

# BÁO CÁO ĐỒ ÁN CUỐI KỲ

## MÔN PHÁT TRIỂN ỨNG DỤNG MOBILE

\*\*\*



## XÂY DỰNG ỨNG DỤNG MUA HÀNG QUẦN ÁO

### Nhóm thực hiện:

1. Nguyễn Duy Nguyễn
2. Lò Sìn Dậu
3. Nguyễn Ngọc Bảo
4. Nguyễn Hoàn Thiện
5. Trần Văn Thịnh

**GVHD:** Ts. Đặng Ngọc Hoàng Thành

<b>CHƯƠNG 1: TỔNG QUAN GIỚI THIỆU ĐỀ TÀI</b>	<b>2</b>
1. Thương mại điện tử	2
2. Tính cấp thiết	2
<b>CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG</b>	<b>3</b>
1. Sơ đồ công việc	3
2. Sơ đồ phân rã chức năng	3
2.1 Phân rã Use Case Chinh sửa thông tin user:	3
2.2 Phân rã Use Case Đặt hàng:	4
2.3 Phân rã Use Case Tìm kiếm sản phẩm:	4
<b>CHƯƠNG 3: THIẾT KẾ GIAO DIỆN</b>	<b>5</b>
1. Giới thiệu tổng quan về giao diện	5
2. Các quy luật thiết kế giao diện người dùng	6
<b>CHƯƠNG 4: XÂY DỰNG CƠ SỞ DỮ LIỆU</b>	<b>9</b>
1. Giới thiệu Firebase	9
2. Cơ sở dữ liệu phi quan hệ	10
<b>CHƯƠNG 5: XÂY DỰNG ỨNG DỤNG</b>	<b>11</b>
1. Giới thiệu mô hình MVVM (Model-View-ViewModel)	11
2. Cấu trúc ứng dụng	12
3. Thiết kế Firebase Repository	13
4. Các chức năng chính trong ứng dụng	15
4.1. Đăng ký tài khoản	15
4.2. Đăng nhập	16
4.3. Chinh sửa thông tin user	17
4.4. Thêm vào giỏ hàng	17
4.5. Thanh toán	18
4.6. Tìm kiếm sản phẩm	19
<b>CHƯƠNG 6: QUY TRÌNH THỰC THI</b>	<b>21</b>
1. Quy trình đăng ký tài khoản	21
2. Quy trình đăng nhập tài khoản	21
3. Quy trình xem và thêm sản phẩm vào giỏ hàng	22
4. Quy trình thanh toán	22
<b>KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN</b>	<b>23</b>
1. Kết luận	23
2. Hướng phát triển	23
<b>TÀI LIỆU THAM KHẢO</b>	<b>24</b>
<b>PHỤ LỤC</b>	<b>25</b>
Link github: <a href="https://github.com/Dua24/AppMobileClothes">https://github.com/Dua24/AppMobileClothes</a>	25
Phân công	25

# CHƯƠNG 1: TỔNG QUAN GIỚI THIỆU ĐỀ TÀI

## 1. Thương mại điện tử

Thương mại điện tử là một xu hướng phát triển rất nhanh trong thời đại kỹ thuật số hiện nay. Với sự gia tăng của số lượng người dùng smartphone và tablet, việc mua sắm và giao dịch trực tuyến thông qua các ứng dụng di động đã trở nên phổ biến hơn bao giờ hết. Nhờ tính tiện dụng và tốc độ, ứng dụng thương mại điện tử cho phép khách hàng dễ dàng duyệt và mua sản phẩm mà mình mong muốn mọi lúc mọi nơi. Đồng thời, các doanh nghiệp có thể tận dụng tiềm năng của ứng dụng thương mại điện tử để tăng doanh thu, mở rộng thị trường và nâng cao trải nghiệm của khách hàng.

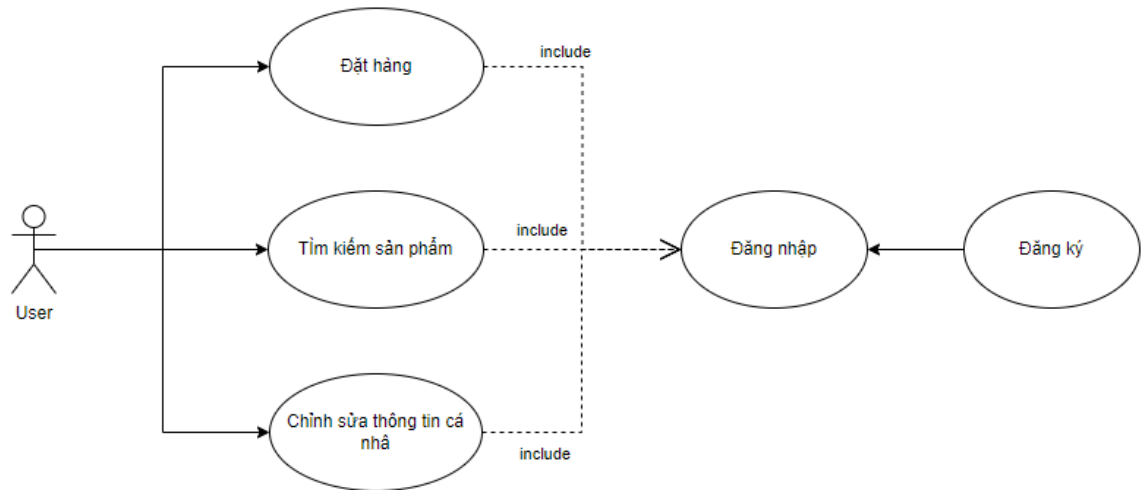
## 2. Tính cấp thiết

Việc phát triển ứng dụng thương mại điện tử giúp các doanh nghiệp có thể tiếp cận khách hàng một cách dễ dàng và hiệu quả hơn. Bên cạnh đó, các doanh nghiệp cũng có thể tận dụng tiềm năng của ứng dụng thương mại điện tử để tăng doanh thu, tăng khả năng cạnh tranh và nâng cao trải nghiệm của khách hàng. Do đó, thương mại điện tử qua app đang trở thành một yếu tố cấp thiết và không thể thiếu cho các doanh nghiệp trong thời đại hiện nay. Vì thế ứng dụng “Mua quần áo online” ra đời.

## CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

### 1. Sơ đồ công việc

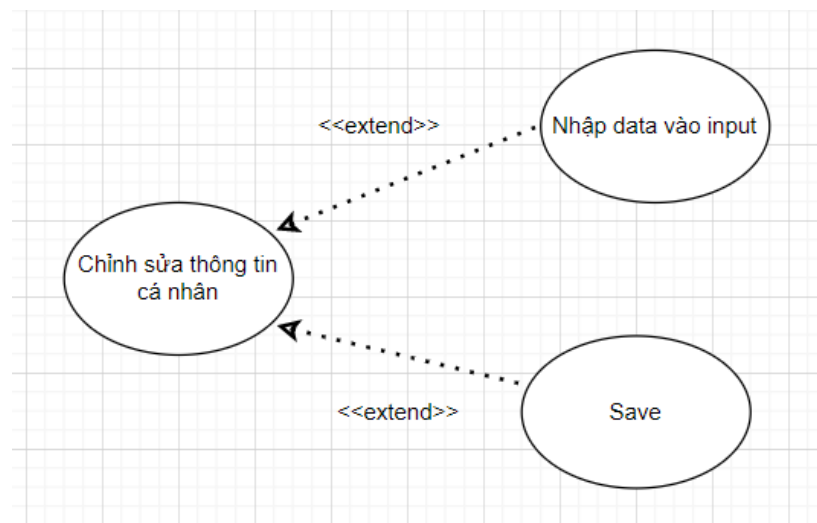
Ứng dụng khi ở vị trí là user, bao gồm các chức năng: Đặt hàng, Tìm kiếm sản phẩm, Chỉnh sửa thông tin cá nhân.



