedStar.isEmpty() ? null : selectedStar.iterator().next();
138         ArrayList<Integer> selectedDiscountList = new ArrayList<>(selectedDiscount);
139
140         // Khởi tạo Intent
141         Intent intent = new Intent(requireContext(), ProductsActivity.class);
142
143         // Truyền dữ liệu qua Intent
144         intent.putExtra("categoryId", categoryId);
145         intent.putExtra("categoryType", categoryType);
146         intent.putExtra("minPrice", minPrice);
147         intent.putExtra("maxPrice", maxPrice);
148         intent.putExtra("productName", productName); // Truyền tên sản phẩm nếu có
149         if (selectedIndex != null) {
150             intent.putExtra("rating", value: selectedIndex + 1); // 1-5 sao
151         }
152
153         intent.putIntegerArrayListExtra("discountList", selectedDiscountList);

```

```
⑥ FilterFragment.java x
29  public class FilterFragment extends Fragment {
56      public View onCreateView(LayoutInflater inflater, ViewGroup container,
123     btn_Apply.setOnClickListener(v -> {
124
125         int categoryType;
126         if (category.equals("Áo & quần")) {
127             categoryType = 0;
128         } else if (category.equals("Giày")) {
129             categoryType = 1;
130         } else {
131             categoryType = -1; // Không chọn
132         }
133
134         String minPrice = edt_price_min.getText().toString().trim();
135         String maxPrice = edt_price_max.getText().toString().trim();
136
137         Integer selectedIndex = selectedStar.isEmpty() ? null : selectedStar.iterator().next();
138         ArrayList<Integer> selectedDiscountList = new ArrayList<>(selectedDiscount);
139
140         // Khởi tạo Intent
141         Intent intent = new Intent(requireContext(), ProductsActivity.class);
142
143         // Truyền dữ liệu qua Intent
144         intent.putExtra("categoryId", categoryId);
145         intent.putExtra("categoryType", categoryType);
146         intent.putExtra("minPrice", minPrice);
147         intent.putExtra("maxPrice", maxPrice);
148         intent.putExtra("productName", productName); // Truyền tên sản phẩm nếu có
149         if (selectedIndex != null) {
150             intent.putExtra("rating", value: selectedIndex + 1); // 1-5 sao
151         }
152
153         intent.putIntegerArrayListExtra("discountList", selectedDiscountList);

```

```
⑥ FilterFragment.java ×
29  public class FilterFragment extends Fragment {
56      public View onCreateView(LayoutInflater inflater, ViewGroup container,
123          btn_Apply.setOnClickListener(v -> {
152              intent.putIntegerArrayListExtra("name: "discountList", selectedDiscountList);
153
154              // Chuyển sang ProductsActivity
155              startActivity(intent);
156          });
157          // Setup giá trị mặc định cho các trường Filter
158          setupfragmentFilter();
159
160
161
162
163
164      return view;
165  }
166
167
168  1 usage
169  @
170  private void setupFrameClick(List<FrameLayout> frames, Set<Integer> selectedSet) {
171      for (int i = 0; i < frames.size(); i++) {
172          final int index = i;
173          FrameLayout frame = frames.get(i);
174          frame.setOnClickListener(v -> {
175              if (selectedSet.contains(index)) {
176                  selectedSet.remove(index);
177                  v.setSelected(false);
178              } else {
179                  selectedSet.add(index);
180                  v.setSelected(true);
181              }
182          });
183      }
184  }
185  1 usage
186  @
187  private void setupStarFrameClick(List<FrameLayout> frames, Set<Integer> selectedSet) {
188      for (int i = 0; i < frames.size(); i++) {
189          final int index = i;
190          FrameLayout frame = frames.get(i);
191          frame.setOnClickListener(v -> {
192              // Nếu đã chọn thì bỏ chọn tất cả
193              if (selectedSet.contains(index)) {
194                  selectedSet.clear();
195                  for (FrameLayout f : frames) f.setSelected(false);
196              } else {
197                  // Bỏ chọn tất cả trước
198                  selectedSet.clear();
199                  for (FrameLayout f : frames) f.setSelected(false);
200
201                  // Chọn cái mới
202                  selectedSet.add(index);
203                  v.setSelected(true);
204              }
205          });
206      }
207  }
208  1 usage
209  private void setupfragmentFilter(){
210      if(isFromApply){
211          // Setup giá trị mặc định cho các trường Filter
212          if (categoryType == 0) {
```

```
⑥ FilterFragment.java ×
29  public class FilterFragment extends Fragment {
56      public View onCreateView(LayoutInflater inflater, ViewGroup container,
123          btn_Apply.setOnClickListener(v -> {
152              intent.putIntegerArrayListExtra("name: "discountList", selectedDiscountList);
153
154              // Chuyển sang ProductsActivity
155              startActivity(intent);
156          });
157          // Setup giá trị mặc định cho các trường Filter
158          setupfragmentFilter();
159
160
161
162
163
164      return view;
165  }
166
167
168  1 usage
169  @
170  private void setupFrameClick(List<FrameLayout> frames, Set<Integer> selectedSet) {
171      for (int i = 0; i < frames.size(); i++) {
172          final int index = i;
173          FrameLayout frame = frames.get(i);
174          frame.setOnClickListener(v -> {
175              if (selectedSet.contains(index)) {
176                  selectedSet.remove(index);
177                  v.setSelected(false);
178              } else {
179                  selectedSet.add(index);
180                  v.setSelected(true);
181              }
182          });
183      }
184  }
185  1 usage
186  @
187  private void setupStarFrameClick(List<FrameLayout> frames, Set<Integer> selectedSet) {
188      for (int i = 0; i < frames.size(); i++) {
189          final int index = i;
190          FrameLayout frame = frames.get(i);
191          frame.setOnClickListener(v -> {
192              // Nếu đã chọn thì bỏ chọn tất cả
193              if (selectedSet.contains(index)) {
194                  selectedSet.clear();
195                  for (FrameLayout f : frames) f.setSelected(false);
196              } else {
197                  // Bỏ chọn tất cả trước
198                  selectedSet.clear();
199                  for (FrameLayout f : frames) f.setSelected(false);
200
201                  // Chọn cái mới
202                  selectedSet.add(index);
203                  v.setSelected(true);
204              }
205          });
206      }
207  }
208  1 usage
209  private void setupfragmentFilter(){
210      if(isFromApply){
211          // Setup giá trị mặc định cho các trường Filter
212          if (categoryType == 0) {
```

