**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: LẬP TRÌNH MOBILE**

**ĐỀ TÀI: XÂY DỰNG APP QUẢN LÝ ALBUM ẢNH**

**Giáo viên hướng dẫn: Phạm Văn Tiệp và Trần Vũ Đại**

**Sinh viên thực hiện:**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Mã sinh viên** | **Họ và tên** | **Lớp** |
| 1 | 1671020041 | Nguyễn Thanh Bình | CNTT16-01 |

**Hà Nội, năm 2025**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÀI TẬP LỚN**

**TÊN HỌC PHẦN: LẬP TRÌNH MOBILE**

**ĐỀ TÀI: XÂY DỰNG APP QUẢN LÝ ALBUM ẢNH**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Điểm** | |
| **Bằng Số** | **Bằng Chữ** |
| **1** | **1671020041** | **Nguyễn Thanh Bình** | **04/11/2004** |  |  |

**CÁN BỘ CHẤM THI**

**Hà Nội, năm 2025**

**LỜI NÓI ĐẦU**

Trong thời đại công nghệ số hiện nay, hình ảnh không chỉ đơn thuần là phương tiện lưu giữ kỷ niệm mà còn đóng vai trò quan trọng trong việc chia sẻ thông tin, thể hiện cảm xúc và cá nhân hóa trải nghiệm người dùng. Với sự phổ biến của các thiết bị di động và máy ảnh kỹ thuật số, số lượng ảnh mà người dùng lưu trữ ngày càng lớn, kéo theo nhu cầu về một hệ thống quản lý ảnh hiệu quả, thuận tiện và dễ sử dụng.

Xuất phát từ thực tế đó, em quyết định thực hiện đề tài “Xây dựng ứng dụng quản lý album ảnh” với mục tiêu tạo ra một nền tảng giúp người dùng dễ dàng lưu trữ, quản lý và tìm kiếm hình ảnh theo cách khoa học. Hệ thống được thiết kế để hỗ trợ các chức năng như tải ảnh lên, phân loại theo album, tìm kiếm thông minh và đồng bộ dữ liệu một cách an toàn.

Trong quá trình phát triển ứng dụng, em đã sử dụng Flutter để xây dựng giao diện người dùng, đảm bảo tính linh hoạt và khả năng chạy trên nhiều nền tảng. Phần backend được phát triển bằng Flask, giúp cung cấp các API hỗ trợ xử lý dữ liệu một cách nhanh chóng và hiệu quả. Hệ thống cơ sở dữ liệu sử dụng MySQL, đảm bảo khả năng lưu trữ và truy xuất dữ liệu lớn một cách ổn định. Với sự kết hợp của các công nghệ trên, ứng dụng không chỉ đáp ứng được nhu cầu quản lý ảnh cá nhân mà còn có thể mở rộng thêm nhiều tính năng trong tương lai.

Trong suốt quá trình nghiên cứu và xây dựng hệ thống, em đã gặp không ít khó khăn, từ việc thiết kế giao diện sao cho trực quan, tối ưu hiệu suất backend, đến việc tổ chức dữ liệu trong cơ sở dữ liệu một cách hợp lý. Tuy nhiên, nhờ sự kiên trì tìm tòi và đặc biệt là sự hướng dẫn tận tình của **Thầy** **Trần Vũ Đại** và **Thầy** **Phạm Văn Tiệp**, em đã từng bước hoàn thiện đề tài. Các thầy không chỉ giúp em định hướng đúng đắn mà còn hỗ trợ em giải quyết những vấn đề kỹ thuật quan trọng trong quá trình triển khai hệ thống.

**MỤC LỤC**

[**LỜI NÓI ĐẦU** 3](#_Toc193818124)

[**Chương 1. TỔNG QUAN VỀ ĐỀ TÀI** 5](#_Toc193818125)

[**1.1.** **Giới thiệu về đề tài** 5](#_Toc193818126)

[1.1.1. Bối cảnh và lý do chọn đề tài 5](#_Toc193818127)

[1.1.2. Tổng quan về công nghệ sử dụng 5](#_Toc193818128)

[**1.2.** **Mục tiêu của đề tài** 6](#_Toc193818129)

[**1.3. Phạm vi của đề tài** 7](#_Toc193818130)

[**Chương 2. CƠ SỞ LÝ THUYẾT** 9](#_Toc193818131)

[**2.1. Tổng quan về Dart** 9](#_Toc193818132)

[2.1.1. Giới thiệu về ngôn ngữ Dart 9](#_Toc193818133)

[2.1.2. Đặc điểm của Dart 9](#_Toc193818134)

[2.1.3. Cấu trúc chương trình trong Dart 9](#_Toc193818135)

[2.1.4. Ứng dụng của Dart trong Flutter 10](#_Toc193818136)

[**2.2. Giới thiệu về Flutter** 10](#_Toc193818137)

[2.2.1. Flutter là gì? 10](#_Toc193818138)

[2.2.2. Kiến trúc của Flutter 10](#_Toc193818139)

[2.2.3. Ưu điểm của Flutter 10](#_Toc193818140)

[2.2.4. Widget trong Flutter 10](#_Toc193818141)

[**2.3. Flask và MySQL** 11](#_Toc193818142)

[2.3.1. Flask là gì? 11](#_Toc193818143)

[2.3.2. Đặc điểm của Flask 11](#_Toc193818144)

[2.3.3. Cấu trúc một ứng dụng Flask cơ bản 11](#_Toc193818145)

[2.3.4. Kết nối Flask với MySQL 12](#_Toc193818146)

[2.3.5. Ứng dụng Flask và MySQL trong đề tài 12](#_Toc193818147)

[**Chương 3. THIẾT KẾ ỨNG DỤNG** 13](#_Toc193818148)

[**3.1. Ý tưởng thiết kế** 13](#_Toc193818149)

[3.1.2. Các chức năng chính 13](#_Toc193818150)

[3.1.2.1. Hệ thống đăng nhập và quản lý tài khoản 13](#_Toc193818151)

[3.1.2.2. Quản lý giao diện 13](#_Toc193818152)

[3.1.2.3. Quản lý album và ảnh 13](#_Toc193818153)

[3.1.2.4. Chế độ Dark Mode 14](#_Toc193818154)

[3.1.3 Tổng kết 14](#_Toc193818155)

[**3.2. Sơ đồ ứng dụng** 14](#_Toc193818156)

[**3.3. Giao diện các trang chính** 14](#_Toc193818157)

[3.3.1. Trang đăng nhập và đăng ký 14](#_Toc193818158)

[3.3.2. Trang Album\_List 18](#_Toc193818159)

[3.3.4. Trang thêm Album 21](#_Toc193818160)

[3.3.5 Trang chi tiết Album 24](#_Toc193818161)

[3.3.6 Trang Photo Details. 26](#_Toc193818162)

[3.3.7 Trang Edit Photo 29](#_Toc193818163)

[3.3.8 Trang Favorite 32](#_Toc193818164)

[3.3.9 Trang All Photos 34](#_Toc193818165)

[3.3.10 Trang tìm kiếm ảnh bằng tiêu đề và tags. 38](#_Toc193818166)

[3.3.11 Trang đổi mật khẩu 43](#_Toc193818167)

[3.3.12 Giao diện Darkmode 45](#_Toc193818168)

[**3.4. Các API liên quan** 47](#_Toc193818169)

[3.4.1 . AddAlbumProvider 47](#_Toc193818170)

[3.4.2. AddPhotoProvider 47](#_Toc193818171)

[3.4.3 AlbumProvider 48](#_Toc193818172)

[3.4.4 AllPhotosProvider 49](#_Toc193818173)

[3.4.5 FavoritesProvider 49](#_Toc193818174)

[3.4.6 PhotoListProvider 50](#_Toc193818175)

[3.4.7 SearchProvider 51](#_Toc193818176)

[3.4.8 PhotoDetailProvider 51](#_Toc193818177)

[**KẾT LUẬN** 53](#_Toc193818178)

# **Chương 1. TỔNG QUAN VỀ ĐỀ TÀI**

* 1. **Giới thiệu về đề tài**
     1. **Bối cảnh và lý do chọn đề tài**

Trong thời đại công nghệ số phát triển mạnh mẽ, việc chụp ảnh và lưu trữ hình ảnh đã trở thành một nhu cầu thiết yếu đối với mọi người. Với sự phát triển của các thiết bị di động, máy ảnh kỹ thuật số và nền tảng mạng xã hội, số lượng ảnh mà người dùng chụp và lưu trữ ngày càng nhiều. Tuy nhiên, việc quản lý số lượng lớn hình ảnh một cách khoa học, hiệu quả và tiện lợi vẫn là một thách thức lớn.

