TRƯỜNG ĐẠI HỌC ĐẠI NAM

**KHOA CÔNG NGHỆ THÔNG TIN**

# BÀI TẬP LỚN

## HỌC PHẦN: LẬP TRÌNH MOBILE

## ĐỀ TÀI: Xây Dựng Ứng Dụng Đặt Đồ Uống Với Flutter

**Giảng viên: Nguyễn Quang Hưng**

**Lớp : CNTT 15-04**

| TT | Mã SV | Họ và Tên | Ngày sinh |
|---|---|---|---|
| 1 | 1571020257 | Nguyễn Ngọc Đan Trường | 05/12/2003 |
| 2 | 1571020100 | Nguyễn Đức Hiệp | 02/08/2003 |
| 3 | 1571020260 | Dương Văn Tuân | 24/04/2003 |
| 4 | 1571020202 | Nguyễn Vũ Phúc | 24/01/2003 |

**Hà Nội, tháng 4 năm 2024**

TRƯỜNG ĐẠI HỌC ĐẠI NAM

**KHOA CÔNG NGHỆ THÔNG TIN**

# BÀI TẬP LỚN

## HỌC PHẦN: LẬP TRÌNH MOBILE

## ĐỀ TÀI: Xây Dựng Ứng Dụng Đặt Đồ Uống Với Flutter

| TT | Mã SV | Họ và Tên | Điểm | |
|---|---|---|---|---|
| | | | Bằng Số | Bằng Chữ |
| 1 | 1571020257 | Nguyễn Ngọc Đan Trường | | |
| 2 | 1571020100 | Nguyễn Đức Hiệp | | |
| 3 | 1571020260 | Dương Văn Tuân | | |
| 4 | 1571020202 | Nguyễn Vũ Phúc | | |

**CÁN BỘ CHẤM THI**

**Nguyễn Quang Hưng**

**Hà Nội, tháng 4 năm 2024**

# LỜI NÓI ĐẦU

Đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến Trường Đại học Đại Nam đã đưa môn Lập Trình Mobile vào chương trình giảng dạy. Đặc biệt, chúng em xin gửi lời cảm ơn sâu sắc đến giảng viên bộ môn – thầy Nguyễn Quang Hưng đã dạy dỗ, truyền đạt những kiến thức quý báu cho chúng em trong suốt thời gian học tập vừa qua. Trong thời gian tham gia lớp học, nhóm em đã có thêm cho mình nhiều kiến thức bổ ích, tinh thần học tập hiệu quả, nghiêm túc. Đây chắc chắn sẽ là những kiến thức quý báu, là hành trang để chúng em có thể vững bước sau này.

Bộ môn Lập Trình Mobile là môn học thú vị, vô cùng bổ ích và có tính thực tế cao. Đảm bảo cung cấp đủ kiến thức, gắn liền với nhu cầu thực tiễn của sinh viên. Tuy nhiên, do vốn kiến thức còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều bỡ ngỡ. Mặc dù nhóm chúng em đã cố gắng hết sức nhưng bài tập lớn khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong thầy cô xem xét và góp ý để bài tập lớn của chúng em được hoàn thiện hơn.

Nhóm chúng em xin chân thành cảm ơn!

# Mục Lục

# MỤC LỤC HÌNH ẢNH

# Chương 1.KIẾN THỨC CƠ BẢN

## 1.Khái niệm Dart

Dart là một ngôn ngữ lập trình đa mục đích mã nguồn mở đã được đặt nền móng bởi Google. Đây là một khía cạnh của ngôn ngữ lập trình hướng đối tượng đặc trưng bởi cú pháp kiểu C. Tinh thần lập trình hướng đối tượng thể hiện qua việc hỗ trợ giao diện và lớp mở ra khả năng sáng tạo không giới hạn khi đặt lên bàn cân với những ngôn ngữ khác. Sự đa dạng của Dart thể hiện thông qua khả năng phát triển ứng dụng web, di động, máy chủ và máy tính để bàn.

Không chỉ dừng lại ở việc phát triển ứng dụng cho một nền tảng, Dart mở ra cánh cửa cho sự sáng tạo trên cả hai nền tảng quan trọng: Android và iOS. Và trong cuộc hành trình này, Flutter nổi lên như một công cụ vượt trội, được Google chắp cánh. Flutter, một framework ra đời từ Google, trở thành điểm đặc biệt giúp xây dựng những ứng dụng tuyệt đẹp trên cả hai nền tảng bằng một nền tảng mã nguồn duy nhất.