```
② FilterFragment.java x
Commit Alt+O
public class FilterFragment extends Fragment {
    private void setupfragmentFilter(){
        205     if(isFromApply){
        206         // Setup giá trị mặc định cho các trường Filter
        207         if (categoryType == 0) {
        208             autoCompleteTxt.setText("Áo & quần", filter: false);
        209         } else if (categoryType == 1) {
        210             autoCompleteTxt.setText("Giày", filter: false);
        211         } else {
        212             autoCompleteTxt.setText("", filter: false);
        213         }
        214
        215         if (minPrice != -1) {
        216             edt_price_min.setText(String.valueOf(minPrice));
        217         }
        218         if (maxPrice != -1) {
        219             edt_price_max.setText(String.valueOf(maxPrice));
        220         }
        221         if (rating != -1) {
        222             int ratingIndex = (int) rating - 1;
        223             if (ratingIndex >= 0 && ratingIndex < framesstar.size()) {
        224                 selectedStar.add(ratingIndex);
        225                 framesstar.get(ratingIndex).setSelected(true);
        226             } else {
        227                 Log.w("FilterFragment", msg:"Rating index ngoài giới hạn: " + ratingIndex);
        228             }
        229         }
        230
        231         if (discountList != null) {
        232             for (int discount : discountList) {
        233                 if (discount >= 0 && discount < framesdiscount.size()) {
        234                     selectedDiscount.add(discount);
        235                     framesdiscount.get(discount).setSelected(true);
        236                 } else {
        237                     Log.w("FilterFragment", msg:"Discount index ngoài giới hạn: " + discount);
        238                 }
        239             }
        240         }
        241     }
        242 }
243
244 }
```

```
② FilterFragment.java x
Commit Alt+O
public class FilterFragment extends Fragment {
    private void setupfragmentFilter(){
        29         if (rating != -1) {
        30             int ratingIndex = (int) rating - 1;
        31             if (ratingIndex >= 0 && ratingIndex < framesstar.size()) {
        32                 selectedStar.add(ratingIndex);
        33                 framesstar.get(ratingIndex).setSelected(true);
        34             } else {
        35                 Log.w("FilterFragment", msg:"Rating index ngoài giới hạn: " + ratingIndex);
        36             }
        37
        38         if (discountList != null) {
        39             for (int discount : discountList) {
        40                 if (discount >= 0 && discount < framesdiscount.size()) {
        41                     selectedDiscount.add(discount);
        42                     framesdiscount.get(discount).setSelected(true);
        43                 } else {
        44                     Log.w("FilterFragment", msg:"Discount index ngoài giới hạn: " + discount);
        45                 }
        46             }
        47         }
        48     }
        49 }
```

3.4.11. Chức năng hiển thị chi tiết sản phẩm

+ File ProductDetailActivity.java:

```
package com.project.clothingstore.view.activity;

import ...

public class ProductDetailActivity extends AppCompatActivity {
    6 usages
    private ProductDetailViewModel viewModel;
    7 usages
    private String productId;
    3 usages
    private UserViewModel userViewModel;
    7 usages
    private Button btnAddToCart;
    5 usages
    private String cartId;
    4 usages
    private boolean isLoggedIn = false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_product_detail);
        viewModel = new ViewModelProvider(owner: this).get(ProductDetailViewModel.class);
        Intent intent = getIntent();
        if (intent != null) {
            productId = intent.getStringExtra(name: "productId");
            if (productId != null) {
                viewModel.loadProductDetails(productId);
            }
        }
        userViewModel = new ViewModelProvider(owner: this).get(UserViewModel.class);
        checkLoginStatus();
    }
}
```

```

    checkLoginStatus();
    Log.d( tag: "FeatureProductAdapter", msg: "Clicked on product: " + productId);
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);
    if (getSupportActionBar() != null) {
        getSupportActionBar().setDisplayShowTitleEnabled(false);
    }
    ImageButton btnBack = findViewById(R.id.btnBack);
    btnBack.setOnClickListener(v -> onBackPressed());
    btnAddToCart = findViewById(R.id.btnAddToCart);
    btnAddToCart.setOnClickListener(v -> handleAddToCart());
    loadFragments();
    viewModel.getProduct().observe( owner: this, product -> {
        if (product != null) {
            loadFragments(product);
        }
    });
}
1 usage
private void checkLoginStatus() {
    FirebaseAuth auth = FirebaseAuth.getInstance();
    if (auth.getCurrentUser() != null) {
        isLoggedIn = true;
        String uid = auth.getCurrentUser().getUid();
        userViewModel.fetchUserInfo(uid);
        userViewModel.getCurrentUser().observe( owner: this, user -> {
            if (user != null) {
                cartId = user.getCartId();
            }
        });
    }
}

```

```

private void checkLoginStatus() {
    FirebaseAuth auth = FirebaseAuth.getInstance();
    if (auth.getCurrentUser() != null) {
        isLoggedIn = true;
        String uid = auth.getCurrentUser().getUid();
        userViewModel.fetchUserInfo(uid);
        userViewModel.getCurrentUser().observe( owner: this, user -> {
            if (user != null) {
                cartId = user.getCartId();
            }
        });
    } else {
        isLoggedIn = false;
        updateAddToCartButton();
    }
}
1 usage
private void updateAddToCartButton() {
    if (btnAddToCart != null) {
        if (isLoggedIn) {
            btnAddToCart.setText("Thêm vào giỏ hàng");
            btnAddToCart.setEnabled(true);
        } else {
            btnAddToCart.setText("Đăng nhập để thêm vào giỏ hàng");
            btnAddToCart.setEnabled(true);
        }
    }
}
1 usage

```

```
private void handleAddToCart() {
    if (isUserLoggedIn) {
        addToCart();
    } else {
        Intent loginIntent = new Intent(packageContext, AuthActivity.class);
        loginIntent.putExtra(name: "returnToProductDetail", value: true);
        loginIntent.putExtra(name: "productId", productId);
        startActivity(loginIntent);
        Toast.makeText(context: this, text: "Vui lòng đăng nhập để thêm sản phẩm vào giỏ hàng", LENGTH_SHORT).show();
    }
}

1 usage
private void loadFragments() {
    FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
    transaction.replace(R.id.productImagesContainer, new ProductImagesFragment());
    transaction.replace(R.id.productInfoContainer, new ProductInfoFragment());
    transaction.replace(R.id.productDescriptionContainer, new ProductDescriptionFragment());
    transaction.replace(R.id.productRatingsContainer, new ProductRatingsFragment());
    transaction.replace(R.id.similarProductsContainer, new SimilarProductsFragment());
    transaction.commit();
}

1 usage
private void loadFragments(Product product) {
    loadFragment(R.id.productImagesContainer, new ProductImagesFragment(), product.getImages().get(0));
    loadFragment(R.id.productInfoContainer, new ProductInfoFragment(), product.getProductName());
    loadFragment(R.id.productDescriptionContainer, new ProductDescriptionFragment(), product.getDescription());
    loadFragment(R.id.productRatingsContainer, new ProductRatingsFragment(), productId);
    loadFragment(R.id.similarProductsContainer, new SimilarProductsFragment(), product.getCategoryId());
}
```

```
private void loadFragment(int containerId, Fragment fragment, String data) {
    Bundle bundle = new Bundle();
    bundle.putString("data", data);
    fragment.setArguments(bundle);
    FragmentTransaction transaction = getSupportFragmentManager().beginTransaction();
    transaction.replace(containerId, fragment);
    transaction.commit();
}