Hiện nay, nhiều người dùng thường lưu trữ ảnh trên thiết bị cá nhân như điện thoại, máy tính hoặc các nền tảng lưu trữ đám mây như Google Photos, iCloud, OneDrive. Tuy nhiên, một số hạn chế của các phương thức lưu trữ này bao gồm:

* Khó khăn trong việc tìm kiếm ảnh khi số lượng ảnh quá nhiều.
* Ảnh bị phân tán trên nhiều thiết bị và nền tảng khác nhau, gây bất tiện trong việc quản lý.
* Một số dịch vụ lưu trữ có giới hạn dung lượng hoặc yêu cầu trả phí để mở rộng.

Vì vậy, một ứng dụng quản lý album ảnh giúp người dùng tổ chức, tìm kiếm và lưu trữ hình ảnh một cách thuận tiện là rất cần thiết. Xuất phát từ thực tế đó, đề tài **“Xây dựng ứng dụng quản lý album ảnh”** được thực hiện nhằm phát triển một hệ thống hỗ trợ quản lý hình ảnh cá nhân một cách dễ dàng và hiệu quả.

* + 1. **Tổng quan về công nghệ sử dụng**

Trong đề tài này, ứng dụng được xây dựng dựa trên các công nghệ hiện đại để đảm bảo hiệu suất và trải nghiệm người dùng tốt nhất:

* **Flutter**: Được sử dụng để phát triển giao diện ứng dụng trên nền tảng di động. Flutter là một framework phát triển ứng dụng đa nền tảng do Google phát triển, hỗ trợ xây dựng giao diện đẹp mắt, linh hoạt và hiệu suất cao.
* **Flask**: Được sử dụng để xây dựng backend và cung cấp API cho ứng dụng. Flask là một microframework nhẹ, linh hoạt và dễ triển khai, phù hợp cho các ứng dụng có backend nhỏ gọn nhưng vẫn đảm bảo hiệu suất cao.
* **MySQL**: Được sử dụng làm hệ quản trị cơ sở dữ liệu để lưu trữ thông tin về người dùng, album và ảnh. MySQL đảm bảo khả năng lưu trữ dữ liệu an toàn, truy vấn nhanh và có thể mở rộng khi cần.
  1. **Mục tiêu của đề tài**

Mục tiêu chính của đề tài là xây dựng một hệ thống quản lý album ảnh hỗ trợ người dùng lưu trữ, phân loại và tìm kiếm hình ảnh một cách dễ dàng. Hệ thống cần đảm bảo tính trực quan, dễ sử dụng, đồng thời có hiệu suất cao để xử lý dữ liệu ảnh.

**Cụ thể, đề tài hướng đến các mục tiêu sau:**

**Mục tiêu chung**

* Xây dựng một ứng dụng quản lý album ảnh có giao diện thân thiện, dễ sử dụng.
* Đảm bảo ứng dụng có hiệu suất cao, xử lý nhanh các thao tác liên quan đến ảnh.
* Cung cấp các tính năng cơ bản như tạo album, tải ảnh lên, tìm kiếm và quản lý ảnh.

**Mục tiêu cụ thể**

* Thiết kế giao diện người dùng với Flutter:
  + Giao diện hiện đại, dễ sử dụng.
  + Hỗ trợ thao tác kéo, thả ảnh vào album.
  + Hiển thị danh sách album, ảnh theo từng danh mục.
* Phát triển hệ thống backend với Flask:
  + Xây dựng API để xử lý các yêu cầu từ ứng dụng di động.
  + Hỗ trợ thao tác thêm, sửa, xóa album và ảnh.
  + Đảm bảo tính bảo mật của hệ thống, chỉ cho phép người dùng có quyền truy cập ảnh của họ.
* Thiết kế và triển khai cơ sở dữ liệu MySQL:
  + Lưu trữ thông tin về album, ảnh, người dùng một cách khoa học.
  + Tối ưu hóa truy vấn để đảm bảo tốc độ xử lý nhanh.
* Hỗ trợ các tính năng nâng cao:
  + Tìm kiếm ảnh theo tên, thẻ (tag) hoặc ngày tháng.
  + Phân loại ảnh theo album, hỗ trợ thao tác kéo thả.
  + Cho phép chỉnh sửa thông tin ảnh như tiêu đề, mô tả.
* Đảm bảo khả năng mở rộng của hệ thống:
  + Thiết kế có thể dễ dàng mở rộng thêm các tính năng nâng cao như AI nhận diện nội dung ảnh.
  + Hỗ trợ khả năng lưu trữ ảnh trên nền tảng đám mây trong tương lai.

## **1.3. Phạm vi của đề tài**

Đề tài tập trung vào việc xây dựng một hệ thống quản lý ảnh cá nhân với các chức năng cơ bản như tạo album, tải ảnh lên, tìm kiếm và hiển thị ảnh. Phạm vi nghiên cứu và phát triển bao gồm:

**Phạm vi về chức năng**

* Giao diện người dùng:
  + Ứng dụng được thiết kế để hoạt động trên thiết bị di động (Android, iOS).
  + Giao diện đơn giản, dễ sử dụng, có thể hiển thị danh sách album và ảnh.
  + Hỗ trợ các thao tác quản lý album như tạo, sửa, xóa album.
* Chức năng quản lý ảnh:
  + Cho phép tải ảnh lên từ thiết bị.
  + Hiển thị danh sách ảnh trong từng album.
  + Hỗ trợ tìm kiếm ảnh theo tên, ngày tải lên hoặc thẻ (tag).
* Xử lý backend:
  + Phát triển API bằng Flask để xử lý các thao tác quản lý ảnh.
  + Đảm bảo tốc độ xử lý nhanh và tính bảo mật cao.
* Cơ sở dữ liệu:
  + Sử dụng MySQL để lưu trữ thông tin album, ảnh, người dùng.
  + Thiết kế bảng dữ liệu hợp lý để tối ưu hóa hiệu suất truy vấn.

**Phạm vi về đối tượng sử dụng**

* Ứng dụng được xây dựng dành cho người dùng cá nhân có nhu cầu lưu trữ và quản lý ảnh.
* Không dành cho doanh nghiệp hoặc hệ thống chia sẻ ảnh công khai.

**Phạm vi về công nghệ**

* Ngôn ngữ và framework sử dụng:
  + Giao diện người dùng: Flutter.
  + Backend: Flask.
  + Cơ sở dữ liệu: MySQL.
* Môi trường triển khai:
  + Ứng dụng chạy trên thiết bị di động (Android, iOS).
  + Backend có thể triển khai trên server hoặc máy chủ đám mây.

Giới hạn của đề tài

* Hệ thống chỉ hỗ trợ quản lý ảnh cá nhân, chưa tích hợp các tính năng chia sẻ ảnh lên mạng xã hội.
* Không tích hợp công nghệ nhận diện hình ảnh nâng cao như AI để phân loại ảnh tự động.
* Chưa hỗ trợ lưu trữ ảnh trên nền tảng đám mây, chỉ hoạt động trên cơ sở dữ liệu cục bộ.

Với phạm vi trên, đề tài tập trung vào việc xây dựng một hệ thống quản lý album ảnh có tính ứng dụng cao, hỗ trợ người dùng cá nhân lưu trữ và tìm kiếm ảnh dễ dàng, đồng thời có thể mở rộng thêm các tính năng nâng cao trong tương lai.

# **Chương 2. CƠ SỞ LÝ THUYẾT**

## **2.1. Tổng quan về Dart**

### 2.1.1. Giới thiệu về ngôn ngữ Dart

Dart là một ngôn ngữ lập trình hiện đại, hướng đối tượng, được Google phát triển nhằm phục vụ cho việc xây dựng ứng dụng đa nền tảng. Dart được thiết kế để tối ưu hóa hiệu suất, dễ đọc, dễ bảo trì và đặc biệt phù hợp với lập trình giao diện người dùng. Ngôn ngữ này được sử dụng phổ biến trong việc phát triển ứng dụng với Flutter – một framework mạnh mẽ giúp tạo ứng dụng chạy trên nhiều nền tảng khác nhau như Android, iOS, Web và Desktop.