### 1.1.Đặc điểm của Dart

Ứng dụng ngôn ngữ Dart trong lập trình di động không chỉ dừng lại ở việc phát triển ứng dụng cho một nền tảng, Dart mở ra cánh cửa cho sự sáng tạo trên cả hai nền tảng quan trọng: Android và iOS. Và trong cuộc hành trình này, Flutter nổi lên như một công cụ vượt trội, được Google chắp cánh. Flutter, một framework ra đời từ Google, trở thành điểm đặc biệt giúp xây dựng những ứng dụng tuyệt đẹp trên cả hai nền tảng bằng một nền tảng mã nguồn duy nhất.

Từ Dart đến Framework FlutterFlutter đã thu hút được sự chú ý của cộng đồng các nhà phát triển bằng cách giới thiệu các style cho phép việc xây dựng UI đẹp hơn và biểu cảm hơn vì thế mà việc code cũng trở nên thú vị hơn nhiều. Nó kết hợp một số khái niệm quen thuộc với những kinh nghiệm phát triển hiện đại như lập trình reactive và widget composition trong khi sử dụng nền tảng Dart làm cơ sở chính cho các hoạt động đó. Nhóm Flutter đã đánh giá nhiều ngôn ngữ khác nhau và cuối cùng họ chọn Dart vì nó phù hợp với cách mà họ xây dựng giao diện người dùng.

### 1.2.Ưu điểm của Dart

**Ưu điểm của Dart :**

- **Dễ học và sử dụng**: Dart có cú pháp đơn giản và dễ hiểu, giúp cho việc học và sử dụng ngôn ngữ này trở nên dễ dàng, đặc biệt là cho các nhà phát triển mới.
- **Tính nhất quán**

- **Flutter framework:** Dart được sử dụng làm ngôn ngữ chính cho Flutter - một framework phát triển ứng dụng di động giao diện người dùng đa nền tảng. Flutter cung cấp một cách tiếp cận hiệu quả để phát triển ứng dụng di động với hiệu suất cao và giao diện người dùng đẹp mắt.
- **Hiệu suất cao**
- **Hỗ trợ công cụ:** Dart đi kèm với một loạt các công cụ hỗ trợ phát triển như trình biên dịch Dart, trình quản lý gói Pub và môi trường phát triển tích hợp như IntelliJ IDEA và Visual Studio Code.

Tóm lại, Dart là một ngôn ngữ lập trình đa năng và mạnh mẽ, đặc biệt là khi được sử dụng trong việc phát triển ứng dụng di động với Flutter. Tuy nhiên, như bất kỳ công nghệ nào khác, nó cũng có nhược điểm và yêu cầu sự hiểu biết và kinh nghiệm để sử dụng một cách hiệu quả.

## 2.Khái niệm Flutter

Flutter là một framework phát triển ứng dụng di động (mobile) và được phát hành bởi Google. Nó cho phép các nhà phát triển xây dựng ứng dụng di động chất lượng, đẹp mắt, nhanh chóng và hiệu quả bằng cách sử dụng một mã nguồn duy nhất. Flutter hỗ trợ viết mã một lần và triển khai trên nhiều nền tảng, bao gồm cả Android và iOS.

Flutter sử dụng ngôn ngữ lập trình Dart, một ngôn ngữ có hiệu suất cao, đáng tin cậy và dễ học. Một trong những đặc điểm nổi bật của Flutter là giao diện người dùng được xây dựng bằng cách kết hợp các thành phần UI tùy chỉnh. Nhờ đó, Flutter tạo ra giao diện người dùng đẹp mắt .

Flutter cung cấp một loạt các công cụ, thư viện và hỗ trợ phong phú, giúp nhà phát triển xây dựng các ứng dụng phức tạp và tùy chỉnh một cách dễ dàng. Đối với ứng dụng di động, Flutter cung cấp Hot Reload, cho phép nhà phát triển ngay lập tức thấy được sự thay đổi trong ứng dụng mà không cần phải khởi động lại toàn bộ ứng dụng. Điều này giúp tăng tốc quá trình phát triển và thử nghiệm.