*Hình 2.1: Sơ đồ Use case tổng quát*

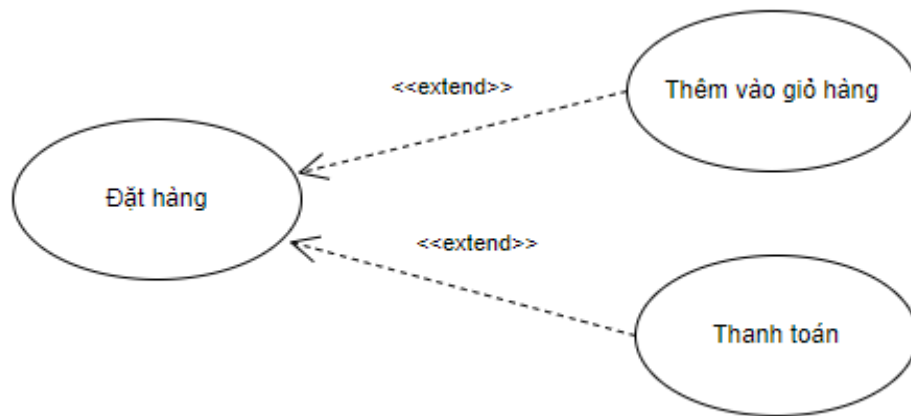
### 2. Sơ đồ phân rã chức năng

#### 2.1 Phân rã Use Case Chỉnh sửa thông tin user:



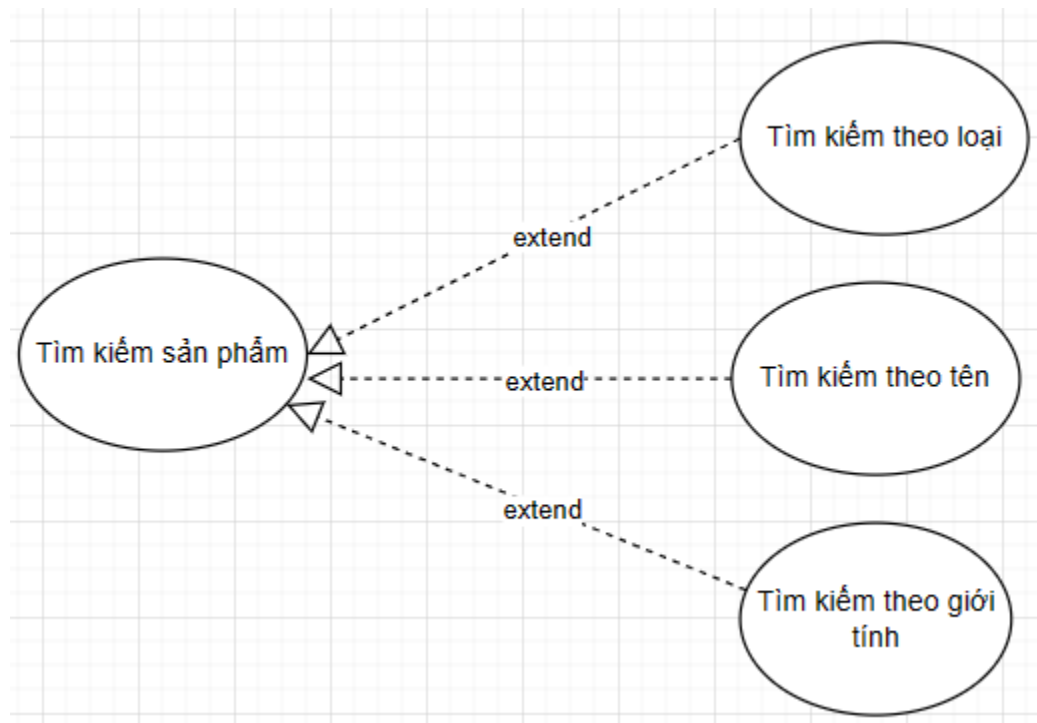
*Hình 2.2: Phân rã use case chỉnh sửa thông tin user*

## 2.2 Phân rã Use Case Đặt hàng:



*Hình 2.3: Phân rã use case đặt hàng*

## 2.3 Phân rã Use Case Tìm kiếm sản phẩm:



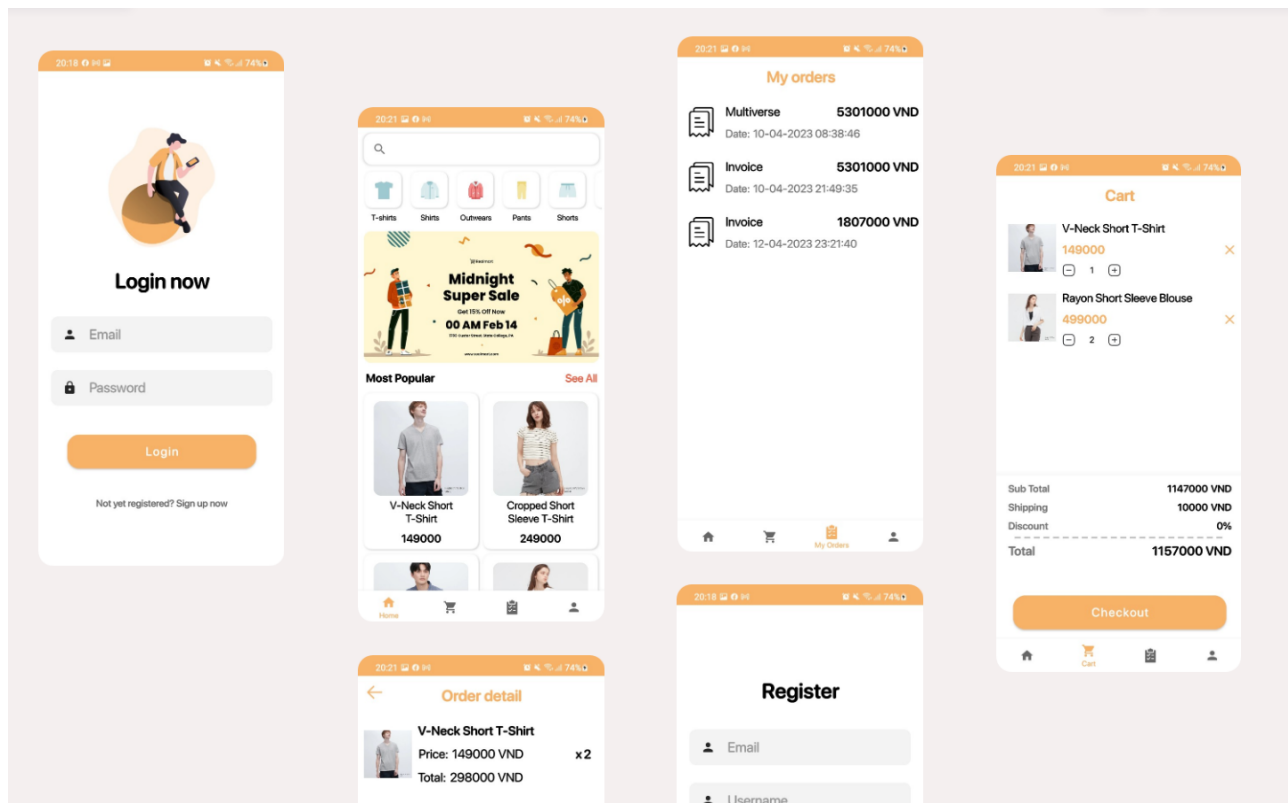
*Hình 2.4: Phân rã use case tìm kiếm sản phẩm*

## CHƯƠNG 3: THIẾT KẾ GIAO DIỆN

### 1. Giới thiệu tổng quan về giao diện

Giao diện và trải nghiệm người dùng (UI/UX) là yếu tố quan trọng khi ta thiết kế một ứng dụng thương mại điện tử (e-commerce application). Mục đích chính của UI/UX là để thiết kế một giao diện thân thiện với người dùng, thúc đẩy trải nghiệm mua hàng, và cải thiện mức độ thoải mái trong quá trình sử dụng ứng dụng.