### 2.1.2. Đặc điểm của Dart

Dart có nhiều đặc điểm nổi bật, giúp lập trình viên dễ dàng làm việc và phát triển ứng dụng:

* Hướng đối tượng: Dart hỗ trợ lập trình hướng đối tượng với các khái niệm như class, object, inheritance, polymorphism…
* Cú pháp dễ đọc: Cú pháp của Dart có nhiều điểm tương đồng với JavaScript và Java, giúp lập trình viên dễ dàng tiếp cận.
* Quản lý bộ nhớ tự động: Dart có garbage collection giúp tối ưu việc quản lý bộ nhớ.
* Chạy trên nhiều nền tảng: Hỗ trợ biên dịch thành mã máy (native code) cho hiệu suất cao trên di động, đồng thời có thể chạy trên trình duyệt thông qua JavaScript.
* Hỗ trợ lập trình bất đồng bộ: Sử dụng async và await giúp xử lý các tác vụ bất đồng bộ như tải dữ liệu từ API một cách dễ dàng.

### 2.1.3. Cấu trúc chương trình trong Dart

- Một chương trình Dart cơ bản có thể được viết như sau:

A black background with white text

AI-generated content may be incorrect.

- Trong đó:

* void main() là hàm chính của chương trình.
* print('Hello, Dart!'); là lệnh in ra màn hình.

Dart hỗ trợ kiểu dữ liệu đa dạng như số (int, double), chuỗi (String), danh sách (List), tập hợp (Set), và Map (Map<K, V>).

### 2.1.4. Ứng dụng của Dart trong Flutter

Dart là ngôn ngữ chính để phát triển ứng dụng Flutter, giúp lập trình viên xây dựng giao diện đẹp mắt, hoạt động mượt mà trên nhiều nền tảng.

## **2.2. Giới thiệu về Flutter**

### 2.2.1. Flutter là gì?

Flutter là một framework mã nguồn mở do Google phát triển, giúp lập trình viên xây dựng ứng dụng di động, web và desktop từ một codebase duy nhất. Flutter sử dụng ngôn ngữ lập trình Dart và có hiệu suất cao nhờ cơ chế biên dịch trực tiếp xuống mã máy (native code).

### 2.2.2. Kiến trúc của Flutter

- Flutter có cấu trúc gồm ba thành phần chính:

* Framework: Bao gồm các widget dựng sẵn giúp lập trình viên xây dựng giao diện.
* Engine: Chịu trách nhiệm hiển thị giao diện, xử lý đồ họa, tương tác người dùng.
* Embedder: Cho phép ứng dụng chạy trên nhiều nền tảng như Android, iOS, Web, Desktop.

### 2.2.3. Ưu điểm của Flutter

* Giao diện đẹp, linh hoạt: Sử dụng hệ thống widget mạnh mẽ giúp thiết kế giao diện trực quan.
* Hiệu suất cao: Biên dịch trực tiếp xuống mã máy giúp ứng dụng chạy mượt mà.
* Phát triển nhanh: Tính năng "Hot Reload" giúp xem thay đổi ngay lập tức mà không cần khởi động lại ứng dụng.
* Đa nền tảng: Chỉ cần viết một lần, có thể chạy trên Android, iOS, Web và Desktop.

### 2.2.4. Widget trong Flutter

- Flutter sử dụng hệ thống widget để xây dựng giao diện. Một số widget quan trọng:

* StatelessWidget: Widget không thay đổi trong suốt vòng đời ứng dụng.
* StatefulWidget: Widget có thể thay đổi trạng thái trong quá trình chạy.

- Ví dụ một ứng dụng Flutter đơn giản:

**A computer screen shot of a program code

AI-generated content may be incorrect.**

## **2.3. Flask và MySQL**

### 2.3.1. Flask là gì?

Flask là một microframework cho Python, được sử dụng để xây dựng ứng dụng web và API RESTful. Flask có thiết kế tối giản nhưng mạnh mẽ, phù hợp cho các ứng dụng có quy mô từ nhỏ đến lớn.

### 2.3.2. Đặc điểm của Flask

* Nhẹ và linh hoạt: Flask chỉ cung cấp các tính năng cốt lõi, giúp lập trình viên tùy chỉnh theo nhu cầu.
* Hỗ trợ RESTful API: Dễ dàng xây dựng API để giao tiếp với ứng dụng frontend.
* Tích hợp tốt với cơ sở dữ liệu: Flask có thể làm việc với MySQL, PostgreSQL, SQLite…

### 2.3.3. Cấu trúc một ứng dụng Flask cơ bản

**A computer screen shot of text

AI-generated content may be incorrect.**

- Trong đó:

* Flask(\_\_name\_\_): Khởi tạo ứng dụng Flask.
* @app.route('/'): Định nghĩa route cho trang chủ.
* app.run(debug=True): Chạy ứng dụng ở chế độ debug.

### ****2.3.4. Kết nối Flask với MySQL****

Flask có thể kết nối với MySQL bằng thư viện mysql.connector. Đây là một thư viện phổ biến giúp kết nối ứng dụng Python với cơ sở dữ liệu MySQL thông qua giao thức MySQL Connector/Python.

Ví dụ kết nối Flask với MySQL bằng mysql.connector:

A screen shot of a computer program

AI-generated content may be incorrect.

### 2.3.5. Ứng dụng Flask và MySQL trong đề tài

* Flask dùng để xây dựng API giúp giao tiếp giữa ứng dụng Flutter và cơ sở dữ liệu MySQL.
* MySQL được sử dụng để lưu trữ thông tin người dùng, album và ảnh.
* API sẽ hỗ trợ các chức năng như: thêm/sửa/xóa album và ảnh, tìm kiếm ảnh theo tiêu đề và theo tag, hiển thị tất cả ảnh, danh sách ảnh yêu thích, .

# **Chương 3. THIẾT KẾ ỨNG DỤNG**

## **3.1. Ý tưởng thiết kế**

Ứng dụng quản lý album ảnh được thiết kế nhằm hỗ trợ người dùng lưu trữ, sắp xếp và quản lý ảnh cá nhân một cách hiệu quả. Hệ thống bao gồm các chức năng chính như đăng nhập, đăng ký, quản lý album, quản lý ảnh, tìm kiếm ảnh theo tiêu đề và tag, cùng với một số tiện ích mở rộng như chế độ tối (dark mode) và album bí mật.

### 3.1.2. Các chức năng chính

### 3.1.2.1. Hệ thống đăng nhập và quản lý tài khoản

* Đăng ký tài khoản: Người dùng có thể tạo tài khoản mới bằng email và mật khẩu.
* Đăng nhập: Người dùng đăng nhập vào hệ thống bằng email và mật khẩu.
* Quên mật khẩu: Cho phép người dùng đặt lại mật khẩu khi quên thông tin đăng nhập.
* Đổi mật khẩu: Người dùng có thể thay đổi mật khẩu sau khi đăng nhập thành công.

### 3.1.2.2. Quản lý giao diện

Ứng dụng có 4 giao diện chính:

1. Hiển thị tất cả ảnh: Hiển thị danh sách tất cả ảnh mà người dùng đã tải lên hệ thống.
2. Danh sách ảnh yêu thích: Người dùng có thể đánh dấu ảnh yêu thích và xem danh sách ảnh đã đánh dấu.
3. Danh sách album: Bao gồm các chức năng:
   * Tạo album mới.
   * Xóa album.
   * Tìm kiếm album theo tên.
   * Album bí mật: Một album có thể được đặt mật khẩu. Khi quên mật khẩu, người dùng có thể nhấn nút reset để xóa hết ảnh bên trong và cài đặt mật khẩu mới.
4. Tìm kiếm ảnh: Tìm kiếm ảnh theo tiêu đề hoặc theo tag. Người dùng có thể chọn nhiều tag cùng lúc để lọc ảnh theo ý muốn.

### 3.1.2.3. Quản lý album và ảnh

* Thêm ảnh vào album: Người dùng có thể tải ảnh lên từ thiết bị và chọn album để lưu trữ.
* Xóa ảnh: Xóa ảnh khỏi album hoặc khỏi hệ thống.
* Di chuyển ảnh giữa các album: Người dùng có thể chọn một ảnh và chuyển nó sang album khác.
* Xem trước ảnh khi thêm hoặc sửa: Khi người dùng chọn ảnh từ thư viện để tải lên, ứng dụng sẽ hiển thị một ảnh preview nhỏ để giúp họ kiểm tra xem đã chọn đúng ảnh hay chưa.