1 usage
private void addToCart() {
    Product product = viewModel.getProduct().getValue();
    String selectedColor = viewModel.getSelectedColor().getValue();
    String selectedSize = viewModel.getSelectedSize().getValue();
    product.setProductId(productId);
    if (product == null) {
        Log.e(tag: "ProductDetail", msg: "Sản phẩm không tồn tại");
        Toast.makeText(context: this, text: "Sản phẩm không tồn tại!", Toast.LENGTH_SHORT).show();
        return;
    }
    if (selectedColor == null || selectedSize == null) {
        Log.e(tag: "ProductDetail", msg: "Chưa chọn màu hoặc size");
        Toast.makeText(context: this, text: "Vui lòng chọn màu sắc và kích cỡ!", Toast.LENGTH_SHORT).show();
        return;
    }
    boolean productAvailable = false;
    int availableQuantity = 0;
    List<Product.Variant> variants = product.getVariants();
    if (variants != null) {
        Log.d(tag: "ProductDetail", msg: "Số lượng variant: " + variants.size());
        for (Product.Variant variant : variants) {
            if (variant.getColor().equals(selectedColor) && variant.getSize().equals(selectedSize)) {
                productAvailable = true;
                availableQuantity = variant.getQuantity();
            }
        }
    }
    if (!productAvailable) {
        Log.e(tag: "ProductDetail", msg: "Sản phẩm không có sẵn");
        Toast.makeText(context: this, text: "Sản phẩm không có sẵn!", Toast.LENGTH_SHORT).show();
        return;
    }
    if (availableQuantity <= 0) {
        Log.e(tag: "ProductDetail", msg: "Số lượng hàng không hợp lệ");
        Toast.makeText(context: this, text: "Số lượng hàng không hợp lệ!", Toast.LENGTH_SHORT).show();
        return;
    }
    // Logic for adding product to cart
}

```

```
        int availableQuantity = 0;
        List<Product.Variant> variants = product.getVariants();
        if (variants != null) {
            Log.d( tag: "ProductDetail", msg: "Số lượng variant: " + variants.size());
            for (Product.Variant variant : variants) {
                if (variant.getColor().equals(selectedColor)) {
                    Log.d( tag: "ProductDetail", msg: "Tim thấy variant với màu: " + selectedColor);
                    List<Product.Variant.SizeQuantity> sizes = variant.getSizes();
                    if (sizes != null) {
                        Log.d( tag: "ProductDetail", msg: "Số lượng size: " + sizes.size());
                        for (Product.Variant.SizeQuantity sizeQty : sizes) {
                            if (sizeQty.getSize().equals(selectedSize)) {
                                Log.d( tag: "ProductDetail", msg: "Tim thấy size: " + selectedSize + " với số lượng: " + sizeQty.getQuantity());
                                if (sizeQty.getQuantity() > 0) {
                                    productAvailable = true;
                                    availableQuantity = sizeQty.getQuantity();
                                    Log.d( tag: "ProductDetail", msg: "Sản phẩm có sẵn, số lượng: " + availableQuantity);
                                } else {
                                    Log.d( tag: "ProductDetail", msg: "Sản phẩm hết hàng với size này");
                                }
                                break;
                            }
                        }
                    }
                }
            }
        }
    }
}
```

```
if (!productAvailable) {
    Toast.makeText( context: this, text: "Sản phẩm đã hết hàng với màu sắc và kích cỡ đã chọn!", Toast.LENGTH_SHORT).show();
    return;
}
Carts.cartItem.variant itemVariant = new Carts.cartItem.variant(selectedColor, selectedSize);
Carts.cartItem item = new Carts.cartItem(
    product.getProductId(),
    product.getProductName(),
    product.getImages().get(0),
    quantity: 1,
    product.getPrice(),
    itemVariant
);
CartItem cartItem = new CartItem(
    product.getProductId(),
    product.getProductName(),
    product.getImages().get(0),
    quantity: 1,
    product.getPrice(),
    itemVariant
);
CartRepository cartRepository = new CartRepository();
if (cartId == null || cartId.isEmpty()) {
    Toast.makeText( context: this, text: "Không thể thêm vào giỏ hàng. Vui lòng thử lại sau!", Toast.LENGTH_SHORT).show();
    return;
}
cartRepository.addToCart(cartId, cartItem, new CartRepository.CartOperationCallback() {
```

```
    CartRepository cartRepository = new CartRepository();
    if (cartId == null || cartId.isEmpty()) {
        Toast.makeText(context: this, text: "Không thể thêm vào giỏ hàng. Vui lòng thử lại sau!", Toast.LENGTH_SHORT).show();
        return;
    }
    cartRepository.addToCart(cartId, cartItem, new CartRepository.CartOperationCallback() {
        2 usages
        @Override
        public void onSuccess() {
            Toast.makeText(context: ProductDetailActivity.this, text: "Đã thêm vào giỏ hàng", Toast.LENGTH_SHORT).show();
        }
        3 usages
        @Override
        public void onFailure(Exception e) {
            Toast.makeText(context: ProductDetailActivity.this, text: "Lỗi: " + e.getMessage(), Toast.LENGTH_SHORT).show();
            Log.e(tag: "CartRepository", msg: "Error adding to cart", e);
        }
    });
}
```

+ FileProductDetailViewModel.java

```
package com.project.clothingstore.viewmodel;
> import ...

18 usages
public class ProductDetailViewModel extends ViewModel {
    3 usages
    private final ProductRepository productRepository;
    2 usages
    private final RatingRepository ratingRepository;
    2 usages
    private final UserRepository userRepository;

    4 usages
    private final MutableLiveData<Product> productLiveData = new MutableLiveData<>();
    2 usages
    private final MutableLiveData<List<String>> productImagesLiveData = new MutableLiveData<>();
    3 usages
    private final MutableLiveData<List<Rating>> ratingsLiveData = new MutableLiveData<>();
    3 usages
    private final MutableLiveData<List<Product>> similarProductsLiveData = new MutableLiveData<>();
    3 usages
    private final MutableLiveData<Map<String, User>> userMapLiveData = new MutableLiveData<>();
    7 usages
    private final MutableLiveData<String> selectedColor = new MutableLiveData<>();
    8 usages
    private final MutableLiveData<String> selectedSize = new MutableLiveData<>();
    3 usages
    private final MutableLiveData<Map<Integer, Integer>> ratingDistributionLiveData = new MutableLiveData<>();
    3 usages
    private final MutableLiveData<List<ColorItem>> colorsLiveData = new MutableLiveData<>();
    3 usages
    private final MutableLiveData<List<SizeItem>> sizesLiveData = new MutableLiveData<>();
```

```

private final MutableLiveData<List<SizeItem>> sizesLiveData = new MutableLiveData<>();
💡 1 usage
private List<Rating> originalRatings = new ArrayList<>();
no usages
public ProductDetailViewModel() {
    productRepository = new ProductRepository();
    ratingRepository = new RatingRepository();
    userRepository = new UserRepository();
}
💡 1 usage
public void loadProductDetails(String productId) {
    productRepository.getProductById(productId, product -> {
        if (product != null) {
            productLiveData.setValue(product);
            productImagesLiveData.setValue(product.getImages());
            extractColorsAndSizes(product);
            loadSimilarProducts(product.getCategoryId());
        }
    });
    ratingRepository.getRatingsByProductId(productId, ratings -> {
        if (ratings != null && !ratings.isEmpty()) {
            originalRatings = new ArrayList<>(ratings);
            ratingsLiveData.setValue(ratings);
        } else {
            ratingsLiveData.setValue(new ArrayList<>());
        }
        calculateRatingDistribution(ratings);
        loadUserDetailsForRatings(ratings);
    });
}

```

