BÁO CÁO CÁC BƯỚC THỰC HIỆN ỨNG DỤNG FLUTTER VỚI FIREBASE

# 1. Giới thiệu

Ứng dụng này là một ứng dụng Flutter kết hợp với Firebase để quản lý sản phẩm. Người dùng có thể thực hiện các chức năng như thêm, sửa, xóa sản phẩm, tìm kiếm và lọc sản phẩm theo loại. Các dữ liệu sản phẩm được lưu trữ trên Firebase.

# 2. Các bước thực hiện

## 2.1 Thiết lập Firebase

- Tạo một dự án Firebase trên trang Firebase Console (<https://console.firebase.google.com>).

- Cài đặt firebase-tools:

npm install –g firebase-tools

- Đăng nhập vào Firebase bằng tài khoản google:

firebase login

-Kích hoạt flutterfire\_cli:

flutter pub global deactivate flutterfire\_cli

-Kết nối project với firebase:

flutter configure

(Hiện ra danh sách database trong tài khoản Firebase đã tạo, chọn database mình làm)

## Cài đặt Expo và các thư viện cần thiết

### 2.2.1 Cài đặt Expo

#### Bước 1: Cài đặt Expo CLI

*npm install -g expo-cli*

#### Bước 2: Sử dụng Expo CLI để tạo và chạy ứng dụng: ```bash npm install -g expo-cli expo init my-app cd my-app npx expo start ``` - Expo giúp tối ưu hóa quá trình phát triển mà không cần cấu hình quá phức tạp cho các môi trường iOS và Android.

* **Điện thoại thật**: Quét mã QR bằng ứng dụng **Expo Go** (có trên Google Play và App Store).
* **Máy ảo (Android Studio):** Nhấn a để chạy trên **Android emulator**

### 2.2.2 Cài đặt các thư viện

-Thêm thư viện firebase\_core:

flutter pub add firebase\_core

-Thêm thư viện cloud\_firestore:

flutter pub add cloud\_firestore

-Thêm thư viện firebase\_core:

flutter pub add image\_picker

-Thêm thư viện firebase\_storage:

flutter pub add firebase\_storage

## 2.3 Xây dựng giao diện người dùng:

**2.3.1 Màn Hình Trang Chủ (HomeScreen)**

**Màn hình Trang Chủ** hiển thị danh sách sản phẩm của người dùng và cho phép xem, quản lý các sản phẩm (sửa, xóa sản phẩm). Màn hình này được phát triển bằng Flutter và kết nối với Firebase để hiển thị dữ liệu.

**2.3.1.1Cấu trúc giao diện**

* **Danh sách sản phẩm:**
  + Hiển thị danh sách sản phẩm của người dùng được lấy từ Firebase.
  + Mỗi sản phẩm bao gồm tên, giá và hình ảnh (nếu có).
* **Nút hành động:**
  + **Sửa sản phẩm:** Nút "Sửa sản phẩm" sẽ điều hướng người dùng đến màn hình chỉnh sửa sản phẩm.
  + **Xóa sản phẩm:** Nút "Xóa sản phẩm" màu đỏ, khi nhấn sẽ xóa sản phẩm từ Firebase.
  + **Thêm sản phẩm:** Nút "Thêm sản phẩm" nằm ở cuối màn hình, khi nhấn sẽ điều hướng đến màn hình thêm sản phẩm.
    - 1. **Xử lý logic hiển thị**

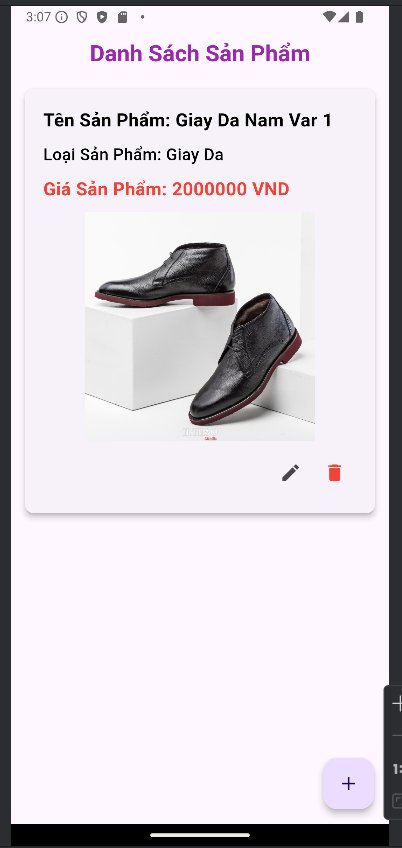
**- Tạo và quản lý giao diện người dùng (UI):**