### 3.1.2.4. Chế độ Dark Mode

* Người dùng có thể bật/tắt chế độ tối (Dark Mode) để thay đổi giao diện theo sở thích cá nhân.

### 3.1.3 Tổng kết

Ứng dụng quản lý album ảnh được thiết kế nhằm mang lại trải nghiệm tốt nhất cho người dùng trong việc lưu trữ và tìm kiếm ảnh cá nhân. Với các chức năng bảo mật như album bí mật có mật khẩu và khả năng khôi phục khi quên mật khẩu, người dùng có thể yên tâm khi lưu trữ ảnh quan trọng. Ngoài ra, các tính năng như tìm kiếm ảnh theo tag, chế độ dark mode và xem trước ảnh khi tải lên giúp nâng cao trải nghiệm sử dụng.

## **3.2. Sơ đồ ứng dụng**

**Sơ đồ hoạt động của đăng nhập:**

**A diagram of a computer program

AI-generated content may be incorrect.**

**Sơ đồ hệ thống sau khi đăng nhập thành công :**

**A diagram of a music album

AI-generated content may be incorrect.**

## **3.3. Giao diện các trang chính**

### 3.3.1. Trang đăng nhập và đăng ký

Cho phép người dùng đăng nhập bằng Email và mật khẩu :

**A screenshot of a login form

AI-generated content may be incorrect.**

**Đoạn mã:**

  Future<void> \_login() async {

    final email = \_emailController.text;

    final password = \_passwordController.text;

    if (email.isEmpty || password.isEmpty) {

      ScaffoldMessenger.of(context).showSnackBar(

        const SnackBar(content: Text('Vui lòng điền đầy đủ thông tin')),

      );

      return;

    }

    setState(() {

      \_isLoading = true;

    });

    final uri = Uri.parse('http://127.0.0.1:5000/api/login');

    try {

      final response = await http.post(

        uri,

        headers: {'Content-Type': 'application/json'},

        body: json.encode({

          'email': email,

          'password': password,

        }),

      );

      if (response.statusCode == 200) {

        final data = json.decode(response.body);

        // Save token, user ID, and user name

        Provider.of<UserProvider>(context, listen: false)

            .login(data['user\_id'], data['username']); // Pass userName

        Navigator.pushReplacement(

          context,

          MaterialPageRoute(

              builder: (context) => MainPage(userId: data['user\_id'])),

        );

      } else {

        final errorData = json.decode(response.body);

        throw Exception('Đăng nhập thất bại: ${errorData['error']}');

      }

    } catch (e) {

      ScaffoldMessenger.of(context).showSnackBar(

        SnackBar(content: Text('Lỗi: $e')),

      );

    } finally {

      setState(() {

        \_isLoading = false;

      });

    }

  }

**Cho phép người dùng đăng ký tài khoản :**

**A screenshot of a phone

AI-generated content may be incorrect.**

**Đoạn mã :**

bool \_isPasswordValid(String password) {

    final hasUppercase = password.contains(RegExp(r'[A-Z]'));

    final hasDigits = password.contains(RegExp(r'[0-9]'));

    final hasMinLength = password.length >= 6;

    return hasUppercase && hasDigits && hasMinLength;

  }

  Future<void> \_register() async {

    final username = \_usernameController.text;

    final email = \_emailController.text;

    final password = \_passwordController.text;

    if (username.isEmpty || email.isEmpty || password.isEmpty) {

      ScaffoldMessenger.of(context).showSnackBar(

        const SnackBar(content: Text('Vui lòng điền đầy đủ thông tin')),

      );

      return;

    }

    if (!\_isPasswordValid(password)) {

      ScaffoldMessenger.of(context).showSnackBar(

        const SnackBar(

            content: Text(

                'Mật khẩu phải có ít nhất 6 ký tự, bao gồm chữ hoa và số')),

      );

      return;

    }

    setState(() {

      \_isLoading = true;

    });

    final uri = Uri.parse('http://127.0.0.1:5000/api/register');

    try {

      final response = await http.post(

        uri,

        headers: {'Content-Type': 'application/json'},

        body: json.encode({

          'username': username,

          'email': email,

          'password': password,

        }),

      );

      if (response.statusCode == 201) {

        ScaffoldMessenger.of(context).showSnackBar(

          const SnackBar(content: Text('Đăng ký thành công!')),

        );

        Navigator.pop(context); // Navigate back to login page

      } else {

        throw Exception('Đăng ký thất bại');

      }

    } catch (e) {

      ScaffoldMessenger.of(context).showSnackBar(

        SnackBar(content: Text('Lỗi: $e')),

      );

    } finally {

      setState(() {

        \_isLoading = false;

      });

    }

  }

### 3.3.2. Trang Album\_List

**A screenshot of a cartoon

AI-generated content may be incorrect.**

**Hiển thị các album và nút thêm album:**

GridView.builder(

                      gridDelegate:

                          const SliverGridDelegateWithFixedCrossAxisCount(

                        crossAxisCount: 2, // Display 2 albums per row

                        crossAxisSpacing: 8.0,

                        mainAxisSpacing: 8.0,

                        childAspectRatio: 1, // Make albums square

                      ),

                      padding:

                          const EdgeInsets.all(8.0), // Keep consistent padding

                      itemCount:

                          filteredAlbums.length + 1, // Add 1 for the "+" card

                      itemBuilder: (context, index) {

                        if (index == filteredAlbums.length) {

                          // Add Album Button

                          return GestureDetector(

                            onTap: \_navigateToAddAlbumPage,

                            child: MouseRegion(

                              cursor: SystemMouseCursors.click,

                              child: Card(

                                color: Colors.grey[300], //  Gray background

                                elevation: 2,

                                shape: RoundedRectangleBorder(

                                  borderRadius: BorderRadius.circular(8.0),

                                ),

                                child: Center(

                                  child: Icon(Icons.add,

                                      size: 50,

                                      color: Theme.of(context).iconTheme.color),

                                ),

                              ),

                            ),

                          );

                        }

                        final album = filteredAlbums[index];

                        final firstPhotoUrl = (album['photos'] != null &&

                                album['photos'].isNotEmpty)

                            ? album['photos'][0]

                                ['photo\_url'] // Use the first photo

                            : null;

                        return Column(

                          crossAxisAlignment: CrossAxisAlignment.stretch,

                          children: [

                            Expanded(

                              child: GestureDetector(

                                onTap: () {

                                  Navigator.push(

                                    context,

                                    MaterialPageRoute(

                                      builder: (context) => PhotoListPage(

                                          albumId: album['album\_id']),

                                    ),

                                  );

                                },

### 3.3.4. Trang thêm Album

A screenshot of a computer

AI-generated content may be incorrect.

**Đoạn mã :**

  TextField(

                        controller: \_titleController,

                        decoration: InputDecoration(

                          labelText: 'Title',

                          filled: true,

                          fillColor: themeProvider.isDarkMode

                              ? Colors.grey[800]

                              : Colors.grey[200],

                          border: OutlineInputBorder(

                            borderRadius: BorderRadius.circular(15),

                            borderSide: BorderSide.none,

                          ),

                          focusedBorder: OutlineInputBorder(

                            borderRadius: BorderRadius.circular(15),

                            borderSide: BorderSide(

                              color: themeProvider.isDarkMode

                                  ? Colors.blueGrey

                                  : Colors.lightBlueAccent,

                              width: 2,

                            ),

                          ),

                        ),

                      ),

                      const SizedBox(height: 8.0),

                      TextField(

                        controller: \_descriptionController,

                        decoration: InputDecoration(

                          labelText: 'Description',

                          filled: true,

                          fillColor: themeProvider.isDarkMode

                              ? Colors.grey[800]

                              : Colors.grey[200],

                          border: OutlineInputBorder(

                            borderRadius: BorderRadius.circular(15),

                            borderSide: BorderSide.none,

                          ),

                          focusedBorder: OutlineInputBorder(

                            borderRadius: BorderRadius.circular(15),

                            borderSide: BorderSide(

                              color: themeProvider.isDarkMode

                                  ? Colors.blueGrey

                                  : Colors.lightBlueAccent,

                              width: 2,

                            ),

                          ),

                        ),

                      ),

                      const SizedBox(height: 16.0),

                      addAlbumProvider.isLoading

                          ? const CircularProgressIndicator()