Để đạt được mục đích đó thì chúng ta nên thiết kế dựa theo những quy định thiết yếu như là giao diện trực quan, thiết kế dễ thao tác, nội dung chữ thống nhất và rõ ràng, dễ dàng chuyển trang và điều hướng.

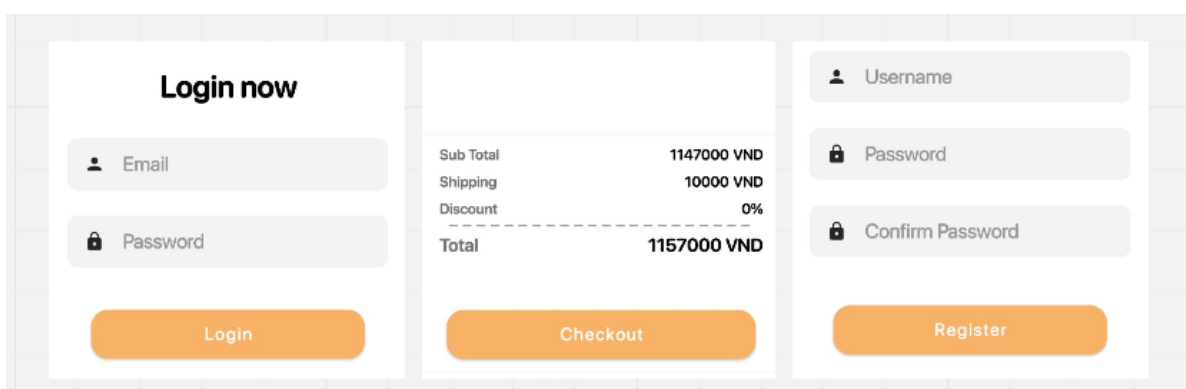


*Hình 3.1: Tổng quan giao diện của ứng dụng*

## 2. Các quy luật thiết kế giao diện người dùng

Cụ thể dưới đây sẽ là những quy luật mà đã được áp dụng để thiết kế ứng dụng trong đề tài này:

**Nút bấm** (Buttons): nên được thiết kế độ lớn sao cho người dùng dễ thao tác cũng như là những chữ viết trên nút bấm phải rõ ràng và thể hiện đủ ý nghĩa để người dùng hiểu nút bấm đó dẫn đến đâu hoặc làm gì. Đặc biệt là những nút bấm quan trọng như nút thêm vào giỏ hàng hoặc nút bấm thanh toán phải có màu sắc tương phản với nền và đặc biệt để kích thích thị giác.



*Hình 3.2: Áp dụng quy luật thiết kế nút bấm*

**Chữ viết** (Text): phải có độ lớn vừa đủ, nội dung rõ ràng và không quá dài.



*Hình 3.3: Áp dụng quy luật thiết kế chữ viết*

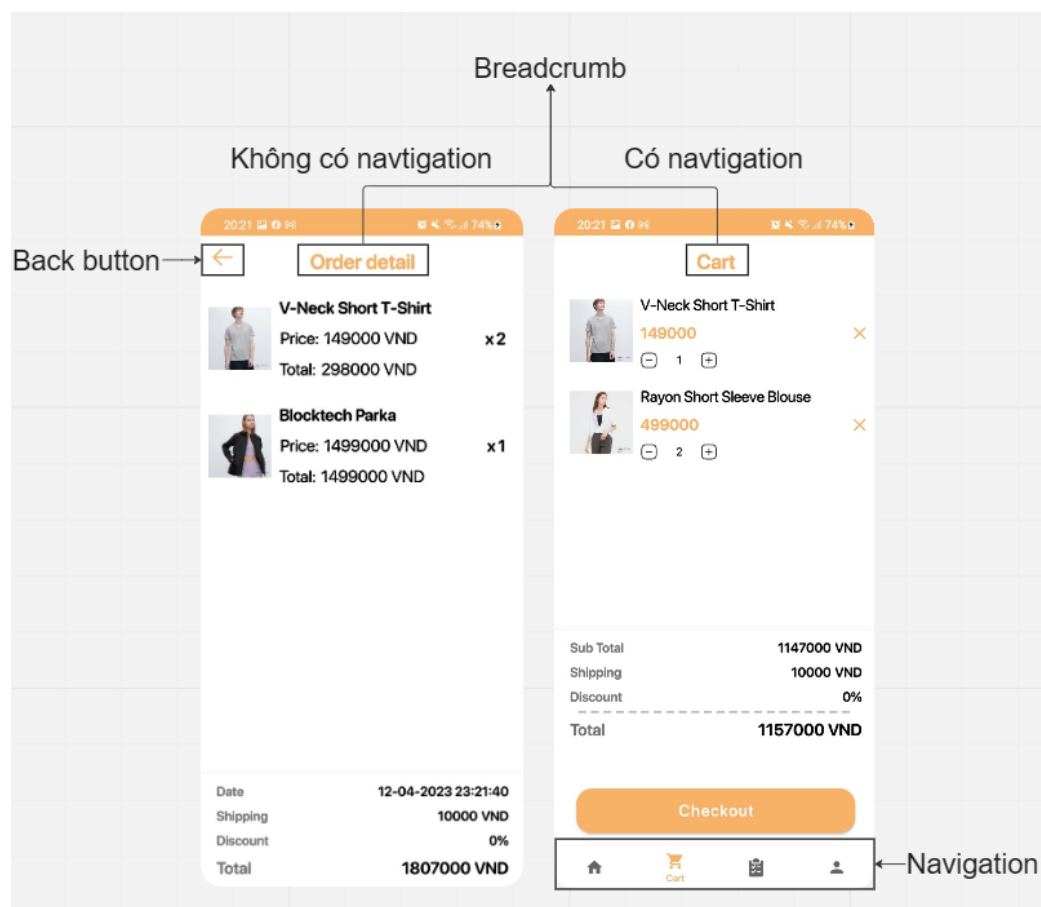
**Bố cục** (Layout): là khái niệm bao quát chứa toàn bộ các thành phần của một giao diện. Vì vậy bố cục rõ ràng là một bố cục được thiết kế với các thành phần được sắp xếp theo thứ tự hợp lý, tận dụng được khoảng trống giữa các thành phần. Hơn thế nữa những nội dung và thành phần quan trọng phải được đặt sao cho nổi bật nhất, dễ đập vào mắt người dùng.



*Hình 3.4: Áp dụng quy luật thiết kế bố cục*



**Điều hướng** (Navigation): là một thành phần không thể thiếu của một ứng dụng trên thiết bị di động. Khác với các ứng dụng trên các thiết bị lớn, ứng dụng trên thiết bị di động thường mang lại cảm giác khó sử dụng do không gian thao tác hạn chế, vì vậy điều hướng sẽ là một thứ giúp loại bỏ đi cảm giác khó chịu đó nếu được thiết kế đúng cách. Thậm chí những thanh điều hướng sẽ giúp trải nghiệm ứng dụng thêm ấn tượng nếu như những biểu tượng (Icon) dễ nhận biết, luồng điều hướng rõ ràng liền mạch không gây rối, ở mọi trang đều có khả năng giúp người dùng điều hướng và dòng chữ để biết người dùng đang ở trang nào (breadcrumb).



*Hình 3.4: Áp dụng quy luật thiết kế điều hướng*

## CHƯƠNG 4: XÂY DỰNG CƠ SỞ DỮ LIỆU

### 1. Giới thiệu Firebase

Firebase là một dịch vụ Backend-as-a-Service (BaaS - là một loại hình dịch vụ giúp tự động hóa việc phát triển phía server-side bao gồm cung cấp cơ sở hạ tầng điện toán đám mây). Firebase cung cấp nhiều loại công cụ và dịch vụ dựa trên cơ sở hạ tầng của Google. Có hai dịch vụ chính được sử dụng trong phạm vi đề tài này là Firebase Realtime Database và Firebase Cloud Storage:

- + Realtime Database: là hệ thống lưu trữ database dùng điện toán đám mây, mà trong đó phần dữ liệu sẽ được lưu theo dạng JSON và được đồng bộ theo thời gian thực (synchronized in realtime) đối với mọi kết nối của người dùng cuối.