```

private void extractColorsAndSizes(Product product) {
    List<Product.Variant> variants = product.getVariants();
    if (variants != null && !variants.isEmpty()) {
        List<ColorItem> colors = new ArrayList<>();
        for (int i = 0; i < variants.size(); i++) {
            Product.Variant variant = variants.get(i);
            colors.add(new ColorItem(String.valueOf(i), variant.getColor(), hexCode: "#000000"));
        }
        colorsLiveData.setValue(colors);
        if (selectedColor.getValue() == null && !colors.isEmpty()) {
            selectedColor.setValue(colors.get(0).getName());
        }
        updateSizesForSelectedColor(product);
    } else {
        setupDefaultColorsAndSizes(product.getProductType());
    }
}
💡 2 usages

```

```

private void updateSizesForSelectedColor(Product product) {
    String selectedColorName = selectedColor.getValue();
    List<Product.Variant> variants = product.getVariants();
    if (variants != null && !variants.isEmpty() && selectedColorName != null) {
        // Find the variant with the selected color
        for (Product.Variant variant : variants) {
            if (variant.getColor().equals(selectedColorName)) {
                List<Product.Variant.SizeQuantity> sizeQuantities = variant.getSizes();
                List<SizeItem> sizes = new ArrayList<>();
                if (sizeQuantities != null) {
                    for (int i = 0; i < sizeQuantities.size(); i++) {
                        Product.Variant.SizeQuantity sizeQty = sizeQuantities.get(i);
                        if (sizeQty.getQuantity() > 0) {
                            sizes.add(new SizeItem(String.valueOf(i), sizeQty.getSize()));
                        }
                    }
                }
                sizesLiveData.setValue(sizes);
                if (selectedSize.getValue() == null && !sizes.isEmpty()) {
                    selectedSize.setValue(sizes.get(0).getName());
                } else {
                    boolean sizeExists = false;
                    for (SizeItem size : sizes) {
                        if (size.getName().equals(selectedSize.getValue())) {
                            sizeExists = true;
                            break;
                        }
                    }
                    if (!sizeExists && !sizes.isEmpty()) {
                        selectedSize.setValue(sizes.get(0).getName());
                    }
                }
            }
        }
    }
}

```

```

private void setupDefaultColorsAndSizes(int productType) {
    List<ColorItem> defaultColors = new ArrayList<>();
    defaultColors.add(new ColorItem(id: "1", name: "Đen", hexCode: "#000000"));
    defaultColors.add(new ColorItem(id: "2", name: "Trắng", hexCode: "#FFFFFF"));
    defaultColors.add(new ColorItem(id: "3", name: "Be", hexCode: "#F5F5DC"));
    colorsLiveData.setValue(defaultColors);
    List<SizeItem> defaultSizes = new ArrayList<>();
    if (productType == 0) { // Áo/quần
        defaultSizes.add(new SizeItem(id: "1", name: "S"));
        defaultSizes.add(new SizeItem(id: "2", name: "M"));
        defaultSizes.add(new SizeItem(id: "3", name: "L"));
        defaultSizes.add(new SizeItem(id: "4", name: "XL"));
        defaultSizes.add(new SizeItem(id: "5", name: "2XL"));
    } else if (productType == 1) { // Giày
        defaultSizes.add(new SizeItem(id: "1", name: "38"));
        defaultSizes.add(new SizeItem(id: "2", name: "39"));
        defaultSizes.add(new SizeItem(id: "3", name: "40"));
        defaultSizes.add(new SizeItem(id: "4", name: "41"));
        defaultSizes.add(new SizeItem(id: "5", name: "42"));
    } else {
        defaultSizes.add(new SizeItem(id: "1", name: "S"));
        defaultSizes.add(new SizeItem(id: "2", name: "M"));
        defaultSizes.add(new SizeItem(id: "3", name: "L"));
    }
    sizesLiveData.setValue(defaultSizes);
    if (selectedColor.getValue() == null && !defaultColors.isEmpty()) {
        selectedColor.setValue(defaultColors.get(0).getName());
    }
    if (selectedSize.getValue() == null && !defaultSizes.isEmpty()) {
        selectedSize.setValue(defaultSizes.get(0).getName());
    }
}

```

```

public void loadSimilarProducts(String categoryId) {
    if (categoryId == null || categoryId.isEmpty()) {
        similarProductsLiveData.setValue(new ArrayList<>());
        return;
    }
    productRepository.getProductsByCategory(categoryId, limit: 10, products -> {
        Product currentProduct = productLiveData.getValue();
        List<Product> filteredProducts = new ArrayList<>();

        for (Product product : products) {
            if (currentProduct != null && !product.getProductId().equals(currentProduct.getProductId())) {
                filteredProducts.add(product);
            }
        }
        List<Product> limitedProducts = filteredProducts.size() > 10 ?
            filteredProducts.subList(0, 10) :
            filteredProducts;

        similarProductsLiveData.setValue(limitedProducts);
    });
}

```

```

private void loadUserDetailsForRatings(List<Rating> ratings) {
    if (ratings == null || ratings.isEmpty()) {
        userMapLiveData.setValue(new HashMap<>());
        return;
    }
    Set<String> uniqueUserIds = new HashSet<>();
    for (Rating rating : ratings) {
        uniqueUserIds.add(rating.getUid());
    }
    Map<String, User> userMap = new HashMap<>();
    AtomicInteger pendingRequests = new AtomicInteger(uniqueUserIds.size());
    for (String userId : uniqueUserIds) {
        userRepository.getUserById(userId, user -> {
            if (user != null) {
                userMap.put(userId, user);
            }
            if (pendingRequests.decrementAndGet() == 0) {
                // Tất cả requests đã hoàn thành
                userMapLiveData.setValue(userMap);
            }
        });
    }
}

```

```
    }
    private void calculateRatingDistribution(List<Rating> ratings) {
        Map<Integer, Integer> distribution = new HashMap<>();
        for (int i = 1; i <= 5; i++) {
            distribution.put(i, 0);
        }
        if (ratings != null) {
            for (Rating rating : ratings) {
                int rate = rating.getRate();
                distribution.put(rate, distribution.get(rate) + 1);
            }
            int totalRatings = ratings.size();
            if (totalRatings > 0) {
                Map<Integer, Integer> percentages = new HashMap<>();
                for (int i = 1; i <= 5; i++) {
                    int percentage = (int) (((float) distribution.get(i) / totalRatings) * 100);
                    percentages.put(i, percentage);
                }
                ratingDistributionLiveData.setValue(percentages);
            } else {
                Map<Integer, Integer> zeroPercentages = new HashMap<>();
                for (int i = 1; i <= 5; i++) {
                    zeroPercentages.put(i, 0);
                }
                ratingDistributionLiveData.setValue(zeroPercentages);
            }
        }
    }
}
```

```
public void setSelectedColor(String color) {
    selectedColor.setValue(color);
    Product product = productLiveData.getValue();
    if (product != null) {
        updateSizesForSelectedColor(product);
    }
}
1 usage
public void setSelectedSize(String size) {
    selectedSize.setValue(size);
}
public LiveData<Product> getProduct() {
    return productLiveData;
}
public LiveData<List<Rating>> getRatings() {
    return ratingsLiveData;
}
1 usage
public LiveData<List<Product>> getSimilarProducts() {
    return similarProductsLiveData;
}
1 usage
public LiveData<Map<String, User>> getUserMap() {
    return userMapLiveData;
}
2 usages
public LiveData<String> getSelectedColor() {
    return selectedColor;
}
```

```
2 usages
public LiveData<String> getSelectedColor() {
    return selectedColor;
}
2 usages
public LiveData<String> getSelectedSize() {
    return selectedSize;
}
1 usage
public LiveData<Map<Integer, Integer>> getRatingDistribution() {
    return ratingDistributionLiveData;
}
no usages
public LiveData<List<String>> getProductImages() {
    return productImagesLiveData;
}
1 usage
public LiveData<List<ColorItem>> getColors() {
    return colorsLiveData;
}
1 usage
public LiveData<List<SizeItem>> getSizes() {
    return sizesLiveData;
}
}
```

+ File ProductRepository.java

```

package com.project.clothingstore.service;

import ...