* Sử dụng StreamBuilder để lắng nghe và cập nhật danh sách sản phẩm từ Firestore theo thời gian thực.
* Mỗi sản phẩm được hiển thị dưới dạng một Card với các thông tin: tên sản phẩm, loại sản phẩm, giá sản phẩm, và hình ảnh.
* Sử dụng ListView.builder để tạo danh sách các sản phẩm từ dữ liệu trong Firestore.
* **Thêm sản phẩm:**
* Sử dụng FloatingActionButton để điều hướng sang trang thêm sản phẩm (Add).
* **Xóa sản phẩm:**
* Nút "Delete" (IconButton) hiển thị hộp thoại xác nhận trước khi xóa sản phẩm.
* Khi người dùng xác nhận xóa, ứng dụng gọi hàm xoaSanPham của FirestoreService để xóa sản phẩm khỏi Firestore.
* Sau khi xóa, hàm setState được gọi để cập nhật lại giao diện.
* **Cập nhật sản phẩm:**
* Mỗi sản phẩm có nút "Edit" (IconButton) cho phép người dùng điều hướng sang trang UpdatePage, truyền vào các thông tin hiện tại của sản phẩm để chỉnh sửa.
* **Hiển thị hình ảnh và xử lý trường hợp không có hình ảnh:**
* Sử dụng Image.network để hiển thị hình ảnh từ URL.
* Nếu URL rỗng, hiển thị thông báo "Không có hình ảnh".

**2.3.1.3 Giao diện**

- Kết quả thực hiện:

* Giao diện HomeScreen sẽ hiển thị danh sách sản phẩm của người dùng, cho phép người dùng tìm kiếm, chỉnh sửa hoặc xóa sản phẩm:



**2.3.2 Màn Hình Thêm Sản Phẩm**

Màn hình Thêm Sản Phẩm cho phép người dùng nhập thông tin sản phẩm mới và lưu vào cơ sở dữ liệu Firebase. Màn hình này được phát triển bằng Flutter và sử dụng Image Picker để chọn hình ảnh cho sản phẩm.

**2.3.2.1Cấu trúc giao diện**

* **Tiêu đề:**
* Hiển thị tiêu đề "Thêm Sản Phẩm".
* **Ô nhập thông tin sản phẩm:**
  + **Tên sản phẩm:**
    - Ô nhập liệu cho phép người dùng nhập tên sản phẩm.

**Giá sản phẩm:**

* + - Ô nhập liệu cho phép người dùng nhập giá sản phẩm.
  + **Loại sản phẩm:**
    - Ô nhập liệu cho phép người dùng nhập loại sản phẩm.
* **Chọn hình ảnh:**
  + Người dùng có thể chọn hình ảnh cho sản phẩm từ thư viện ảnh.
  + Ô vuông trống được sử dụng để hiển thị hình ảnh đã chọn hoặc thông báo "Chưa chọn hình ảnh".
* **Nút hành động:**
  + **Chọn hình ảnh:**
    - Nút "Chọn hình ảnh" cho phép người dùng mở thư viện ảnh để chọn hình ảnh cho sản phẩm.
  + **Thêm sản phẩm:**
    - Nút "Thêm sản phẩm" cho phép người dùng lưu thông tin sản phẩm vào Firebase.

**2.3.2.2 Xử lý logic hiển thị**

**- Giao diện người dùng:**

* Sử dụng TextField để nhập thông tin về sản phẩm: tên sản phẩm, loại sản phẩm, và giá sản phẩm.
* Sử dụng GestureDetector để cho phép người dùng chọn một ảnh sản phẩm từ thư viện.
* Sử dụng ElevatedButton để thêm sản phẩm vào Firestore khi người dùng nhấn nút.