**Hình 4.1:** Lưu trữ dữ liệu trên Realtime Database

- + Cloud Storage: Dùng để lưu trữ các tập tin hình ảnh hoặc video và cho phép truy cập thông qua ứng dụng được phát triển để lưu và tải tập tin.

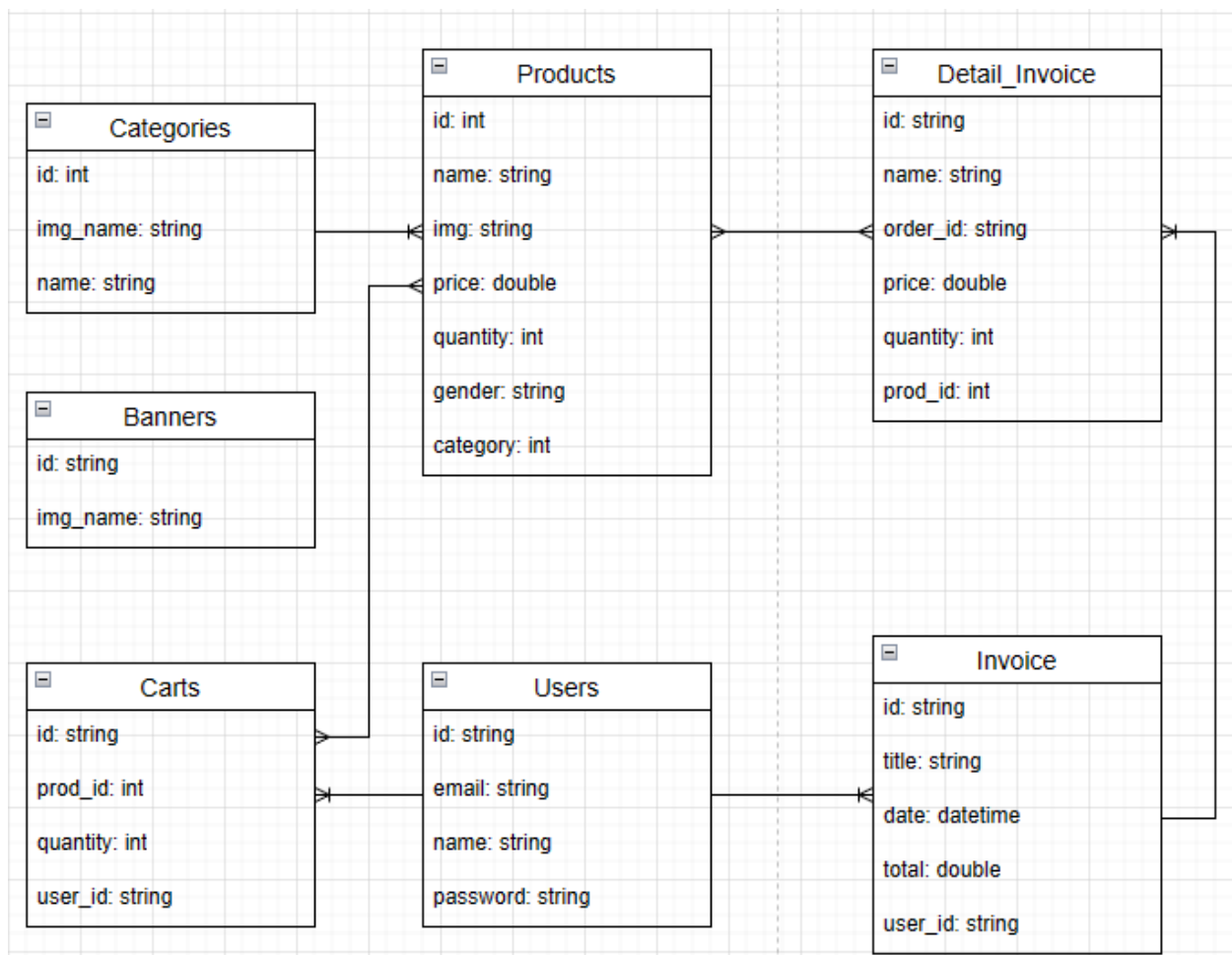
The image shows the Google Cloud Storage console interface. The address bar displays `gs://mobileappclothes.appspot.com > product-img`. There is an 'Upload file' button in the top right corner. Below the header, there is a table listing files:

<input type="checkbox"/>	Name	Size	Type	Last modified
<input type="checkbox"/>	product-000.png	47.02 KB	image/png	Apr 5, 2023
<input type="checkbox"/>	product-001.png	56.21 KB	image/png	Apr 5, 2023
<input type="checkbox"/>	product-002.png	71.12 KB	image/png	Apr 5, 2023
<input type="checkbox"/>	product-003.png	29.04 KB	image/png	Apr 5, 2023

**Hình 4.2:** Lưu trữ hình ảnh trên Cloud Storage

## 2. Cơ sở dữ liệu phi quan hệ

Cơ sở dữ liệu phi quan hệ (NoSQL database) được thiết kế để lưu trữ và quản lý lượng lớn dữ liệu phi cấu trúc và bán cấu trúc (unstructured and semi-structured data). Không giống như cơ sở dữ liệu quan hệ truyền thống (SQL database), thì NoSQL database không sử dụng cột và bảng để lưu trữ dữ liệu mà thay vào đó là sử dụng mô dữ liệu hướng tài liệu (document-oriented data) và key-value. Chính vì vậy mà NoSQL database giúp việc truy cập nhanh dữ liệu nhanh hơn và tiện lợi hơn. NoSQL database có khả năng mở rộng cao và xử lý dữ liệu cực lớn mà không làm giảm hiệu suất của ứng dụng.



**Hình 4.3:** Sơ đồ mô hình dữ liệu phi quan hệ

## CHƯƠNG 5: XÂY DỰNG ỨNG DỤNG

### 1. Giới thiệu mô hình MVVM (Model-View-ViewModel)

MVVM (Model – View – ViewModel) là một mô hình phổ biến trong quy trình phát triển ứng dụng Android. Mô hình này được đề ra với ý tưởng là tách phần giao diện người dùng (View – Activity, Fragment) với phần dữ liệu (Model) bằng cách là thêm một thành phần trung gian ở giữa là ViewModel.

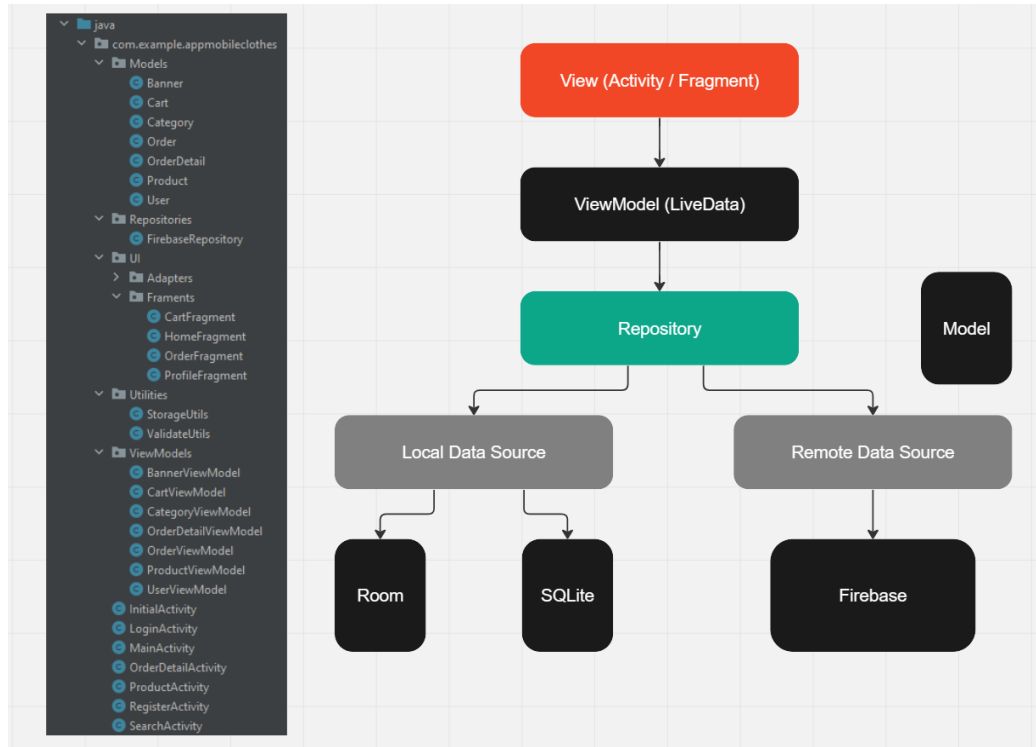
Trong Android Studio có hỗ trợ ViewModel (là một phần của thư viện Lifecycle). Khi ta sử dụng ViewModel trong Android Studio ta sẽ bắt gặp LiveData, MutableLiveData, hàm observe(), sau đây sẽ là những giải thích cơ bản về những khái niệm trên:

- + LiveData: là một class có thể quan sát được dùng để giữ data (observable data holder class).