3 usages
public class ProductRepository {
    4 usages
    private final FirebaseFirestore db = FirebaseFirestore.getInstance();
    1 usage
    public void getProductById(String productId, Consumer<Product> callback) {
        DocumentReference productRef = db.collection( collectionPath: "products").document(productId);
        productRef.get().addOnSuccessListener(documentSnapshot -> {
            if (documentSnapshot.exists()) {
                Product product = documentSnapshot.toObject(Product.class);
                if (product != null) {
                    callback.accept(product);
                }
            }
        }).addOnFailureListener(e -> {
            e.printStackTrace();
            callback.accept( t: null);
        });
    }
}

```

```

public void getProductsByCategory(String categoryId, int limit, Consumer<List<Product>> callback) {
    if (categoryId == null || categoryId.isEmpty()) {
        callback.accept(new ArrayList<>()); // Return empty list if categoryId is invalid
        return;
    }

    db.collection( collectionPath: "products") CollectionReference
        .whereEqualTo( field: "categoryId", categoryId) Query
        .limit(10) // Lấy tối đa 10 sản phẩm
        .get() Task<QuerySnapshot>
        .addOnSuccessListener(queryDocumentSnapshots -> {
            List<Product> products = new ArrayList<>();
            for (DocumentSnapshot document : queryDocumentSnapshots) {
                Product product = document.toObject(Product.class);
                if (product != null) {
                    product.setProductId(document.getId());
                    products.add(product);
                }
            }
            callback.accept(products);
        })
        .addOnFailureListener(e -> {
            e.printStackTrace();
            callback.accept(new ArrayList<>()); // Return empty list on failure
        });
}

```

3.4.12. Chức năng hiển thị lịch sử mua hàng.

- File OrderFragment.java

```

package com.project.clothingstore.view.fragment;

import ...

public class OrderFragment extends Fragment {

    3 usages
    private OrderViewModel viewModel;
    3 usages
    private OrderAdapter adapter;
    5 usages
    private Button btnPending, btnCompleted, btnCanceled;

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_order, container, false);
    }
    @Override
    public void onViewCreated(@NonNull View view, Bundle savedInstanceState) {
        viewModel = new ViewModelProvider(owner: this).get(OrderViewModel.class);
        adapter = new OrderAdapter(requireContext(), new ArrayList<>());
        RecyclerView recyclerView = view.findViewById(R.id.rvOrders);
        recyclerView.setLayoutManager(new LinearLayoutManager(requireContext()));
        recyclerView.setAdapter(adapter);
        btnPending = view.findViewById(R.id.btnPending);
        btnCompleted = view.findViewById(R.id.btnCompleted);
        btnCanceled = view.findViewById(R.id.btnCanceled);
        viewModel.getOrdersLiveData().observe(getViewLifecycleOwner(), orders -> {
            if (orders != null) adapter.setOrders(orders);
        });
        filterOrders(status: "PENDING", btnPending);

        viewModel.getOrdersLiveData().observe(getViewLifecycleOwner(), orders -> {
            if (orders != null) adapter.setOrders(orders);
        });
        filterOrders(status: "PENDING", btnPending);
        btnPending.setOnClickListener(v -> filterOrders(status: "PENDING", btnPending));
        btnCompleted.setOnClickListener(v -> filterOrders(status: "SUCCESS", btnCompleted));
        btnCanceled.setOnClickListener(v -> filterOrders(status: "CANCEL", btnCanceled));
    }
    4 usages
    private void filterOrders(String status, Button selectedButton) {
        viewModel.loadOrdersByStatus(status);
        setSelectedButton(selectedButton);
    }
    1 usage
    private void setSelectedButton(Button selectedBtn) {
        btnPending.setSelected(false);
        btnCompleted.setSelected(false);
        btnCanceled.setSelected(false);
        selectedBtn.setSelected(true);
    }
}

```

- File OrderRepository.java

```

package com.project.clothingstore.service;

import ...

3 usages
public class OrderRepository {
    1 usage
    private final FirebaseFirestore db = FirebaseFirestore.getInstance();

    1 usage
    public void getOrdersByStatus(String userId, String status, MutableLiveData<List<Orders>> liveData) {
        db.collection( collectionPath: "orders" ) CollectionReference
            .whereEqualTo( field: "uid", userId ) Query
            .whereEqualTo( field: "status", status )
            .get() Task<QuerySnapshot>
            .addOnSuccessListener(queryDocumentSnapshots -> {
                List<Orders> ordersList = new ArrayList<>();
                for (QueryDocumentSnapshot doc : queryDocumentSnapshots) {
                    Orders order = doc.toObject(Orders.class);
                    order.setOrderId(doc.getId());
                    ordersList.add(order);
                }
                liveData.setValue(ordersList);
            })
            .addOnFailureListener(e -> liveData.setValue(null));
    }
}

```

- File OrderViewModel.java

```

package com.project.clothingstore.viewmodel;

import ...

3 usages
public class OrderViewModel extends AndroidViewModel {
    2 usages
    private final OrderRepository repository;
    2 usages
    private final MutableLiveData<List<Orders>> ordersLiveData = new MutableLiveData<>();

    no usages
    public OrderViewModel(@NonNull Application application) {
        super(application);
        repository = new OrderRepository();
    }

    1 usage
    public LiveData<List<Orders>> getOrdersLiveData() { return ordersLiveData; }

    1 usage
    public void loadOrdersByStatus(String status) {
        FirebaseUser currentUser = FirebaseAuth.getInstance().getCurrentUser();
        if (currentUser != null) {
            repository.getOrdersByStatus(currentUser.getUid(), status, ordersLiveData);
        }
    }
}

```

3.4.13. Chức năng hiện chi tiết đơn hàng

- File OrderDetailActivity.java

```
package com.project.clothingstore.view.activity;

import ...