                          : ElevatedButton(

                              onPressed: () async {

                                await addAlbumProvider.addAlbum(

                                  context,

                                  userProvider

                                      .userId!, // Use logged-in user's ID

                                  \_titleController.text,

                                  \_descriptionController.text,

                                );

                                Navigator.pushAndRemoveUntil(

                                  context,

                                  MaterialPageRoute(

                                    builder: (context) =>

                                        MainPage(userId: userProvider.userId!),

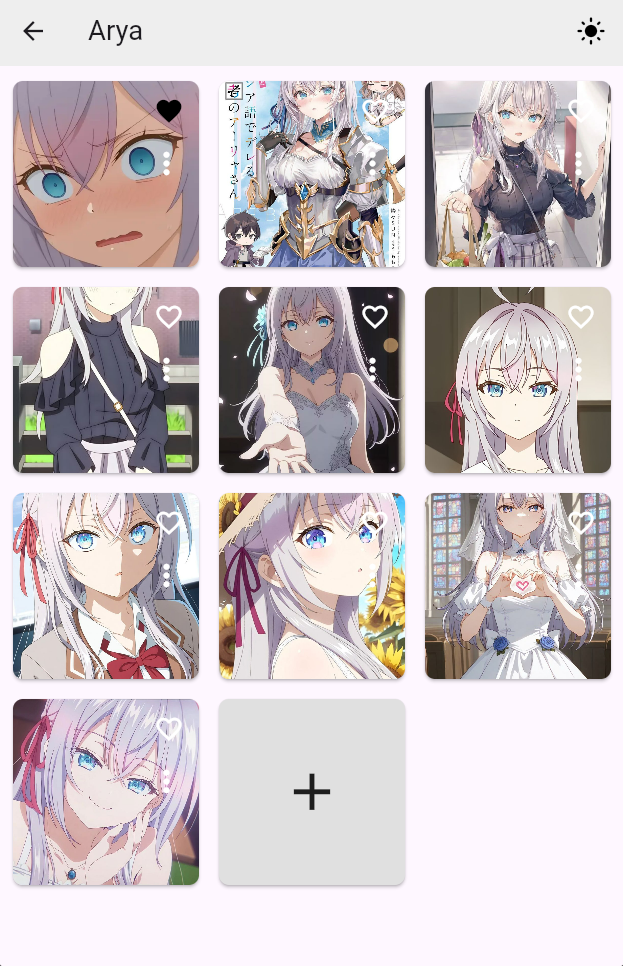
                                  ),

                                  (route) => false,

                                );

                              },

### 3.3.5 Trang chi tiết Album

** A screenshot of a cartoon

AI-generated content may be incorrect.**

**Hiển thị ảnh và nút thêm ảnh:**

GridView.builder(

  gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(

    crossAxisCount: 3, // Hiển thị 3 ảnh mỗi hàng

    crossAxisSpacing: 8.0,

    mainAxisSpacing: 8.0,

  ),

  padding: const EdgeInsets.all(8.0),

  itemCount: photoListProvider.photos.length + 1, // Thêm 1 cho nút "Thêm ảnh"

  itemBuilder: (context, index) {

    if (index == photoListProvider.photos.length) {

      // Nút thêm ảnh

      return GestureDetector(

        onTap: \_navigateToAddPhotoPage,

        child: Card(

          color: Colors.grey[300],

          elevation: 2,

          shape: RoundedRectangleBorder(

            borderRadius: BorderRadius.circular(8.0),

          ),

          child: const Center(

            child: Icon(Icons.add, size: 50),

          ),

        ),

      );

    }

**Hiển thị nút yêu thích:**

final isFavorite = favoritesProvider.favorites

    .any((fav) => fav['photo\_id'] == photo['photo\_id']);

IconButton(

  icon: Icon(

    isFavorite ? Icons.favorite : Icons.favorite\_border,

  ),

  color: isFavorite ? Colors.red : Colors.white,

  onPressed: () {

    if (isFavorite) {

      favoritesProvider.removeFavorite(photo['photo\_id']);

    } else {

      favoritesProvider.addFavorite(

          userProvider.userId!, photo['photo\_id']);

    }

  },

),

**Hiển thị menu tùy chọn (xóa hoặc di chuyển ảnh):**

PopupMenuButton<String>(

  onSelected: (value) {

    if (value == 'delete') {

      \_confirmDeletePhoto(photo['photo\_id']);

    } else if (value == 'move') {

      \_movePhoto(photo['photo\_id']);

    }

  },

  itemBuilder: (context) => [

    const PopupMenuItem(

      value: 'delete',

      child: Row(

        children: [

          Icon(Icons.delete, color: Colors.black),

          SizedBox(width: 8),

          Text('Delete'),

        ],

      ),

    ),

    const PopupMenuItem(

      value: 'move',

      child: Row(

        children: [

          Icon(Icons.drive\_file\_move, color: Colors.blue),

          SizedBox(width: 8),

          Text('Move'),

        ],

      ),

    ),

  ],

  icon: const Icon(

    Icons.more\_vert,

    color: Colors.white,

    size: 28,

  ),

),

### 3.3.6 Trang Photo Details.

**A cartoon of a person with purple hair

AI-generated content may be incorrect.**

**Hiển thị thông tin chi tiết ảnh:**

Expanded(

                child: \_photo['photo\_url'] != null

                    ? Center(

                        child: Image.network(

                          \_photo['photo\_url'],

                          fit: BoxFit.contain,

                        ),

                      )

                    : const Icon(Icons.image, size: 100),

              ),

              const SizedBox(height: 16.0),

              Text(

                \_photo['title'] ?? 'No Title',

                style: const TextStyle(

                  fontSize: 20,

                  fontWeight: FontWeight.bold,

                  color: Colors.blueAccent,

                ),

              ),

              const SizedBox(height: 8.0),

              Text(

                \_photo['description'] ?? 'No Description',

                style: const TextStyle(

                  fontSize: 16,

                  color: Colors.black54,

                ),

              ),

              const SizedBox(height: 8.0),

              Text(

                'Tags: ${\_photo['tags'] ?? 'No Tags'}',

                style: const TextStyle(

                  fontSize: 14,

                  fontStyle: FontStyle.italic,

                  color: Colors.green,

                ),

              ),

              const SizedBox(height: 8.0),

              Text(

                'Size: ${\_photo['size\_kb'] ?? 'Unknown'} KB',

                style: const TextStyle(

                  fontSize: 14,

                  color: Colors.redAccent,

                ),

              ),

**Hiển thị nút yêu thích :**

final isFavorite = favoritesProvider.favorites

    .any((fav) => fav['photo\_id'] == \_photo['photo\_id']);

IconButton(

  icon: Icon(

    isFavorite ? Icons.favorite : Icons.favorite\_border,

  ),

  color: isFavorite ? Colors.red : Colors.white,

  onPressed: () async {

    if (isFavorite) {

      await favoritesProvider.removeFavorite(\_photo['photo\_id']);

    } else {

      await favoritesProvider.addFavorite(

          userProvider.userId!, \_photo['photo\_id']);

    }

    await Provider.of<PhotoDetailProvider>(context, listen: false)

        .checkIfFavorite(\_photo['photo\_id'], userProvider.userId!);

  },

),

**Chuyển hướng qua trang Edit Photos:**

IconButton(

  icon: const Icon(Icons.edit),

  onPressed: () async {

    final result = await Navigator.push(

      context,

      MaterialPageRoute(

        builder: (context) => EditPhotoPage(photo: \_photo),

      ),

    );

    if (result == true) {

      await \_refreshPhotoDetails();

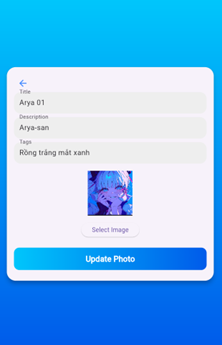
      Navigator.pop(context); // Navigate back automatically

    }

  },

),

### 3.3.7 Trang Edit Photo



**Các trường nhập liệu và nút chọn ảnh:**

TextField(

                        controller: \_titleController,

                        decoration: InputDecoration(

                          labelText: 'Title',

                          filled: true,

                          fillColor: themeProvider.isDarkMode

                              ? Colors.grey[800]