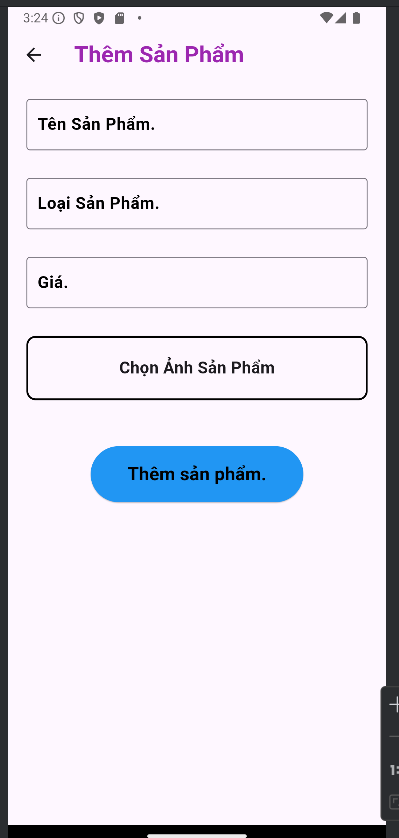
**- Xử lý chọn ảnh:**

* Hàm \_pickImage() được gọi khi người dùng chạm vào khu vực chọn ảnh. Hàm này sử dụng ImagePicker để chọn ảnh từ thư viện của thiết bị.
* Nếu người dùng chọn một ảnh, biến \_image sẽ lưu đường dẫn của ảnh đó và gọi setState để cập nhật giao diện hiển thị thông báo "Ảnh đã chọn".
* **Thêm sản phẩm vào Firestore:**
  + Khi người dùng nhấn nút "Thêm sản phẩm", hàm onPressed sẽ thực hiện các bước:
* Chuyển đổi giá sản phẩm từ chuỗi (String) sang số nguyên (int).
* Gọi hàm themSanPham của FirestoreService để thêm sản phẩm vào Firestore.
* Hàm này nhận các thông tin: tên sản phẩm, loại sản phẩm, giá, và ảnh.
* Sau khi thêm thành công, các trường nhập liệu và biến \_image sẽ được xóa và thiết lập lại.
* Hiển thị thông báo thành công cho người dùng bằng SnackBar.
* **Tương tác với FirestoreService:**
  + FirestoreService là một lớp xử lý thao tác với cơ sở dữ liệu Firestore, cụ thể là hàm themSanPham dùng để lưu thông tin sản phẩm vào Firestore.

**2.3.2.3 Giao diện**

- Kết quả thực hiện:

* Giao diện Add sẽ cho phép người dùng nhập thông tin sản phẩm mới và chọn hình ảnh từ thư viện ảnh. Người dùng có thể thêm sản phẩm mới vào hệ thống:



**2.3.3 Màn Hình Sửa Sản Phẩm (EditProductScreen)**

Màn hình Sửa Sản Phẩm cho phép người dùng chỉnh sửa thông tin của một sản phẩm đã có trong danh sách sản phẩm.

* + - 1. **Cấu trúc giao diện**
* **Tiêu đề:**
  + Hiển thị tiêu đề "Sửa sản phẩm".
* **Ô nhập liệu:**
  + **Tên sản phẩm:** Ô nhập liệu cho phép người dùng chỉnh sửa tên sản phẩm.
  + **Loại sản phẩm:** Ô nhập liệu cho phép người dùng chỉnh sửa loại.
  + **Giá sản phẩm:** Ô nhập liệu cho phép người dùng chỉnh sửa giá sản phẩm, với định dạng số.
* **Chọn hình ảnh:**
  + Một ô vuông trống cho phép người chọn hình ảnh mới cho sản phẩm.
  + Nút "Chọn hình ảnh" nằm dưới ô hình ảnh cho phép người dùng mở thư viện ảnh.
* **Nút lưu thay đổi:**
  + Nút "Lưu thay đổi" cho phép người dùng lưu thông tin đã chỉnh sửa và điều hướng trở lại màn hình chính sau khi thành công.

**2.3.3.2 Xử lý logic hiển thị**

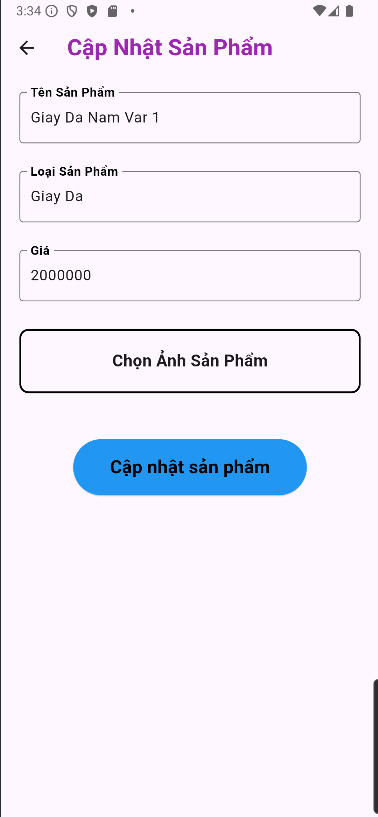
**- Khởi tạo và thiết lập ban đầu:**

* + UpdatePage nhận các tham số đầu vào gồm: docID, currentName, currentType, currentPrice, và currentImageUrl, tương ứng với ID của sản phẩm và các thông tin hiện tại.
  + Trong initState, các giá trị ban đầu được thiết lập cho các TextEditingController để hiển thị thông tin sản phẩm hiện tại.
* **Chọn ảnh mới cho sản phẩm:**
  + Hàm \_pickImage sử dụng ImagePicker để cho phép người dùng chọn ảnh từ thư viện của thiết bị.
  + Nếu người dùng chọn một ảnh, biến \_image sẽ lưu đường dẫn của ảnh và setState được gọi để cập nhật giao diện..
* **Xử lý cập nhật sản phẩm:**
  + Khi người dùng nhấn vào nút "Cập nhật sản phẩm", hàm onPressed sẽ thực hiện các bước sau:
* Chuyển đổi giá sản phẩm từ chuỗi (String) sang số nguyên (int).
* Gọi hàm capnhatSanPham của FirestoreService để cập nhật thông tin sản phẩm trong Firestore. Các tham số gồm: docID, tenSanPham.text, loaiSanPham.text, price, và \_image (nếu có ảnh mới).
* Sau khi cập nhật thành công, Navigator.pop được sử dụng để quay lại trang trước, đồng thời hiển thị thông báo bằng SnackBar.