- + MutableLiveData: là một subclass của LiveData có hỗ trợ method để có thể set value của một LiveData object. Có nghĩa là value của LiveData object có thể được update nhờ có MutableLiveData.

Nhờ vào LiveData và MutableLiveData, khi ta gọi ViewModel tại một View, ta sẽ dùng hàm observe() để quan sát sự thay đổi của LiveData và tự động cập nhật những value của chính nó. Đặc biệt hơn với dịch vụ Firebase Realtime Database đều cung cấp các hàm truy xuất dữ liệu bất đồng bộ, thì sử dụng LiveData là hoàn toàn hợp lý để có thể tự động cập nhật dữ liệu mới mỗi khi dữ liệu được truy xuất thành công từ Firebase.

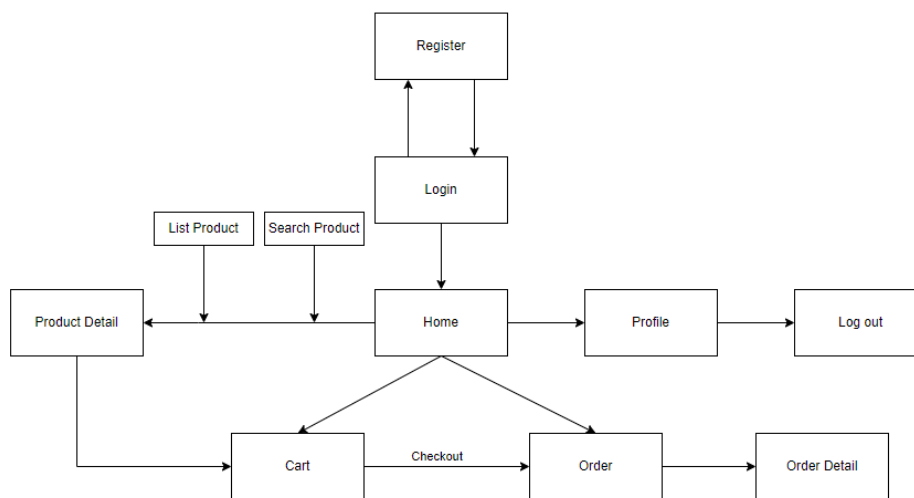
Dựa vào những ý tưởng, đặc tính, cách thức hoạt động trên, MVVM sẽ giúp chúng ta dễ dàng mô-đun hóa những quy trình nghiệp vụ trong ứng dụng. Hơn nữa mô hình sẽ làm giảm đi bớt sự phụ thuộc, ràng buộc cứng giữa những tính năng. Từ đó cấu trúc chương trình sẽ bớt đi những đoạn mã lặp, giúp những tính năng trở nên dễ tái sử dụng hơn và dễ test hơn.



**Hình 5.1:** Cấu trúc của mô hình MVVM

## 2. Cấu trúc ứng dụng

Khi trải qua quá trình đăng nhập, cấu trúc chương trình sẽ chia ra làm một luồng đi chính, đồng thời có các trang được mô tả chi tiết bởi sơ đồ sau:



**Hình 5.2:** Cấu trúc ứng dụng mua quần áo

### 3. Thiết kế Firebase Repository

Để có thể ứng dụng được mô hình MVVM thì ta phải thiết kế một Repository để chứa các hàm generic thao tác với Firebase Realtime Database để thực hiện truy xuất hoặc ghi dữ liệu. Từ đó các ViewModel đều có thể gọi Firebase Repository để tái sử dụng các hàm thao tác này. Dưới đây là giải thích cụ thể các hàm quan trọng:

- + Hàm `getFirebaseData`: Giúp truy xuất tất cả dữ liệu từ Firebase theo node path bất kỳ và đổ vào model bất kỳ. Ví dụ: `getFirebaseData("Products", Product)`.

```
7 usages  🔍 Lỗ Sin Dâu
public <T> LiveData<ArrayList<T>> getFirebaseData(String nodePath, Class<T> dataType) {
    MutableLiveData<ArrayList<T>> firebaseData = new MutableLiveData<>();
    🔍 Lỗ Sin Dâu
    dbRef.child(nodePath).addValueEventListener(new ValueEventListener() {
        2 usages  🔍 Lỗ Sin Dâu
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            ArrayList<T> dataList = new ArrayList<>();
            for (DataSnapshot dataSnapshot : snapshot.getChildren()) {
                T data = dataSnapshot.getValue(dataType);
                dataList.add(data);
            }
            firebaseData.setValue(dataList);
        }

        🔍 Lỗ Sin Dâu
        @Override
        public void onCancelled(@NonNull DatabaseError error) { firebaseData.setValue(null); }
    });
    return firebaseData;
}
```

- + Hàm `getFirebaseSingleData`: Cũng giống như hàm trên nhưng thay vì lấy hết thì chỉ lấy dữ liệu đơn, node path thì phải chi tiết hơn vì phải trả về key cụ thể. Ví dụ: `getFirebaseSingleData("Products/01", Product)`.

```
2 usages  🔍 Lỗ Sin Dâu
public <T> LiveData<T> getFirebaseSingleData(String nodePath, Class<T> dataType) {
    MutableLiveData<T> firebaseData = new MutableLiveData<>();
    🔍 Lỗ Sin Dâu
    dbRef.child(nodePath).addListenerForSingleValueEvent(new ValueEventListener() {
        2 usages  🔍 Lỗ Sin Dâu
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            T data = snapshot.getValue(dataType);
            firebaseData.setValue(data);
        }

        🔍 Lỗ Sin Dâu
        @Override
        public void onCancelled(@NonNull DatabaseError error) { firebaseData.setValue(null); }
    });
    return firebaseData;
}
```

- + Hàm `updateFirebaseData`: Cập nhật trường cụ thể dựa trên node path, key và detail path là trường cụ thể mà mình muốn cập nhật, có hàm overload để giúp in thông báo. Ví dụ: `updateFirebaseData("Products", "01", "name", "T-shirt New")`.

```
1 usage  Nguyễn Ngọc Bảo
public void updateFirebaseData(String nodePath, String key, String detailPath, Object newDetail) {
    dbRef.child(nodePath).child(key).child(detailPath).setValue(newDetail);
}

1 usage  Nguyễn Ngọc Bảo
public void updateFirebaseData(String nodePath, String key, String detailPath, Object newDetail, Context context, String success, String fail) {
    Nguyễn Ngọc Bảo
    dbRef.child(nodePath).child(key).child(detailPath).setValue(newDetail).addOnSuccessListener(new OnSuccessListener<Void>() {
        Nguyễn Ngọc Bảo
        @Override
        public void onSuccess(Void unused) {
            Toast.makeText(context, success, Toast.LENGTH_SHORT).show();
        }
    }).addOnFailureListener(new OnFailureListener() {
        Nguyễn Ngọc Bảo
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(context, fail, Toast.LENGTH_SHORT).show();
        }
    });
}
```