public class OrderDetailActivity extends AppCompatActivity {
    2 usages
    private TextView tvOrderId, tvOrderAdress, tvTotalPrice, tvShippingFee, tvSubTotalPrice;
    2 usages
    private ImageView btnBack;
    4 usages
    private TextView tvOrderStatus, tvOrderSubStatus;
    4 usages
    private ImageView imvOrderStatus;
    6 usages
    private Button btnShopping, btnRating;
    3 usages
    private RecyclerView rvOrderItems;
    2 usages
    private OrderItemAdapter itemAdapter;
    13 usages
    private Orders order;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_order_detail);
        tvOrderId = findViewById(R.id.tvOrderId_Detail);
        tvOrderAdress = findViewById(R.id.tvOrderAdress_Detail);
        tvSubTotalPrice = findViewById(R.id.tvSubTotalPrice_Detail);
        tvShippingFee = findViewById(R.id.tvShippingFee_Detail);
        tvTotalPrice = findViewById(R.id.tvTotalPrice_Detail);
        rvOrderItems = findViewById(R.id.rvOrderProducts);

        tvOrderStatus = findViewById(R.id.tvOrderStatus);
        imvOrderStatus = findViewById(R.id.imvOrderStatus);
        tvOrderSubStatus = findViewById(R.id.tvOrderSubStatus);
        order = getIntent().getParcelableExtra("name: "order");
        btnBack = findViewById(R.id.btnBack);
        btnBack.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                finish();
            }
        });
        btnShopping = findViewById(R.id.btnHomePage);
        btnShopping.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String uid = FirebaseAuth.getInstance().getUid();
                getUnratedProducts(order.getOrderItems(), uid, unratedItems -> {
                    if (unratedItems.isEmpty()) {
                        Toast.makeText(context: OrderDetailActivity.this,
                            text: "Bạn đã đánh giá tất cả sản phẩm trong đơn hàng này", Toast.LENGTH_SHORT).show();
                    } else {
                        Orders unratedOrder = new Orders(order);
                        unratedOrder.setOrderItems(unratedItems);
                        Intent intent = new Intent(packageContext: OrderDetailActivity.this, RatingActivity.class);
                        intent.putExtra(name: "orderList", unratedOrder);
                        startActivity(intent);
                    }
                });
            }
        });
    }
}
```

```

    });
    btnRating = findViewById(R.id.btnRating);
    btnRating.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(packageContext, MainActivity.class);
            startActivity(intent);
        }
    });
    if (order != null) {
        if (order.getStatus().equals("PENDING")) {
            tvOrderStatus.setText("Đơn hàng của bạn đang \nđang giao");
            imvOrderStatus.setImageResource(R.drawable.dmq_ic_delivery_truck);
            tvOrderSubStatus.setVisibility(View.GONE);
            btnShopping.setVisibility(View.GONE);
            btnRating.setText("Tiếp tục mua sắm");
            btnRating.setVisibility(View.VISIBLE);
        } else if (order.getStatus().equals("CANCEL")) {
            tvOrderStatus.setText("Đơn hàng của bạn đã bị hủy");
            imvOrderStatus.setImageResource(R.drawable.ic_order_cancelled);
            tvOrderSubStatus.setVisibility(View.GONE);
            btnShopping.setVisibility(View.GONE);
            btnRating.setText("Tiếp tục mua sắm");
            btnRating.setVisibility(View.VISIBLE);
        } else if (order.getStatus().equals("SUCCESS")) {
            tvOrderStatus.setText("Đơn hàng của bạn đã \nđang giao thành công");
            imvOrderStatus.setImageResource(R.drawable.ic_hand_box);
            tvOrderSubStatus.setVisibility(View.VISIBLE);
            btnShopping.setVisibility(View.VISIBLE);
            btnShopping.setText("Đánh giá");
            btnRating.setText("Mua sắm");
            btnRating.setVisibility(View.VISIBLE);
        }
        tvOrderId.setText("#" + order.getOrderId());
        tvOrderAddress.setText("đ" + order.getAddress());

        NumberFormat formatter = NumberFormat.getInstance(new Locale(language: "vi", country: "VN"));

        tvSubTotalPrice.setText("đ " + formatter.format(order.calculateSubPrice()));
        tvShippingFee.setText("đ " + formatter.format(order.getShippingPrice()));
        tvTotalPrice.setText("đ " + formatter.format(order.calculateTotalPrice()));
        itemAdapter = new OrderItemAdapter(context: this, order.getOrderItems());
        rvOrderItems.setLayoutManager(new LinearLayoutManager(context: this));
        rvOrderItems.setAdapter(itemAdapter);
    }
}
1 usage
private void getUnratedProducts(List<OrderItems> orderItems, String uid, OnUnratedProductsListener listener) {
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    final int[] checkedProducts = {0};
    List<OrderItems> unratedItems = new ArrayList<>();
}

```

```

private void getUnratedProducts(List<OrderItems> orderItems, String uid, OnUnratedProductsListener listener) {
    FirebaseFirestore db = FirebaseFirestore.getInstance();
    final int[] checkedProducts = {0};
    List<OrderItems> unratedItems = new ArrayList<>();

    for (OrderItems item : orderItems) {
        String productId = item.getProductId();

        db.collection(collectionPath: "ratings") CollectionReference
            .whereEqualTo(field: "uid", uid) Query
            .whereEqualTo(field: "productId", productId)
            .get() Task<QuerySnapshot>
            .addOnCompleteListener(task -> {
                checkedProducts[0]++;
                if (task.isSuccessful() && task.getResult().isEmpty()) {
                    unratedItems.add(item);
                }
                if (checkedProducts[0] == orderItems.size()) {
                    listener.onUnratedProductsFound(unratedItems);
                }
            });
    }
}

interface OnUnratedProductsListener {
    void onUnratedProductsFound(List<OrderItems> unratedItems);
}

```

- File

OrderRepository.java

```

package com.project.clothingstore.service;

import ...

3 usages
public class OrderRepository {
    1 usage
    private final FirebaseFirestore db = FirebaseFirestore.getInstance();

    1 usage
    public void getOrdersByStatus(String userId, String status, MutableLiveData<List<Orders>> liveData) {
        db.collection(collectionPath: "orders") CollectionReference
            .whereEqualTo(field: "uid", userId) Query
            .whereEqualTo(field: "status", status)
            .get() Task<QuerySnapshot>
            .addOnSuccessListener(queryDocumentSnapshots -> {
                List<Orders> ordersList = new ArrayList<>();
                for (QueryDocumentSnapshot doc : queryDocumentSnapshots) {
                    Orders order = doc.toObject(Orders.class);
                    order.setOrderId(doc.getId());
                    ordersList.add(order);
                }
                liveData.setValue(ordersList);
            })
            .addOnFailureListener(e -> liveData.setValue(null));
    }
}

```

3.4.14 Chức năng đánh giá sản phẩm

- File RatingActivity.java

```
package com.project.clothingstore.view.activity;

import ...

public class RatingActivity extends AppCompatActivity {
    3 usages
    private ImageView star1;
    3 usages
    private ImageView star2;
    3 usages
    private ImageView star3;
    3 usages
    private ImageView star4;
    3 usages
    private ImageView star5;
    3 usages
    private EditText etComment;
    2 usages
    private TextView tvCharCount;
    2 usages
    private Button btnSubmitRating;
    4 usages
    private Orders order;
    4 usages
    private int currentRating = 4;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_rating);
        star1 = findViewById(R.id.star1);
        star2 = findViewById(R.id.star2);
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_rating);
    star1 = findViewById(R.id.star1);
    star2 = findViewById(R.id.star2);
    star3 = findViewById(R.id.star3);
    star4 = findViewById(R.id.star4);
    star5 = findViewById(R.id.star5);
    etComment = findViewById(R.id.etComment);
    tvCharCount = findViewById(R.id.tvCharCount);
    btnSubmitRating = findViewById(R.id.btnSubmitRating);
    setupStarRating();
    setupCharacterCounter();
    findViewById(R.id.btnExit).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { finish(); }
    });
    Log.d("RatingActivity", "Order received: ");

    order = getIntent().getParcelableExtra("name: "orderList");
    if (order == null) {
        Log.d("RatingActivity", "Không tìm thấy thông tin đơn hàng");
    }
    Log.d("RatingActivity", "Order received: " + order.getOrderItems().get(0).getPrice());
    btnSubmitRating.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { submitRating(); }
    });
}
```