**2.3.3.3 Giao diện**

- Kết quả thực hiện:

* Giao diện UpdatePage cho phép người dùng dễ dàng chỉnh sửa thông tin sản phẩm, loại sản phẩm, giá, và tải hình ảnh lên:



* Người dùng có thể cập nhật thông tin sản phẩm một cách nhanh chóng và hiệu quả, đồng thời nhận thông báo về tình trạng cập nhật sản phẩm.

# 3. File firestore.dart

Đoạn mã trong file firestore.dart triển khai các thao tác CRUD (Create, Read, Update, Delete) trên Firestore và Firebase Storage cho việc quản lý thông tin sản phẩm trong ứng dụng Flutter.

Dưới đây là chi tiết chức năng của từng phần:

* Thêm sản phẩm (themSanPham):
* Hàm này thêm thông tin sản phẩm vào Firestore và tải lên ảnh của sản phẩm (nếu có) vào Firebase Storage.
* Nếu ảnh được chọn, ảnh sẽ được tải lên và URL của ảnh sẽ được lưu trữ trong Firestore.
* Thông tin sản phẩm như tên, loại, giá và URL của ảnh được lưu vào bộ sưu tập 'sanphams' trong Firestore.
* Lấy thông tin sản phẩm (laythongtinSanPham):
* Hàm này trả về một Stream<QuerySnapshot> để theo dõi và lắng nghe thay đổi trong bộ sưu tập 'sanphams'.
* Sản phẩm được sắp xếp theo tên, loại, giá, và URL của ảnh theo thứ tự giảm dần.
* Cập nhật sản phẩm (capnhatSanPham): Hàm này cho phép cập nhật thông tin của sản phẩm trong Firestore.
* Nếu người dùng chọn một ảnh mới, ảnh đó sẽ được tải lên và URL của ảnh sẽ được cập nhật.
* Nếu không có ảnh mới, URL của ảnh cũ sẽ được giữ lại.
* Sau khi xử lý ảnh, thông tin sản phẩm sẽ được cập nhật trong Firestore.
* Xóa sản phẩm (xoaSanPham):
* Hàm này xóa một sản phẩm khỏi Firestore và xóa ảnh liên quan trong Firebase Storage.
* Trước tiên, hàm sẽ lấy URL của ảnh từ Firestore, sau đó xóa ảnh khỏi Storage.
* Cuối cùng, thông tin của sản phẩm sẽ được xóa khỏi bộ sưu tập 'sanphams'.

# 4. Kết luận

Ứng dụng Flutter kết hợp với Firebase đã hoàn thiện các chức năng cốt lõi của một hệ thống quản lý sản phẩm. Nhờ tích hợp Firebase Realtime Database, ứng dụng đảm bảo khả năng lưu trữ và truy xuất dữ liệu theo thời gian thực, mang đến trải nghiệm nhanh chóng, mượt mà cho người dùng. Các chức năng như đăng nhập, đăng ký, thêm mới, chỉnh sửa, và quản lý sản phẩm được xây dựng hiệu quả, giúp người dùng dễ dàng quản lý danh sách sản phẩm một cách trực quan và thuận tiện.

Một điểm nổi bật là việc xử lý hình ảnh sản phẩm thông qua thư viện image\_picker, cho phép người dùng chọn ảnh từ thiết bị của họ, tạo nên tính tương tác và nâng cao thẩm mỹ cho giao diện. Hệ thống quản lý dữ liệu sản phẩm được thực hiện nhanh chóng, đồng bộ và luôn cập nhật theo thời gian thực, đảm bảo tính chính xác và kịp thời trong việc theo dõi thông tin sản phẩm.

Tổng thể, ứng dụng đã đáp ứng tốt các mục tiêu đề ra, cung cấp một hệ thống quản lý sản phẩm đầy đủ chức năng với khả năng mở rộng trong tương lai. Nhờ sự linh hoạt của Flutter và Firebase, ứng dụng có thể dễ dàng nâng cấp để bổ sung thêm các tính năng phức tạp hơn như phân tích dữ liệu, tối ưu hóa trải nghiệm người dùng, và tích hợp thêm các dịch vụ khác. Đây là một nền tảng vững chắc để phát triển thêm và nâng cao hiệu suất của ứng dụng trong các phiên bản tiếp theo.

LINK SOURCE CODE: https://github.com/TruyenGau/gk-react-native/tree/develop