### 2.1.Thành phần Flutter

Flutter bao gồm 2 thành phần quan trọng:

- **SDK (Software Development Kit):** Flutter là một SDK - bộ công cụ phát triển phần mềm - chứa các công cụ cần thiết để phát triển ứng dụng di động. Flutter

SDK bao gồm môi trường phát triển, trình biên dịch, thư viện, thư viện hỗ trợ và các công cụ khác giúp bạn viết, kiểm tra và triển khai ứng dụng của mình.

- **Framework (UI Library based on widgets):** Flutter là một framework UI (User Interface) xây dựng dựa trên widget. Widget trong Flutter là các thành phần cơ bản để xây dựng giao diện người dùng. Mọi thứ trong Flutter đều là widget, từ các thành phần đơn giản như văn bản, nút, hình ảnh đến các thành phần phức tạp hơn như danh sách cuộn và màn hình.

## 2.2.Đặc điểm của ngôn ngữ lập trình Flutter

Một số đặc điểm chính giúp Flutter trở nên hấp dẫn và phổ biến trong lĩnh vực phát triển ứng dụng di động hiện nay:

- **Hot Reload – JIT (Just-in-time)** : Flutter sở hữu một tính năng mạnh mẽ được tích hợp sâu bên trong kiến trúc của nó, được gọi là Hot Reload. Tính năng này có khả năng giúp bạn xem các bản cập nhật trong thời gian thực sau mỗi lần sửa đổi mã nguồn. Với Hot Reload, lập trình viên Flutter có thể sửa lỗi ngay lập tức tại nơi bị lỗi mà không cần khởi động lại toàn bộ ứng dụng, tiết kiệm thời gian đáng kể khi kiểm thử mã nguồn (test) và tìm lỗi (debug).
- **AOT – Ahead-of-time Compiler** : Khi đóng gói sản phẩm, Dart sẽ sử dụng trình biên dịch mang tên AOT (Ahead-Of-Time) để sinh ra các tệp thực thi cực nhanh. Điều này cho phép các tệp biên dịch AOT có hiệu năng cao hơn khi chạy so với JIT. AOT giúp Flutter vừa hỗ trợ tính năng Hot Reload khi phát triển, vừa mang đến hiệu năng tốt như các ngôn ngữ kiểu static.
- **Expressive and Flexible UI** : Trước đây, khi xây dựng giao diện ứng dụng di động với các yếu tố như animations là điều rất khó khăn đối với hầu hết các lập trình viên. Ngược lại, với Flutter, việc tạo ra giao diện ứng dụng lôi cuốn là điều dễ dàng và thú vị hơn. Với Flutter, giao diện được xây dựng bởi các widget, những widget này có thể lắp ráp với nhau một cách dễ dàng để tạo nên giao diện phức tạp hơn.

## 2.3.Các tính năng của Flutter

Các tính năng giúp cho Flutter trở thành một lựa chọn tốt để phát triển ứng dụng di động. Dưới đây là các tính năng đáng chú ý của Flutter, bao gồm:

- Flutter sử dụng ngôn ngữ Dart, một ngôn ngữ lập trình đơn giản, dễ đọc và dễ sử dụng. Dart được thiết kế để hỗ trợ Flutter và cung cấp nhiều tính năng hữu ích cho việc phát triển ứng dụng di động và web.

- Tính năng Hot Reload trong Flutter cho phép bạn ngay lập tức thấy được những thay đổi trong mã nguồn của ứng dụng trên môi trường phát triển. Điều này cho phép bạn dễ dàng xây dựng giao diện, sửa lỗi và trải nghiệm ứng dụng nhanh chóng.
- Flutter giải quyết những thách thức khó khăn trong giao diện người dùng bằng cách tập hợp các layout, platform và widget phong phú, giúp bạn xây dựng giao diện một cách dễ dàng và hiệu quả.
- Kiến trúc dựa trên widget trong Flutter giúp xây dựng giao diện người dùng một cách dễ dàng, linh hoạt và đẹp mắt. Các widget built-in và tự tạo giúp giao diện hoạt động phong phú, scroll mượt mà, linh hoạt và tự nhiên trên nhiều nền tảng.
- Flutter cung cấp một loạt các widget built-in giúp bạn tạo giao diện người dùng đẹp và chức năng phong phú. Các widget này cung cấp khả năng tự nhận thức nền tảng, giúp ứng dụng có cùng một giao diện trên nhiều nền tảng.