```
private void setupStarRating() {
    final List<ImageView> stars = Arrays.asList(star1, star2, star3, star4, star5);
    updateStarDisplay(currentRating);
    for (int i = 0; i < stars.size(); i++) {
        final int position = i;
        stars.get(i).setOnTouchListener(new View.OnTouchListener() {
            @Override
            public void onClick(View v) {
                currentRating = position + 1;
                updateStarDisplay(currentRating);
            }
        });
    }
}
```

2 usages

```
private void updateStarDisplay(int rating) {
    List<ImageView> stars = Arrays.asList(star1, star2, star3, star4, star5);

    for (int i = 0; i < stars.size(); i++) {
        if (i < rating) {
            stars.get(i).setImageResource(R.drawable.ic_star_filled);
            stars.get(i).setImageTintList(ColorStateList.valueOf(
                ContextCompat.getColor(context, R.color.green)));
        } else {
            stars.get(i).setImageResource(R.drawable.ic_star_empty);
            stars.get(i).setImageTintList(ColorStateList.valueOf(
                ContextCompat.getColor(context, R.color.gray_light)));
        }
    }
}
```

```

private void setupCharacterCounter() {
    etComment.addTextChangedListener(new TextWatcher() {
        @Override
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {
        }

        @Override
        public void onTextChanged(CharSequence s, int start, int before, int count) {
        }
        @Override
        public void afterTextChanged(Editable s) {
            int length = s.length();
            tvCharCount.setText(length + " kí tự");
        }
    });
}

1 usage
private void submitRating() {
    String comment = etComment.getText().toString().trim();
    if (comment.isEmpty()) {
        Toast.makeText(context: this, text: "Vui lòng nhập cảm nghĩ của bạn", Toast.LENGTH_SHORT).show();
        return;
    }
    List<OrderItems> orderItems = order.getOrderItems();

    if (orderItems == null || orderItems.isEmpty()) {
        Toast.makeText(context: this, text: "Không tìm thấy thông tin sản phẩm để đánh giá", Toast.LENGTH_SHORT).show();
        return;
    }
    final int[] completedRatings = {0};

    }

    final int[] completedRatings = {0};
    final boolean[] hasError = {false};
    String uid = FirebaseAuth.getInstance().getUid();
    for (OrderItems item : orderItems) {
        String productId = item.getProductId();
        Rating rating = new Rating(uid, productId, currentRating, comment);
        FirebaseFirestore db = FirebaseFirestore.getInstance();
        db.collection(collectionPath: "ratings") CollectionReference
            .add(rating) Task<DocumentReference>
            .addOnSuccessListener(documentReference -> {
                String ratingId = documentReference.getId();
                documentReference.update(field: "ratingId", ratingId);
                completedRatings[0]++;
                if (completedRatings[0] == orderItems.size() && !hasError[0]) {
                    Toast.makeText(context: RatingActivity.this, text: "Cảm ơn bạn đã đánh giá sản phẩm!", Toast.LENGTH_SHORT).show();
                    finish();
                }
            })
            .addOnFailureListener(e -> {
                hasError[0] = true;
                Toast.makeText(context: RatingActivity.this, text: "Lỗi khi gửi đánh giá: " + e.getMessage(), Toast.LENGTH_SHORT).show();
            });
    }
    Intent intent = new Intent(packageContext: RatingActivity.this, MainActivity.class);
    startActivity(intent);
}
}

```

- File RatingRepository.java

```
package com.project.clothingstore.service;

import ...

3 usages
public class RatingRepository {
    1 usage
    private FirebaseFirestore firestore = FirebaseFirestore.getInstance();