                              : Colors.grey[200],

                          border: OutlineInputBorder(

                            borderRadius: BorderRadius.circular(15),

                            borderSide: BorderSide.none,

                          ),

                          focusedBorder: OutlineInputBorder(

                            borderRadius: BorderRadius.circular(15),

                            borderSide: BorderSide(

                              color: themeProvider.isDarkMode

                                  ? Colors.blueGrey

                                  : Colors.lightBlueAccent,

                              width: 2,

                            ),

                          ),

                        ),

                      ),

                      const SizedBox(height: 8.0),

                      TextField(

                        controller: \_descriptionController,

                        decoration: InputDecoration(

                          labelText: 'Description',

                          filled: true,

                          fillColor: themeProvider.isDarkMode

                              ? Colors.grey[800]

                              : Colors.grey[200],

                          border: OutlineInputBorder(

                            borderRadius: BorderRadius.circular(15),

                            borderSide: BorderSide.none,

                          ),

                          focusedBorder: OutlineInputBorder(

                            borderRadius: BorderRadius.circular(15),

                            borderSide: BorderSide(

                              color: themeProvider.isDarkMode

                                  ? Colors.blueGrey

                                  : Colors.lightBlueAccent,

                              width: 2,

                            ),

                          ),

                        ),

                      ),

                      const SizedBox(height: 8.0),

                      TextField(

                        controller: \_tagsController,

                        decoration: InputDecoration(

                          labelText: 'Tags',

                          filled: true,

                          fillColor: themeProvider.isDarkMode

                              ? Colors.grey[800]

                              : Colors.grey[200],

                          border: OutlineInputBorder(

                            borderRadius: BorderRadius.circular(15),

                            borderSide: BorderSide.none,

                          ),

                          focusedBorder: OutlineInputBorder(

                            borderRadius: BorderRadius.circular(15),

                            borderSide: BorderSide(

                              color: themeProvider.isDarkMode

                                  ? Colors.blueGrey

                                  : Colors.lightBlueAccent,

                              width: 2,

                            ),

                          ),

                        ),

                      ),

                      const SizedBox(height: 16.0),

                      \_imagePreviewUrl == null

                          ? const Text('No image selected')

                          : Image.network(

                              \_imagePreviewUrl!,

                              height: 100,

                              width: 100,

                              fit: BoxFit.cover,

                            ),

                      const SizedBox(height: 16.0),

                      ElevatedButton(

                        onPressed: \_pickImage,

                        child: const Text('Select Image'),

                      ),

### 3.3.8 Trang Favorite

**A screenshot of a cartoon

AI-generated content may be incorrect.**

**Tải ảnh yêu thích khi khởi tạo trang:**

@override

void initState() {

  super.initState();

  Future.microtask(() {

    final userId = Provider.of<UserProvider>(context, listen: false).userId!;

    Provider.of<FavoritesProvider>(context, listen: false)

        .fetchFavorites(userId: userId);

  });

}

**Hiển thị danh sách yêu thích:**

GridView.builder(

  gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(

    crossAxisCount: 3, // Hiển thị 3 ảnh mỗi hàng

    crossAxisSpacing: 8.0,

    mainAxisSpacing: 8.0,

  ),

  padding: const EdgeInsets.all(8.0),

  itemCount: favoritesProvider.favorites.length,

  itemBuilder: (context, index) {

    final photo = favoritesProvider.favorites[index];

    return GestureDetector(

      onTap: () {

        \_navigateToPhotoDetailPage(photo);

      },

      child: Stack(

        children: [

          Card(

            shape: RoundedRectangleBorder(

              borderRadius: BorderRadius.circular(8.0),

            ),

            elevation: 2,

            child: ClipRRect(

              borderRadius: BorderRadius.circular(8.0),

              child: Image.network(

                photo['photo\_url'],

                fit: BoxFit.cover,

                width: double.infinity,

                height: 150,

              ),

            ),

          ),

        ],

      ),

    );

  },

),

### 3.3.9 Trang All Photos

**A screenshot of a cartoon

AI-generated content may be incorrect.**

**Tải danh sách ảnh và danh sách yêu thích:**

@override

void initState() {

  super.initState();

  Future.microtask(() async {

    final userId = Provider.of<UserProvider>(context, listen: false).userId!;

    await Provider.of<AllPhotosProvider>(context, listen: false)

        .fetchAllPhotos(userId: userId);

    await Provider.of<FavoritesProvider>(context, listen: false)

        .fetchFavorites(userId: userId);

  });

}

**Hiển thị danh sách ảnh:**

GridView.builder(

  gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(

    crossAxisCount: 3, // Hiển thị 3 ảnh mỗi hàng

    crossAxisSpacing: 8.0,

    mainAxisSpacing: 8.0,

  ),

  padding: const EdgeInsets.all(8.0),

  itemCount: allPhotosProvider.photos.length,

  itemBuilder: (context, index) {

    final photo = allPhotosProvider.photos[index];

    final isFavorite = favoritesProvider.favorites

        .any((fav) => fav['photo\_id'] == photo['photo\_id']);

    return GestureDetector(

      onTap: () {

        Navigator.push(

          context,

          MaterialPageRoute(

            builder: (context) => PhotoDetailPage(photo: photo),

          ),

        );

      },

      child: Stack(

        children: [

          Card(

            shape: RoundedRectangleBorder(

              borderRadius: BorderRadius.circular(8.0),

            ),

            elevation: 2,

            child: ClipRRect(

              borderRadius: BorderRadius.circular(8.0),

              child: Image.network(

                photo['photo\_url'],

                fit: BoxFit.cover,

                width: double.infinity,

                height: 150,

              ),

            ),

          ),

        ],

      ),

    );

  },

),

**Nút yêu thích ảnh:**

Positioned(

  top: 8,

  right: 8,

  child: IconButton(

    icon: Icon(

      isFavorite ? Icons.favorite : Icons.favorite\_border,

    ),

    color: isFavorite ? Colors.black : Colors.white,

    onPressed: () {

      if (isFavorite) {

        favoritesProvider.removeFavorite(photo['photo\_id']);

      } else {

        favoritesProvider.addFavorite(userProvider.userId!, photo['photo\_id']);

      }

    },

  ),

),

**Menu tùy chọn xóa ảnh:**

Positioned(

  top: 48.0,

  right: 8,

  child: MouseRegion(

    cursor: SystemMouseCursors.click,

    child: PopupMenuButton<String>(

      onSelected: (value) {

        if (value == 'delete') {

          \_deletePhoto(photo['photo\_id']);

        }

      },

      itemBuilder: (context) => [

        const PopupMenuItem(

          value: 'delete',

          child: Row(

            children: [

              Icon(Icons.delete, color: Colors.black),

              SizedBox(width: 8),

              Text('Delete'),

            ],

          ),

        ),

      ],

      icon: const Icon(

        Icons.more\_vert,

        color: Colors.black,

        size: 28,

      ),

    ),

  ),

),

### 3.3.10 Trang tìm kiếm ảnh bằng tiêu đề và tags.

**A screenshot of a cartoon

AI-generated content may be incorrect. A screenshot of a cell phone screen

AI-generated content may be incorrect.**

**Tải danh sách tag khi khởi tạo trang:**

@override

void initState() {

  super.initState();

  Future.microtask(() {

    final userId = Provider.of<UserProvider>(context, listen: false).userId!;

    Provider.of<SearchProvider>(context, listen: false).fetchTags(userId);

  });

  \_animationController = AnimationController(

    duration: const Duration(milliseconds: 300),

    vsync: this,

  );

  \_scaleAnimation = CurvedAnimation(

    parent: \_animationController,

    curve: Curves.easeInOut,

  );

}

**Trường nhập liệu tìm kiếm:**

TextField(

  controller: \_searchController,

  decoration: InputDecoration(

    hintText: 'Search photos...',

    suffixIcon: IconButton(

      icon: const Icon(Icons.search),

      onPressed: \_searchPhotos,

    ),

    border: OutlineInputBorder(

      borderRadius: BorderRadius.circular(16.0),

      borderSide: BorderSide.none,

    ),

    filled: true,

    fillColor: themeProvider.isDarkMode

        ? Colors.grey[800]

        : Colors.grey[200],

  ),

  onChanged: (value) {

    if (value.isEmpty) {

      \_searchPhotos();