## 2.4.Ưu điểm của Flutter

**Ưu điểm :**

- **Sử dụng ngôn ngữ dễ đọc**: Flutter sử dụng ngôn ngữ lập trình Dart, một ngôn ngữ dễ đọc, đơn giản và mạnh mẽ. Dart giúp tạo mã nguồn dễ hiểu và dễ bảo trì.
- **Mã nguồn duy nhất, ứng dụng đa nền tảng**: Flutter cho phép viết mã một lần và triển khai ứng dụng trên nhiều nền tảng khác nhau như Android và iOS. Điều này giúp tiết kiệm công sức và thời gian trong việc phát triển ứng dụng đa nền tảng.
- **Hot Reload**: Tính năng Hot Reload cho phép nhà phát triển thấy các thay đổi trong mã nguồn của ứng dụng trên môi trường phát triển. Điều này giúp tăng tốc quá trình phát triển và thử nghiệm ứng dụng, cho phép bạn dễ dàng xây dựng giao diện, sửa lỗi và trải nghiệm ứng dụng nhanh chóng.
- **Xây dựng giao diện người dùng đẹp mắt**: Flutter sử dụng kiến trúc dựa trên widget, mọi thứ trong Flutter đều là widget. Kiến trúc này giúp xây dựng giao diện người dùng một cách dễ dàng, linh hoạt và đẹp mắt. Các widget built-in và tự tạo giúp giao diện hoạt động phong phú, scroll mượt mà và tự nhiên.
- **Cộng đồng hỗ trợ mạnh mẽ**: Flutter có một cộng đồng lớn và nhiệt tình, điều này đảm bảo bạn nhận được sự hỗ trợ và tài nguyên từ cộng đồng khi gặp khó khăn trong quá trình phát triển.

- **Mã nguồn mở**: Flutter là mã nguồn mở, cho phép bạn truy cập vào mã nguồn, đóng góp và tùy chỉnh theo yêu cầu của bạn.
- **Tốc độ cao**: Flutter sử dụng ngôn ngữ lập trình Dart, được biên dịch thành mã nguồn gốc, giúp tạo ra các ứng dụng nhanh chóng và khả năng phản hồi nhanh.
- **Không tốn chi phí**
- **Hỗ trợ từ Google**
- **Khắc phục sự cố đơn giản với các công cụ hỗ trợ hiệu quả**
- **Kiểm định chất lượng đơn giản với các công cụ kiểm thử tự động**
- **Linh hoạt với mọi kích thước màn hình trên thiết bị di động**

## 3.Provider Trong Flutter

### 3.1.Khái niệm Provider

Trong Flutter, "Provider" là một gói (package) được sử dụng để quản lý trạng thái (state management) và cung cấp dữ liệu (data) cho các widget trong ứng dụng. Provider giúp giải quyết vấn đề chia sẻ và cập nhật dữ liệu giữa các widget một cách hiệu quả, giúp tạo ra các ứng dụng có cấu trúc tốt và dễ bảo trì.

### 3.2.Đặc điểm của Provider

Là một mô hình quản lý trạng thái: Provider được sử dụng để quản lý trạng thái của ứng dụng và cung cấp dữ liệu cho các widget. Nó giúp tách biệt logic trạng thái ra khỏi giao diện người dùng.

Cung cấp dữ liệu tới các widget con: Provider cho phép các widget con có thể truy cập và sử dụng dữ liệu một cách thuận tiện và an toàn.

Cơ chế cập nhật tự động: Khi dữ liệu được cập nhật trong Provider, các widget được liên kết sẽ tự động cập nhật lại giao diện để phản ánh các thay đổi mới.

Khả năng giảm thiểu rebuild không cần thiết: Provider hỗ trợ cập nhật chỉ những widget cần thiết khi dữ liệu thay đổi, giúp giảm thiểu các quá trình rebuild không cần thiết và tối ưu hóa hiệu suất ứng dụng.

Dễ dàng tích hợp với các cách tiếp cận khác: Provider có thể được kết hợp với các gói khác trong Flutter như Provider.of, Consumer, Selector để cung cấp các cơ chế linh hoạt trong quản lý trạng thái.