    1 usage
    public void getRatingsByProductId(String productId, Consumer<List<Rating>> callback) {
        firestore.collection( collectionPath: "ratings" ) CollectionReference
            .whereEqualTo( field: "productId", productId ) Query
            .get() Task<QuerySnapshot>
            .addOnCompleteListener(task -> [
                if (task.isSuccessful() && task.getResult() != null) {
                    List<Rating> ratings = new ArrayList<>();
                    for (DocumentSnapshot doc : task.getResult()) {
                        Rating rating = doc.toObject(Rating.class);
                        rating.setRatingId(doc.getId());
                        ratings.add(rating);
                    }
                    Log.d( tag: "RatingRepository", msg: "Loaded " + ratings.size() + " ratings");
                    callback.accept(ratings);
                } else {
                    Log.e( tag: "RatingRepository", msg: "Error loading ratings", task.getException());
                    callback.accept(new ArrayList<>());
                }
            ]);
    }
}
```

KẾT LUẬN

1. Kết quả đạt được

Sau quá trình nghiên cứu và triển khai, nhóm đã xây dựng thành công phần mềm bán quần áo hoạt động trên cả nền tảng web và thiết bị di động, với các KQ cụ thể như sau:

- **Xây dựng thành công ứng dụng di động Android** viết bằng ngôn ngữ Java và Kotlin theo kiến trúc MVVM. Việc áp dụng kiến trúc MVVM giúp tách biệt rõ ràng giữa giao diện người dùng và logic xử lý, từ đó nâng cao khả năng bảo trì, mở rộng và tái sử dụng mã nguồn.
- **Phát triển website bán hàng bằng Next.js**, tận dụng ưu điểm về tốc độ, tối ưu trải nghiệm người dùng. Website có giao diện hiện đại, tương thích với nhiều kích thước màn hình (responsive).
- **Tích hợp cơ sở dữ liệu thời gian thực Firebase** cho cả ứng dụng web và mobile, giúp đồng bộ hóa dữ liệu nhanh chóng và hiệu quả. Các chức năng quản lý người dùng, sản phẩm, đơn hàng, mã giảm giá và đánh giá sản phẩm được triển khai đầy đủ.
- **Ứng dụng và website hỗ trợ đầy đủ các chức năng thương mại điện tử cơ bản**, bao gồm:
 - Đăng ký/đăng nhập tài khoản người dùng.
 - Tìm kiếm, duyệt sản phẩm theo danh mục.
 - Chọn biến thể sản phẩm (kích cỡ, màu sắc).
 - Thêm vào giỏ hàng, tạo đơn hàng/Quản lý lịch sử mua hàng.
 - Áp dụng mã khuyến mãi, tính phí vận chuyển và đánh giá sản phẩm sau khi nhận hàng.
- **Giao diện người dùng thân thiện, hiện đại**, được tối ưu trải nghiệm trên cả thiết bị di động và trình duyệt web, đảm bảo trải nghiệm nhất quán và dễ sử dụng.

2. Nhược điểm

Mặc dù dự án đã hoàn thiện được các chức năng chính và mang lại trải nghiệm người dùng tốt, tuy nhiên vẫn tồn tại một số nhược điểm và hạn chế như sau:

- **Chưa tích hợp cổng thanh toán trực tuyến**: Việc thanh toán đơn hàng mới chỉ dừng lại ở mức đơn giản (COD - thanh toán khi nhận hàng), chưa hỗ trợ các phương thức thanh toán hiện đại như ví điện tử, thẻ ngân hàng hoặc QR code.
- **Khả năng bảo mật còn cơ bản**: Dữ liệu người dùng được lưu trữ trên Firebase

nhưng chưa có cơ chế mã hóa hoặc kiểm soát truy cập nâng cao, tiềm ẩn rủi ro nếu hệ thống bị tấn công.

- **Hiệu năng ứng dụng chưa được tối ưu khi có nhiều dữ liệu:** Khi số lượng sản phẩm và đơn hàng lớn, thời gian tải dữ liệu từ Firebase có thể bị chậm và ảnh hưởng đến trải nghiệm người dùng.
- **Thiếu các tính năng nâng cao về trải nghiệm người dùng:** Chưa có hệ thống để xuất sản phẩm thông minh, chưa hỗ trợ đa ngôn ngữ, và chưa có hệ thống thông báo (notification) theo thời gian thực.
- **Phản địa chỉ đặt hàng chưa định vị chính xác:** Ứng dụng chưa sử dụng bản đồ hoặc GPS để xác định vị trí thực của người dùng, dẫn đến việc nhập địa chỉ thủ công và chưa thể xác định khoảng cách chính xác để tính phí vận chuyển hợp lý.
- **Chưa có hệ thống phân quyền linh hoạt và nâng cao cho nhân viên,** ví dụ như chủ cửa hàng có thể cấp hoặc thu hồi các quyền cho nhân viên bán hàng để họ có thể thực hiện một số chức năng nhất định ngoài chức năng mặc định của mình.
- **Chưa có thống kê lợi nhuận:** Hiện tại, hệ thống mới chỉ cung cấp các báo cáo về doanh thu mà chưa có các báo cáo chi tiết về lợi nhuận, điều này khiến cho việc theo dõi và đánh giá hiệu quả kinh doanh chưa đầy đủ.

3. Hướng phát triển

Dự án bán quần áo Fluxstore hiện tại đã hoàn thiện các chức năng cơ bản, tuy nhiên để nâng cao tính cạnh tranh và cải thiện trải nghiệm người dùng, có thể triển khai các hướng phát triển sau đây:

1. Tích hợp tính năng phân quyền nâng cao:

Hệ thống phân quyền cho nhân viên bán hàng cần được mở rộng hơn nữa, cho phép chủ cửa hàng có thể linh hoạt **giao quyền** và **quản lý quyền hạn** của từng nhân viên bán hàng. Các quyền có thể được phân chia chi tiết hơn, như quyền **thêm/xoá sản phẩm**, quyền **thay đổi giá bán**, hoặc quyền **quản lý các đơn hàng** cụ thể. Điều này giúp tối ưu công việc trong cửa hàng và kiểm soát tốt hơn các chức năng quản lý.

2. Tích hợp cổng thanh toán trực tuyến:

Để đáp ứng nhu cầu mua sắm trực tuyến của khách hàng, việc tích hợp các **cổng thanh toán trực tuyến** (Momo, ZaloPay, Thẻ ngân hàng, etc.) là một bước đi quan trọng. Điều này giúp người dùng thanh toán nhanh chóng, dễ dàng, đồng thời **giảm thiểu rủi ro liên quan đến thanh toán tiền mặt khi giao hàng**.

3. Cải tiến phản địa chỉ giao hàng:

Việc sử dụng **GPS hoặc hệ thống bản đồ** để tự động định vị và xác định **địa chỉ giao hàng chính xác** giúp giảm thiểu sai sót trong việc nhập địa chỉ và tính toán phí vận chuyển. Hệ thống này có thể **tính toán khoảng cách tự động** và tối ưu hóa phí giao hàng cho khách hàng và cửa hàng.

- 4. Thông kê lợi nhuận và báo cáo chi tiết hơn:**
Cần phải bổ sung các **báo cáo thống kê về lợi nhuận**, bao gồm **chi phí nhập hàng, chi phí vận chuyển**, và các khoản chi khác. Điều này sẽ giúp chủ cửa hàng có cái nhìn toàn diện về tình hình tài chính và đưa ra các quyết định kinh doanh hợp lý hơn.
- 5. Cải thiện hiệu năng và khả năng mở rộng:**
Khi dữ liệu sản phẩm và đơn hàng gia tăng, cần cải thiện hiệu năng của hệ thống, đặc biệt là trong việc **truy vấn và hiển thị dữ liệu**. Có thể sử dụng các phương pháp như **phân trang, bộ nhớ đệm (caching)**, và **tối ưu hóa truy vấn** để tăng tốc độ hoạt động của ứng dụng.
- 6. Thêm tính năng đề xuất sản phẩm:**
Cải thiện trải nghiệm mua sắm bằng cách triển khai hệ thống **đề xuất sản phẩm** thông minh dựa trên hành vi người dùng, lịch sử mua hàng, và sở thích cá nhân. Điều này sẽ giúp **tăng tỷ lệ chuyển đổi** và **nâng cao trải nghiệm người dùng**.
- 7. Mở rộng hỗ trợ đa ngôn ngữ:**
Để phục vụ nhiều đối tượng khách hàng hơn, có thể triển khai **hệ thống đa ngôn ngữ** giúp người dùng dễ dàng lựa chọn ngôn ngữ giao diện phù hợp. Điều này sẽ giúp mở rộng phạm vi thị trường và phục vụ khách hàng quốc tế hiệu quả hơn.
- 8. Thông báo và thông báo theo thời gian thực:**
Cần triển khai hệ thống **Thông báo tự động (push notifications)** để người dùng nhận thông báo về các chương trình khuyến mãi, trạng thái đơn hàng, hoặc các sự kiện đặc biệt. Việc thông báo kịp thời giúp giữ chân khách hàng và tạo sự gắn kết lâu dài.
- 9. Tối ưu hóa và mở rộng hệ thống quản lý kho hàng:**
Hệ thống quản lý kho cần được nâng cấp để theo dõi **tồn kho chính xác** và **tự động cảnh báo khi sản phẩm sắp hết hàng**. Điều này sẽ giúp cửa hàng duy trì lượng hàng hóa phù hợp và tránh tình trạng thiếu hàng hoặc quá tải.
- 10. Phát triển ứng dụng trên các nền tảng khác:**
Mở rộng dự án ra các nền tảng như **iOS**, để mở rộng đối tượng người dùng và tăng tính linh hoạt của ứng dụng. Điều này giúp tiếp cận nhiều khách hàng hơn và tạo cơ hội phát triển thị trường rộng lớn hơn.

TÀI LIỆU THAM KHẢO

- [1] David Flanagan, JavaScript: The Definitive Guide, 7th Edition, O'Reilly Media, 2020.
- [2] Adam Freeman, “Pro jQuery”, Apress, 2018.
- [3] Benjamin Jakobus, “Mastering Bootstrap 5”, Packt Publishing, 2018.