    }

  },

),

**Hiện thị danh sách thẻ :**

Wrap(

  spacing: 8.0,

  children: searchProvider.tags.map((tag) {

    final isSelected = \_selectedTags.contains(tag);

    return MouseRegion(

      onEnter: (\_) => \_animationController.forward(),

      onExit: (\_) => \_animationController.reverse(),

      child: ScaleTransition(

        scale: \_scaleAnimation,

        child: FilterChip(

          label: Text(tag),

          selected: isSelected,

          onSelected: (bool selected) {

            setState(() {

              if (selected) {

                \_selectedTags.add(tag);

              } else {

                \_selectedTags.remove(tag);

              }

            });

            \_searchPhotos();

          },

          selectedColor: themeProvider.isDarkMode

              ? Colors.blueGrey

              : Colors.lightBlueAccent,

          backgroundColor: themeProvider.isDarkMode

              ? Colors.grey[700]

              : Colors.grey[300],

          labelStyle: TextStyle(

            color: isSelected ? Colors.white : Colors.black,

          ),

          shape: RoundedRectangleBorder(

            borderRadius: BorderRadius.circular(8.0),

          ),

          elevation: 4,

          shadowColor: Colors.black.withOpacity(0.5),

        ),

      ),

    );

  }).toList(),

),

**Hàm tìm kiếm ảnh:**

void \_searchPhotos() {

  final searchProvider = Provider.of<SearchProvider>(context, listen: false);

  final userId = Provider.of<UserProvider>(context, listen: false).userId!;

  final query = \_searchController.text.trim();

  searchProvider.searchPhotos(query, tags: \_selectedTags, userId: userId);

}

**Hiển thị kết quả ảnh:**

GridView.builder(

  gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(

    crossAxisCount: 3, // Hiển thị 3 ảnh mỗi hàng

    crossAxisSpacing: 8.0,

    mainAxisSpacing: 8.0,

  ),

  padding: const EdgeInsets.all(8.0),

  itemCount: searchProvider.searchResults.length,

  itemBuilder: (context, index) {

    final photo = searchProvider.searchResults[index];

    return GestureDetector(

      onTap: () {

        Navigator.push(

          context,

          MaterialPageRoute(

            builder: (context) => PhotoDetailPage(photo: photo),

          ),

        );

      },

      child: Card(

        shape: RoundedRectangleBorder(

          borderRadius: BorderRadius.circular(8.0),

        ),

        elevation: 2,

        child: ClipRRect(

          borderRadius: BorderRadius.circular(8.0),

          child: Image.network(

            photo['photo\_url'],

            fit: BoxFit.cover,

            width: double.infinity,

            height: double.infinity,

          ),

        ),

      ),

    );

  },

),

### 3.3.11 Trang đổi mật khẩu

**A screenshot of a cartoon

AI-generated content may be incorrect. A screenshot of a phone

AI-generated content may be incorrect.**

**Hàm kiểm tra mật khẩu :**

bool \_isPasswordValid(String password) {

  final hasUppercase = password.contains(RegExp(r'[A-Z]'));

  final hasDigits = password.contains(RegExp(r'[0-9]'));

  final hasMinLength = password.length >= 6;

  return hasUppercase && hasDigits && hasMinLength;

}

**Hàm đổi mật khẩu:**

Future<void> \_changePassword() async {

  final oldPassword = \_oldPasswordController.text;

  final newPassword = \_newPasswordController.text;

  if (oldPassword.isEmpty || newPassword.isEmpty) {

    ScaffoldMessenger.of(context).showSnackBar(

      const SnackBar(content: Text('Vui lòng điền đầy đủ thông tin')),

    );

    return;

  }

  if (!\_isPasswordValid(newPassword)) {

    ScaffoldMessenger.of(context).showSnackBar(

      const SnackBar(

          content: Text(

              'Mật khẩu mới phải có ít nhất 6 ký tự, bao gồm chữ hoa và số')),

    );

    return;

  }

  setState(() {

    \_isLoading = true;

  });

  final userId = Provider.of<UserProvider>(context, listen: false).userId!;

  final uri =

      Uri.parse('http://127.0.0.1:5000/api/users/$userId/change\_password');

  try {

    final response = await http.put(

      uri,

      headers: {'Content-Type': 'application/json'},

      body: json.encode({

        'old\_password': oldPassword,

        'new\_password': newPassword,

      }),

    );

    if (response.statusCode == 200) {

      ScaffoldMessenger.of(context).showSnackBar(

        const SnackBar(content: Text('Đổi mật khẩu thành công!')),

      );

      Navigator.pop(context);

    } else {

      throw Exception('Đổi mật khẩu thất bại');

    }

  } catch (e) {

    ScaffoldMessenger.of(context).showSnackBar(

      SnackBar(content: Text('Lỗi: $e')),

    );

  } finally {

    setState(() {

      \_isLoading = false;

    });

  }

}

### 3.3.12 Giao diện Darkmode

**A screenshot of a cartoon

AI-generated content may be incorrect. A screenshot of a cartoon

AI-generated content may be incorrect.**

**Hàm chuyển đổi chế độ:**

void toggleTheme() {

  \_themeMode =

      \_themeMode == ThemeMode.light ? ThemeMode.dark : ThemeMode.light;

  notifyListeners();

}

**Tích hợp với giao diện người dùng :**

AppBar(

  title: const Text('App Title'),

  backgroundColor: themeProvider.isDarkMode

      ? Colors.grey[850] // Màu nền cho Dark Mode

      : Colors.grey[200], // Màu nền cho Light Mode

  actions: [

    IconButton(

      icon: Icon(

        themeProvider.isDarkMode ? Icons.dark\_mode : Icons.light\_mode,

        color: Colors.black,

      ),

      onPressed: () {

        themeProvider.toggleTheme();

      },

    ),

  ],

),

**Áp dụng chế độ sáng/tối toàn cục:**

MaterialApp(

  themeMode: themeProvider.themeMode,

  theme: ThemeData.light(), // Giao diện cho Light Mode

  darkTheme: ThemeData.dark(), // Giao diện cho Dark Mode

  home: HomePage(),

);

## **3.4. Các API liên quan**

### 3.4.1 . AddAlbumProvider

* **API**: POST /api/albums
  + **Mục đích**: Thêm một album mới.
  + **Tham số**:
    - user\_id: ID của người dùng.
    - title: Tiêu đề album.
    - description: Mô tả album.
  + **Mã liên quan**:

final uri = Uri.parse('http://127.0.0.1:5000/api/albums');

### 3.4.2. AddPhotoProvider

* **API**: POST /api/albums/{albumId}/photos
  + **Mục đích**: Thêm một ảnh mới vào album.
  + **Tham số**:
    - user\_id: ID của người dùng.
    - title: Tiêu đề ảnh.
    - description: Mô tả ảnh.
    - tags: Thẻ liên quan đến ảnh.
    - photo\_url: File ảnh.
  + **Mã liên quan**:

final uri = Uri.parse('http://127.0.0.1:5000/api/albums/$albumId/photos');

### 3.4.3 AlbumProvider

* **API**: GET /api/albums?user\_id={userId}
* **Mục đích**: Lấy danh sách album của người dùng.
* **Tham số**:
* user\_id: ID của người dùng.
  + **Mã liên quan**:

final uri = Uri.parse('http://127.0.0.1:5000/api/albums?user\_id=$userId');

* **API**: GET /api/albums/{albumId}/photos
* **Mục đích**: Lấy danh sách ảnh trong một album.
* **Tham số**:
* albumId: ID của album.
* **Mã liên quan**:

final uri = Uri.parse('http://127.0.0.1:5000/api/albums/$albumId/photos');

* **API**: POST /api/secret\_albums
* **Mục đích**: Thêm một album bí mật.
* **Tham số**:
* user\_id: ID của người dùng.
* title: Tiêu đề album.
* description: Mô tả album.
* password: Mật khẩu bảo vệ album.
* **Mã liên quan**:

final uri = Uri.parse('http://127.0.0.1:5000/api/secret\_albums');

### 3.4.4 AllPhotosProvider

* **API**: GET /api/photos?user\_id={userId}
* **Mục đích**: Lấy danh sách tất cả ảnh của người dùng.
* **Tham số**:
* user\_id: ID của người dùng.
* **Mã liên quan**:

final uri = Uri.parse('http://127.0.0.1:5000/api/photos?user\_id=$userId');

* **API**: DELETE /api/photos/{photoId}
* **Mục đích**: Xóa một ảnh.
* **Tham số**:
* photoId: ID của ảnh cần xóa.
* **Mã liên quan**:

final uri = Uri.parse('http://127.0.0.1:5000/api/photos/$photoId');