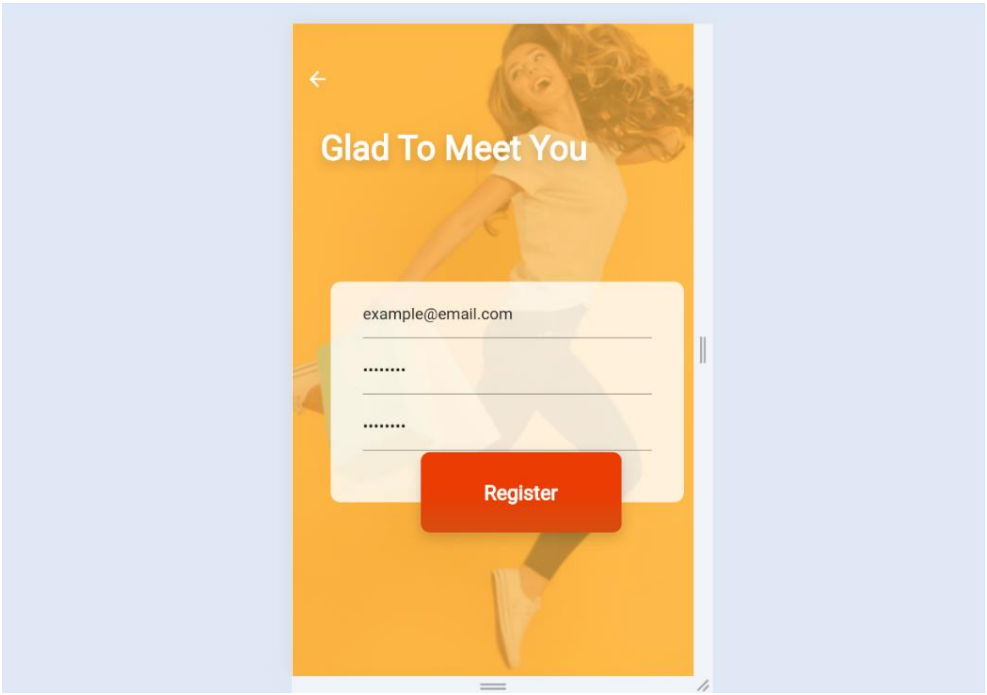
## 3.3.Ưu điểm của Provider

Ưu điểm của Provider gồm có :

- Dễ sử dụng: Provider là một gói đơn giản và linh hoạt để sử dụng. Nó không đòi hỏi nhiều cấu hình phức tạp và dễ dàng tích hợp vào các dự án Flutter hiện có.
- Tối ưu hóa tái sử dụng: Provider cho phép bạn tổ chức dữ liệu và logic ứng dụng thành các "provider" riêng biệt, giúp tăng tính tái sử dụng của mã và tách biệt logic ứng dụng.
- Khả năng linh hoạt và mở rộng: Provider hỗ trợ nhiều cách tiếp cận khác nhau để quản lý trạng thái, bao gồm ChangeNotifierProvider, StreamProvider, FutureProvider, và ValueListenableProvider, phù hợp với nhiều loại ứng dụng và tình huống sử dụng khác nhau.
- Tích hợp tốt với Flutter: Provider là một trong những gói phổ biến nhất cho Flutter State Management và được hỗ trợ rộng rãi trong cộng đồng Flutter. Nó liên tục nhận được cập nhật và sự phát triển từ cộng đồng để cải thiện hiệu suất và trải nghiệm phát triển.