- + Hàm `addFirebaseData`: Để tạo mới dữ liệu thì ta phải push một key mới và dẫn vào nơi key vừa tạo đó để tạo mới dữ liệu, có hàm overload để giúp in thông báo. Ví dụ: `String key = getKey("Categories")`.  
`AddFirebaseData("Categories", newCategory, key)`

```
4 usages  Nguyễn Ngọc Bảo
public String getKey(String nodePath) {
    String key = dbRef.child(nodePath).push().getKey();
    return key;
}

1 usage  Nguyễn Ngọc Bảo
public <T> void addFirebaseData(String nodePath, T data, String key) {
    dbRef.child(nodePath).child(key).setValue(data);
}

3 usages  Nguyễn Ngọc Bảo
public <T> void addFirebaseData(String nodePath, T data, String key, Context context, String success, String fail) {
    Nguyễn Ngọc Bảo
    dbRef.child(nodePath).child(key).setValue(data).addOnSuccessListener(new OnSuccessListener<Void>() {
        Nguyễn Ngọc Bảo
        @Override
        public void onSuccess(Void unused) {
            Toast.makeText(context, success, Toast.LENGTH_SHORT).show();
        }
    }).addOnFailureListener(new OnFailureListener() {
        Nguyễn Ngọc Bảo
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(context, fail, Toast.LENGTH_SHORT).show();
        }
    });
}
```

## 4. Các chức năng chính trong ứng dụng

### 4.1. Đăng ký tài khoản

```
+ Nguyễn Ngọc Bảo +2
registerBtn.setOnClickListener(new View.OnClickListener() {
    + Nguyễn Ngọc Bảo +2
    @Override
    public void onClick(View view) {
        ValidateUtils validator = new ValidateUtils();
        if (mEmail.length() > 0 && validator.isValidEmail(mEmail)) {
            if (validator.isValidLenght(mUserName) && validator.isValidLenght(mPassword)) {
                if (mConfirm.length() > 0 && validator.isValidConfirmPassword(mConfirm, mPassword)) {
                    String key = userModel.getUserKey();
                    User us = new User(key, mUserName, mEmail, mPassword);

                    userModel.addUser(us, key, getBaseContext(), success: "Sign up successfully", fail: "Sign up failed");

                    Intent intent = new Intent(getBaseContext(), LoginActivity.class);
                    startActivity(intent);
                } else {
                    Toast.makeText( context: RegisterActivity.this, text: "Incorrect confirm password", Toast.LENGTH_SHORT).show();
                }
            } else {
                Toast.makeText( context: RegisterActivity.this, text: "Invalid username or password (min_length=6)", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText( context: RegisterActivity.this, text: "Invalid Email", Toast.LENGTH_SHORT).show();
        }
    }
});

+ Duy Nguyen +1
signIn.setOnClickListener(new View.OnClickListener() {
    + Duy Nguyen +1
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(getBaseContext(), LoginActivity.class);
        startActivity(intent);
    }
});
```

Phân tích:

- Khi người dùng ấn nút đăng ký, ứng dụng sẽ validate dữ liệu trong các ô input. Nếu validate thành công (nếu thất bại sẽ hiện toast thông báo), tạo idUser ngẫu nhiên nhờ hàm getUserKey trong class UserViewModel.
- Tạo user mới thông qua class User và gọi hàm addUser trong class UserViewModel. Dữ liệu sẽ được add vào firebase.
- Đăng ký thành công, sẽ chuyển sang trang login. Nếu thất bại sẽ hiện toast thông báo.



## 4.2. Đăng nhập

```

    btnLogin.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            ValidateUtils validator = new ValidateUtils();
            if (email.getText().toString().length() > 0 && validator.isValidEmail(email.getText().toString())) {
                if (validator.isValidLength(password.getText().toString())) {
                    userModel getUsersLiveData().observe(LoginActivity.this, users -> {
                        for (User user : users) {
                            if (email.getText().toString().equals(user.getEmail())) {
                                if (password.getText().toString().equals(user.getPassword())) {
                                    Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                                    intent.putExtra("id", user.getId());
                                    intent.putExtra("username", user.getName());
                                    intent.putExtra("email", user.getEmail());

                                    //Save login
                                    SharedPreferences.Editor editor = mPreferences.edit();
                                    editor.putBoolean("isLoggedIn", true);
                                    editor.putString("userId", user.getId());
                                    editor.apply();
                                    startActivity(intent);
                                    finish();
                                    break;
                                } else {
                                    Toast.makeText(LoginActivity.this, "Incorrect password", Toast.LENGTH_LONG).show();
                                    break;
                                }
                            }
                        }
                    });
                } else {
                    Toast.makeText(LoginActivity.this, "Minimum password length of 6 characters required", Toast.LENGTH_LONG).show();
                }
            } else {
                Toast.makeText(LoginActivity.this, "Invalid email", Toast.LENGTH_LONG).show();
            }
        }
    });
}

```

Phân tích:

- Người dùng ấn vào nút đăng nhập, ứng dụng sẽ validate các trường input. Nếu thất bại, hiện toast thông báo. Nếu thành công, lấy tất cả user từ firebase thông qua hàm getUsersLiveData từ UserViewModel được khởi tạo trước đó.
- Data trả về là 1 arrayList. Tiến hành lặp từng user trong mảng và so sánh từng email. Nếu email giống nhau sẽ so sánh mật khẩu. Đăng nhập thành công nếu đúng mật khẩu của email đó và chuyển sang trang chủ. Nếu thất bại sẽ hiện toast thông báo.
- Khi đăng nhập thành công, tiến hành gửi thông tin của user vào các trang tiếp theo thông qua hàm intent.put(). Và lưu dữ liệu lại, lần vào ứng dụng sau, user không cần đăng nhập lại.

### 4.3. Chỉnh sửa thông tin user

```
saveInfo.setOnClickListener(new View.OnClickListener() {
    Nguyễn Ngọc Bảo
    @Override
    public void onClick(View view) {
        userRef.child(id).child(pathString: "name").setValue(editUsername.getText().toString());
        Toast.makeText(getContext(), text: "Update info successfully", Toast.LENGTH_LONG).show();
    }
});
```

Phân tích:

- Khi người dùng ấn nút save, ứng dụng sẽ tiến hành setValue của field 'name' mà người dùng đã cập nhật lại.
- Từ collection User, tìm user đang đăng nhập bằng id nhận được từ trang home. Từ đó, khi người dùng update name sẽ lưu đúng theo user đó.

### 4.4. Thêm vào giỏ hàng

```
cartArrayList = new ArrayList<Cart>();

cartViewModel = new ViewModelProvider( owner: this).get(CartViewModel.class);
cartViewModel.getCartsLiveData().observe( owner: this, carts -> {
    if (carts != null) {
        cartArrayList = carts;
    }
});
userId = getIntent().getStringExtra( name: "userId");

product = (Product) getIntent().getSerializableExtra( name: "data");

case R.id.bt_add2cart: {
    if (cartArrayList.size() > 0) {
        ArrayList<Cart> cartArrayListById = CartViewModel.getCartsByUserId(cartArrayList, userId);
        for (Cart cart : cartArrayListById) {
            if (cart.getProd_id() == product.getId()) {
                int newQuan = quantity + cart.getQuantity();
                CartViewModel.updateCart(cart.getId(), newQuan, context: this, success: "You have been added this product successfully", fail: "Fail");
                Intent intent = new Intent( packageContext: this, MainActivity.class);
                intent.putExtra( name: "id", userId);
                startActivity(intent);
                return;
            }
        }
    }
    addCart();
    break;
}

usage Nguyễn Ngọc Bảo
public void addCart() {
    String key = cartViewModel.getCartKey();
    Cart newCart = new Cart(key, userId, product.getId(), quantity);
    cartViewModel.addCart(newCart, key, getBaseContext(), success: "You have been added this product successfully", fail: "Fail");
    Intent intent = new Intent( packageContext: this, MainActivity.class);
    intent.putExtra( name: "id", userId);
    startActivity(intent);
}
```

Phân tích:

- Tại Product Activity sẽ gồm: CartViewModel được gọi để lấy toàn bộ Cart, user id, thông tin của product hiện tại
- Khi người dùng tại trang chi tiết sản phẩm và click vào nút thêm vào giỏ hàng. Thì chương trình sẽ kiểm tra xem product đó đã tồn tại trong Cart chưa, nếu tồn tại rồi thì cộng dồn số lượng lên nếu chưa thì thêm mới.