### 3.4.5 FavoritesProvider

* **API:**GET /api/favorites?user\_id={userId}
* **Mục đích:** Lấy danh sách ảnh yêu thích của người dùng**.**
* **Tham số:**
* user\_id: ID của người dùng**.**
* **Mã liên quan:**

final uri = Uri.parse('http://127.0.0.1:5000/api/favorites?user\_id=$userId');

* **API:**POST /api/favorite
* **Mục đích:** Thêm một ảnh vào danh sách yêu thích.
* **Tham số:**
* user\_id: ID của người dùng.
* photo\_id: ID của ảnh.
* **Mã liên quan:**

final uri = Uri.parse('http://127.0.0.1:5000/api/favorite');

* **API:**DELETE /api/favorites/{photoId}
* **Mục đích:** Xóa một ảnh khỏi danh sách yêu thích.
* **Tham số:**
* photoId: ID của ảnh cần xóa.
* **Mã liên quan:**

final uri = Uri.parse('http://127.0.0.1:5000/api/favorites/$photoId');

### 3.4.6 PhotoListProvider

* **API**: GET /api/albums/{albumId}/photos
* **Mục đích**: Lấy danh sách ảnh trong một album.
* **Tham số**:
* albumId: ID của album.
* **Mã liên quan**:

final uri = Uri.parse('http://127.0.0.1:5000/api/albums/$albumId/photos');

* **API**: DELETE /api/photos/{photoId}
* **Mục đích**: Xóa một ảnh.
* **Tham số**:
* photoId: ID của ảnh cần xóa.
* **Mã liên quan**:

final uri = Uri.parse('http://127.0.0.1:5000/api/photos/$photoId');

### 3.4.7 SearchProvider

* **API**: GET /api/photos/tags?user\_id={userId}
* **Mục đích**: Lấy danh sách thẻ (tags) của người dùng.
* **Tham số**:
* user\_id: ID của người dùng.
* **Mã liên quan**:

final uri = Uri.parse('http://127.0.0.1:5000/api/photos/tags?user\_id=$userId');

* **API**: GET /api/photos/search
* **Mục đích**: Tìm kiếm ảnh theo tiêu đề và thẻ.
* **Tham số**:
* title: Tiêu đề ảnh.
* tags: Danh sách thẻ.
* user\_id: ID của người dùng.
* **Mã liên quan**:

finaluri=Uri.parse('http://127.0.0.1:5000/api/photos/search?title=$query&tags=$tagsQuery&user\_id=$userId');

### 3.4.8 PhotoDetailProvider

* **API**: GET /api/favorites
* **Mục đích**: Kiểm tra xem ảnh có nằm trong danh sách yêu thích không.
* **Tham số**:
* photo\_id: ID của ảnh.
* user\_id: ID của người dùng.
* **Mã liên quan**:

final uri = Uri.parse('http://127.0.0.1:5000/api/favorites');

* **API**: POST /api/favorites
* **Mục đích**: Thêm ảnh vào danh sách yêu thích.
* **Tham số**:
* photo\_id: ID của ảnh.
* **Mã liên quan**:

final uri = Uri.parse('http://127.0.0.1:5000/api/favorites');

* **API**: DELETE /api/favorites/{photoId}
* **Mục đích**: Xóa ảnh khỏi danh sách yêu thích.
* **Tham số**:
* photoId: ID của ảnh.
* **Mã liên quan**:

final uri = Uri.parse('$uri/$photoId');

**KẾT LUẬN**

**1. Kết quả đạt được**

* Ứng dụng đã hoàn thiện các chức năng chính, bao gồm:
  + **Quản lý album**: Người dùng có thể tạo, xem, và quản lý album (bao gồm cả album bí mật).
  + **Quản lý ảnh**: Hỗ trợ thêm, chỉnh sửa, xóa ảnh và hiển thị chi tiết ảnh.
  + **Tìm kiếm ảnh**: Tìm kiếm ảnh theo tiêu đề hoặc thẻ (tags) với giao diện thân thiện.
  + **Danh sách yêu thích**: Người dùng có thể thêm hoặc xóa ảnh khỏi danh sách yêu thích.
  + **Chế độ sáng/tối (Dark/Light Mode)**: Tích hợp giao diện sáng và tối, mang lại trải nghiệm người dùng tốt hơn.
  + **Đổi mật khẩu**: Hỗ trợ người dùng thay đổi mật khẩu với các tiêu chí bảo mật.
  + **Tích hợp API**: Hệ thống sử dụng REST API để giao tiếp giữa ứng dụng và máy chủ, đảm bảo tính linh hoạt và mở rộng.

**2. Nhược điểm**

* **Hiệu suất**:
  + Khi số lượng ảnh hoặc album lớn, việc tải dữ liệu có thể chậm do thiếu cơ chế phân trang (pagination).
  + Một số thao tác như tìm kiếm hoặc xóa ảnh có thể gây chậm nếu không tối ưu hóa API.
* **Bảo mật**:
  + API chưa được bảo vệ bằng cơ chế xác thực (authentication) hoặc mã hóa dữ liệu, dễ bị tấn công nếu triển khai trên môi trường thực tế.
  + Album bí mật chỉ dựa vào mật khẩu đơn giản, chưa có cơ chế mã hóa mạnh mẽ.
* **Giao diện**:
  + Giao diện người dùng chưa tối ưu hoàn toàn cho các thiết bị có màn hình nhỏ hoặc lớn.
  + Thiếu các thông báo chi tiết khi xảy ra lỗi (ví dụ: lỗi kết nối mạng hoặc lỗi từ máy chủ).
* **Khả năng mở rộng**:
  + Chưa hỗ trợ tải ảnh lên từ các nguồn khác như Google Drive hoặc Dropbox.
  + Chưa có tính năng chia sẻ album hoặc ảnh với người dùng khác.

**3. Hướng phát triển**

* **Cải thiện hiệu suất**:
  + Tích hợp cơ chế phân trang (pagination) để giảm tải dữ liệu khi hiển thị danh sách ảnh hoặc album.
  + Sử dụng bộ nhớ đệm (caching) để tăng tốc độ tải dữ liệu.
* **Tăng cường bảo mật**:
  + Tích hợp cơ chế xác thực người dùng (JWT hoặc OAuth2) để bảo vệ API.
  + Mã hóa dữ liệu nhạy cảm như mật khẩu album bí mật.
  + Sử dụng HTTPS để bảo mật giao tiếp giữa ứng dụng và máy chủ.
* **Nâng cấp giao diện**:
  + Tối ưu giao diện cho các thiết bị di động và máy tính bảng.
  + Thêm các thông báo chi tiết và thân thiện hơn khi xảy ra lỗi.
* **Mở rộng tính năng**:
  + Hỗ trợ chia sẻ album hoặc ảnh qua email hoặc liên kết.
  + Tích hợp các dịch vụ lưu trữ đám mây như Google Drive, Dropbox để tải ảnh lên.
  + Thêm tính năng chỉnh sửa ảnh cơ bản (cắt, xoay, thêm bộ lọc).
* **Triển khai thực tế**:
  + Đưa ứng dụng lên các nền tảng như Google Play Store hoặc Apple App Store.
  + Tối ưu hóa API để hoạt động ổn định trên môi trường thực tế.

Kết luận, ứng dụng đã đáp ứng được các yêu cầu cơ bản về quản lý album và ảnh, mang lại trải nghiệm tốt cho người dùng. Tuy nhiên, để triển khai thực tế và cạnh tranh trên thị trường, cần cải thiện hiệu suất, bảo mật, và mở rộng thêm các tính năng hữu ích.

**TÀI LIỆU THAM KHẢO**

[1] Nielsen, J. (1994). 10 Usability Heuristics for User Interface Design. Retrieved from <https://www.nngroup.com/articles/ten-usability-heuristics/>

[2] Flutter Community. (n.d.). HTTP package for Dart. Retrieved from

https://pub.dev/packages/http

[3] Google Developers. (n.d.). Flutter Documentation. Retrieved from

https://flutter.dev/docs

[4] Pallets Projects. (n.d.). Flask Documentation. Retrieved from

<https://flask.palletsprojects.com/>

[5] Oracle Corporation. (n.d.). MySQL Documentation. Retrieved from

<https://dev.mysql.com/doc/>