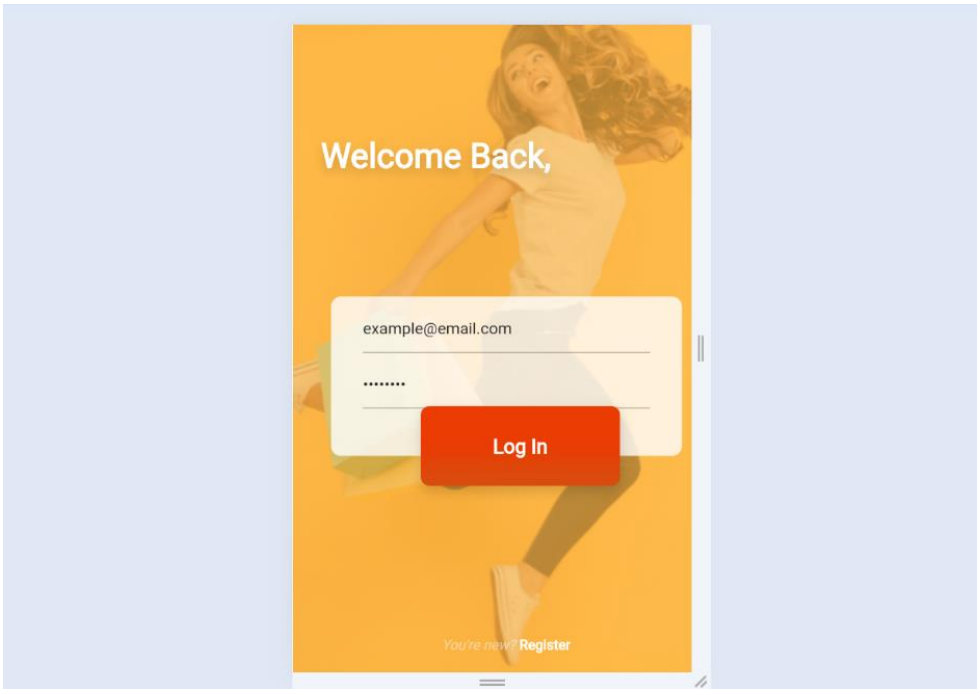
# Chương 2.THIẾT KẾ GIAO DIỆN

## 1.Giao diện đăng ký



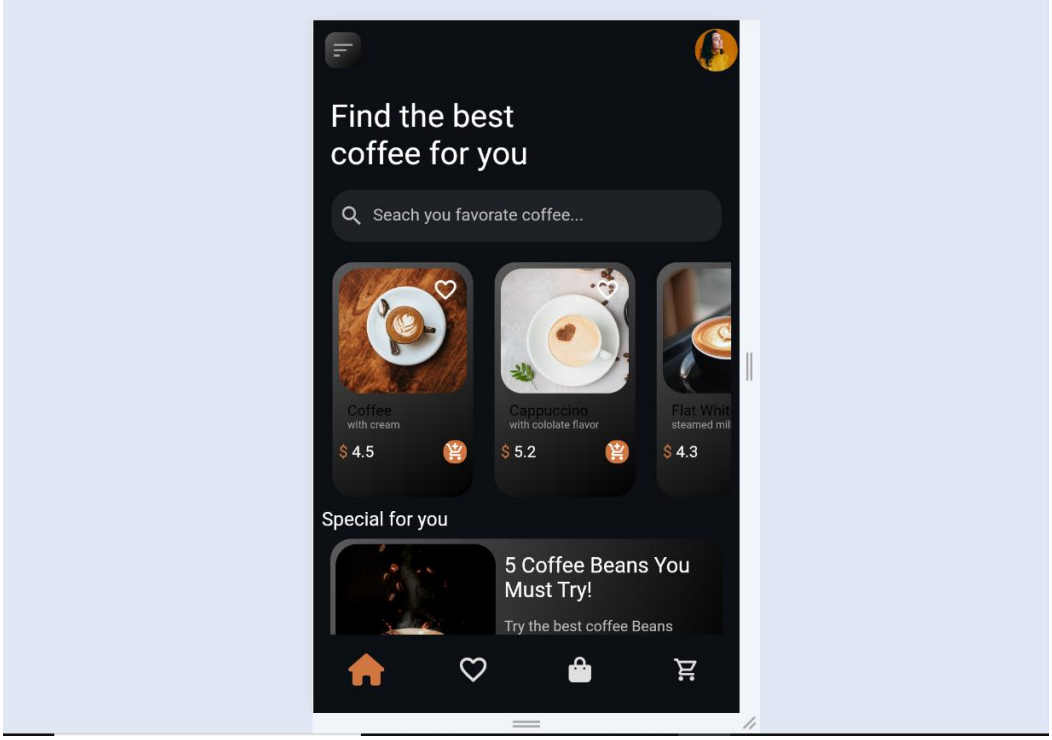

*Hình 1 Giao diện đăng ký*