#### 4.5. Thanh toán

```
OrderViewModel orderViewModel = new OrderViewModel();
String key = orderViewModel.getOrderKey();
Order order = new Order(key, strDate, title: "Invoice", user_id, total);

orderViewModel.addOrder(order, key, getContext(), success: "Purchase successfully", fail: "Purchase failed");

//add Order Detail into firebase
for (Cart cart : cartArrayList) {
    OrderDetailViewModel orderDetailViewModel = new OrderDetailViewModel();
    String detailKey = orderDetailViewModel.getOrderDetailKey();

    Product product = ProductViewModel.getProductByIdFromList(productArrayList, cart.getProd_id());

    OrderDetail orderDetail = new OrderDetail(detailKey, key, cart.getProd_id(), product.getName(),
        cart.getQuantity(), product.getPrice());
    orderDetailViewModel.addOrderDetail(orderDetail, detailKey);

    //Delete Cart
    CartViewModel.deleteCart(cart.getId());
}
```

Phân tích:

- Khi người dùng ấn nút Checkout. idOrder sẽ được tạo ngẫu nhiên nhờ hàm getOrderKey
- Tạo Order mới và đưa vào firebase nhờ hàm addOrder
- Đồng thời chúng ta cũng phải lưu Order Detail cho Order, bằng cách tạo vòng lặp for để lấy từng data Cart trong cartArrayList hiện tại của userId
  - + Tạo detailKey ngẫu nhiên nhờ getOrderDetailKey
  - + Sử dụng hàm getProductByIdFromList để lấy Product theo prod\_id
  - + Tạo OrderDetail và đưa vào firebase theo addOrderDetail
  - + Xóa hết Cart đã thanh toán của người dùng hiện tại

## 4.6. Tìm kiếm sản phẩm

Tại trang Home Fragment, người dùng sẽ có 3 cách để chuyển qua trang Search Activity:

- + Cách 1: nhập từ khóa trên thanh tìm kiếm và submit

```
searchView = contentView.findViewById(R.id.searchView);
searchView.clearFocus();

// Lỗi Sin Dấu +1
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    1 usage  + Lỗi Sin Dấu +1
    @Override
    public boolean onQueryTextSubmit(String productName) {
        Intent i = new Intent(getActivity(), SearchActivity.class);
        i.putExtra( name: "productName", productName);
        i.putExtra( name: "userId", userId);
        startActivity(i);
        return true;
    }

    1 usage  + Lỗi Sin Dấu
    @Override
    public boolean onQueryTextChange(String newText) { return false; }
});
```

Phân tích:

- Khi người dùng nhập từ khóa mà muốn tìm kiếm vào và submit thì chương trình sẽ put từ khóa đó cùng với user id hiện tại qua trang Search Activity và hiện ra toàn bộ sản phẩm không phân biệt loại phù hợp với từ khóa.

- + Cách 2: click vào các icon loại sản phẩm để tìm kiếm theo loại

```
//Retrieve categories data
categoryViewModel = new ViewModelProvider( owner: this).get(CategoryViewModel.class);
categoryViewModel.getCategoriesLiveData().observe(getViewLifecycleOwner(), categories -> {
    if (categories != null) {
        categoryAdapter.setCategories(categories);
    }
});
```

```
+ Lỗi Sin Dấu
@Override
public void onClick(View view) {
    Intent i = new Intent(context, SearchActivity.class);
    i.putExtra( name: "categoryId", id);
    context.startActivity(i);
}
```

Phân tích:

- Khi người dùng muốn tìm sản phẩm theo loại thì click vào các icon loại chương trình sẽ put category id qua Search Activity và hiện ra toàn bộ sản phẩm theo loại.

### + Cách 3: Ấn vào nút See All

```
btn_seeAll = contentView.findViewById(R.id.btn_seeAll);
btn_seeAll.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent i = new Intent(getActivity(), SearchActivity.class);
        i.putExtra("name", "userId", userId);
        startActivity(i);
    }
});
```

Phân tích:

- Khi người dùng muốn xem toàn bộ sản phẩm thì người dùng click vào nút See All chương trình sẽ put user id qua Search Activity và hiện toàn bộ sản phẩm.

Tại trang Search Activity, là nơi chính để thực hiện truy vấn tìm kiếm. Hơn thế nữa, bất kể người dùng sử dụng cách nào trong 3 cách trên thì tại đây đều cho phép người dùng tiếp tục tìm kiếm theo từ khóa trên thanh search và tìm kiếm theo giới tính.

```
searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String productName) {
        productViewModel.searchProducts(productName, categoryId, gender).observe(SearchActivity.this, products -> {
            if (products != null) {
                productAdapter.setProducts(products);
            }
        });
        return true;
    }

    @Override
    public boolean onQueryTextChange(String productName) {
        productViewModel.searchProducts(productName, categoryId, gender).observe(SearchActivity.this, products -> {
            if (products != null) {
                productAdapter.setProducts(products);
            }
        });
        return true;
    }
});
```

Phân tích:

- Nhận các dữ liệu được put từ Home Fragment thông qua intent (user id, category id, từ khóa), tùy vào cách người dùng tìm kiếm mà sẽ có những trường không được put qua mà sẽ có giá trị mặc định ở trang này.
- Dựa vào những thông tin đầu vào chương trình sẽ thực hiện truy vấn và trả về sản phẩm phù hợp.

## CHƯƠNG 6: QUY TRÌNH THỰC THI

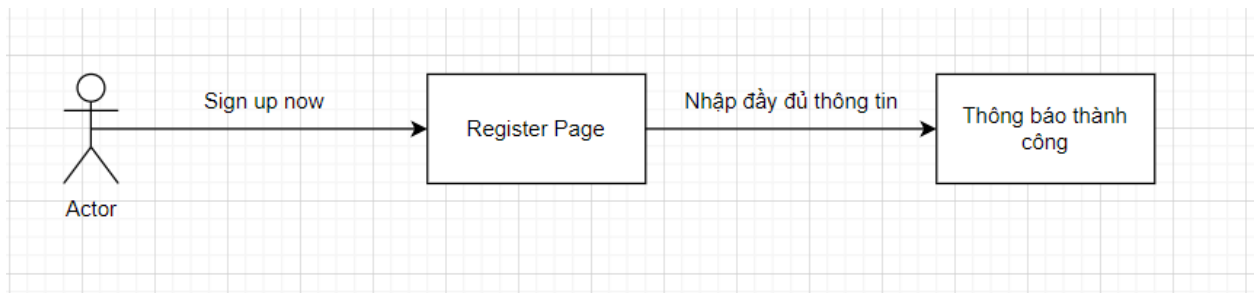
### 1. Quy trình đăng ký tài khoản

**Mô tả:**

**B1:** User vào App. Ấn vào “Sign up now” để chuyển sang trang đăng ký

**B2:** Nhập các thông tin được yêu cầu (email,username,password) và ấn nút Register

**B3:** Hệ thống thông báo lỗi thì đăng ký chưa thành công, còn đăng ký thành công thì hệ thống sẽ chuyển đến trang Đăng nhập.



*Hình 6.1: Quy trình đăng ký tài khoản*

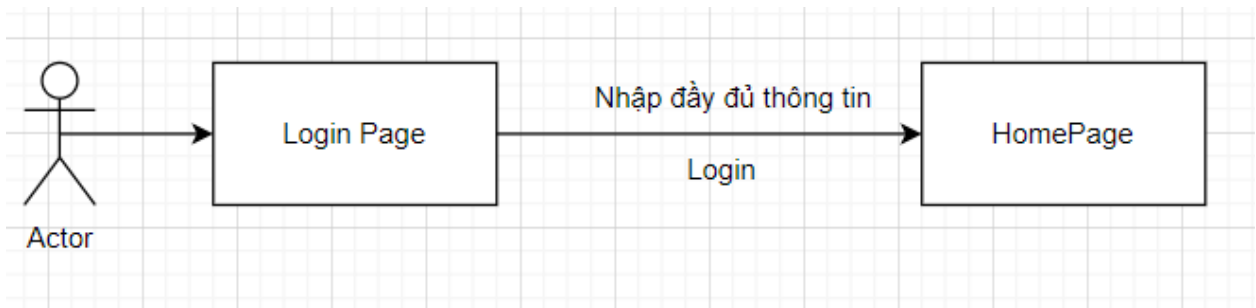
### 2. Quy trình đăng nhập tài khoản

**Mô tả:**

**B1:** User vào app sẽ chuyển đến trang login

**B2:** User nhập email và password và bấm vào Login.

**B3:** Hệ thống thông báo lỗi thì đăng nhập chưa thành công, còn đăng nhập thành công thì hệ thống sẽ chuyển đến trang Homepage.



*Hình 6.2: Quy trình đăng nhập tài khoản*

### 3. Quy trình xem và thêm sản phẩm vào giỏ hàng

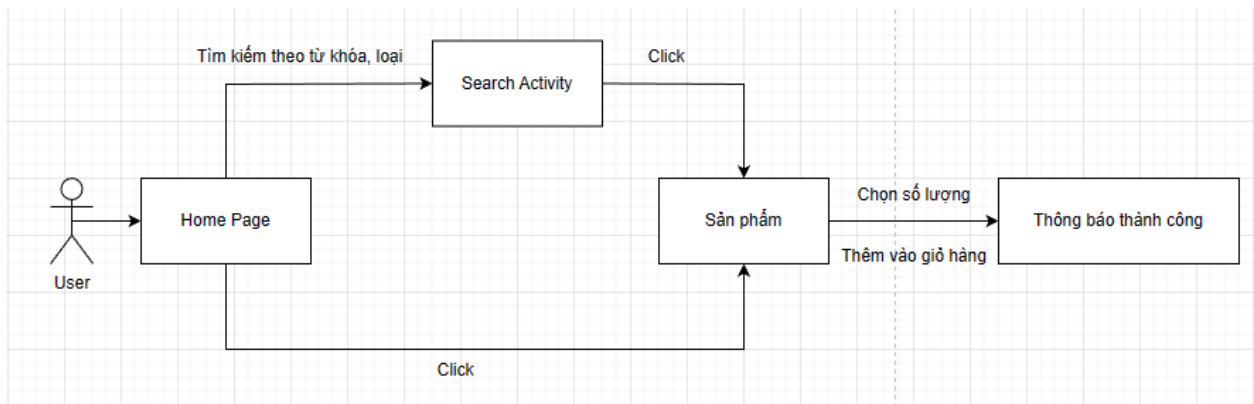
#### Mô tả:

**B1:** Người dùng ở trang chủ có thể tìm kiếm sản phẩm ở trang chủ hoặc tìm kiếm theo từ khóa hoặc chọn loại sản phẩm.

**B2:** Người dùng click vào sản phẩm rồi chọn số lượng

**B3:** Người dùng click vào thêm vào giỏ hàng

**B4:** Thêm vào giỏ hàng thành công thì sẽ thông báo thành công



*Hình 6.3: Quy trình thêm vào giỏ hàng*

### 4. Quy trình thanh toán

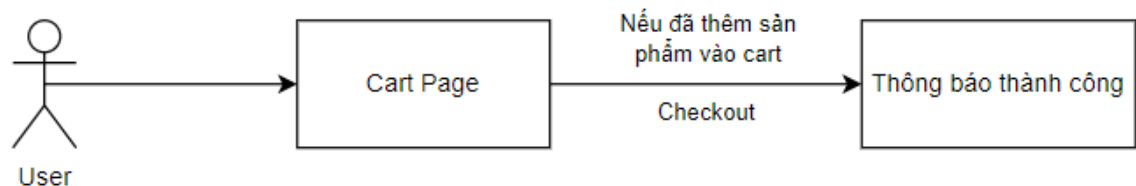
#### Mô tả:

**B1:** Sau khi đã thêm sản phẩm vào giỏ hàng, ấn vào Cart ở thanh navigation

**B2:** Nếu user đã add sản phẩm vào cart thì sẽ hiện ra sản phẩm muốn mua

**B3:** User bấm nút Checkout để thanh toán.

**B4:** Thanh toán thành công thì hệ thống sẽ thông báo thành công.



*Hình 6.4: Quy trình thanh toán*

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 1. Kết luận

### Mức độ hoàn thành

Thiết kế giao diện và chức năng Android: có đầy đủ các giao diện và các chức năng cơ bản như là Đăng ký, Đăng nhập, Trang chủ, Chi tiết sản phẩm, Tìm kiếm và hiển thị kết quả tìm kiếm, Giỏ hàng, Hóa đơn, Chi tiết hóa đơn.

Tạo và kết nối cơ sở dữ liệu trên Firebase: tìm hiểu các đối tượng, thực thể tham gia. Tạo lược đồ use case, phân rã use case và thiết kế cơ sở dữ liệu.

### Hạn chế:

Vẫn là ứng dụng cho mô hình bán lẻ, chưa có nhiều tính năng nổi trội so với thị trường.

## 2. Hướng phát triển

- Thêm nhiều hình thức thanh toán như tiền mặt, thẻ tín dụng, ví điện tử,...
- Phát triển ứng dụng dành cho mô hình sàn thương mại (có nhiều người bán).
- Thêm các chương trình ưu đãi, tặng quà nhân ngày sinh nhật cho khách hàng.
- Cập nhật về điều khoản và tính bảo mật của ứng dụng.
- Ứng dụng hệ thống AI Recommendation để gợi ý sản phẩm phù hợp thị hiếu.



## TÀI LIỆU THAM KHẢO

1. Đặng, Hoàng Thành Ngọc. “Bài giảng Phát triển ứng dụng Mobile - TS. Đặng Ngọc Hoàng Thành.”
1. “W3Schools Tutorials.” *W3Schools Tutorials*, <https://www.w3schools.com/>.
2. <https://developer.android.com/topic/libraries/architecture/viewmodel>
3. <https://developer.android.com/topic/libraries/architecture/livedata>
4. <https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/>
5. <https://www.techighness.com/post/how-to-draw-no-sql-data-model-diagram/>
6. <https://www.solutelabs.com/blog/mobile-ux-design-principles>
7. <https://dribbble.com/shots/16904676-Fashion-eCommerce-App>
8. <https://firebase.google.com/docs>
9. <https://aws.amazon.com/nosql/>

## PHỤ LỤC

Link github: <https://github.com/Dua24/AppMobileClothes>

### Phân công

Họ và tên	Nhiệm vụ	Mức độ
Nguyễn Duy Nguyễn	<ul style="list-style-type: none"><li>- Đăng ký, đăng nhập</li><li>- SharedPreferences login</li><li>- Connect app to firebase</li><li>- Báo cáo chương 2</li></ul>	100%
Lồ Sìn Dậu	<ul style="list-style-type: none"><li>- Home, tìm kiếm, ứng dụng MVVM</li><li>- Báo cáo chương 3</li></ul>	100%
Nguyễn Ngọc Bảo	<ul style="list-style-type: none"><li>- Cart, Order, OrderDetail</li><li>- Báo cáo chương 5</li></ul>	100%
Nguyễn Hoàn Thiện	<ul style="list-style-type: none"><li>- Product, ProductDetail</li><li>- Báo cáo chương 4</li></ul>	100%
Trần Văn Thịnh	<ul style="list-style-type: none"><li>- Báo cáo chương 6</li><li>- Profile user</li></ul>	100%