## 2.Giao diện đăng nhập



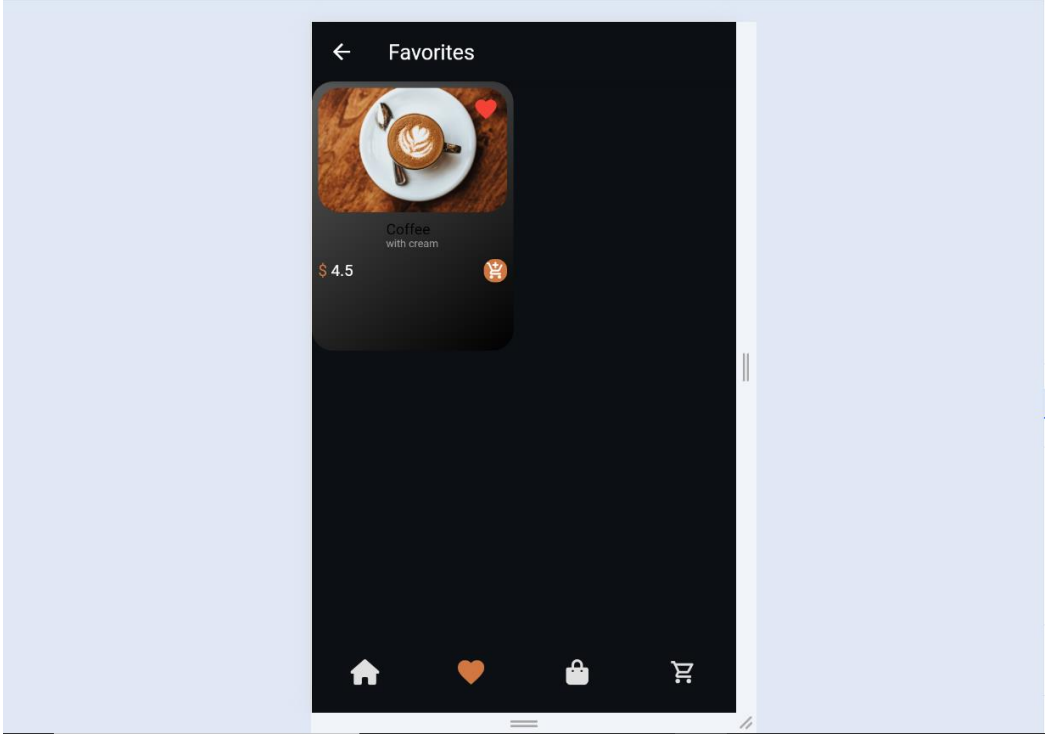

*Hình 2. Giao diện đăng nhập*

## 3.Giao diện chính



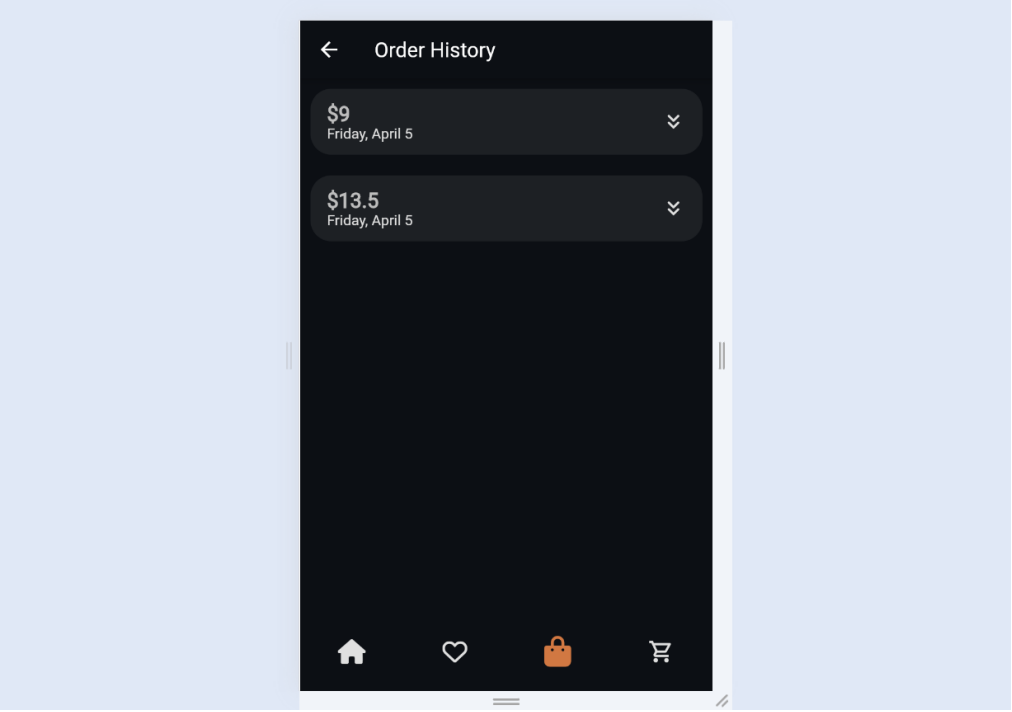

*Hình 3. Giao diện chính*

## 4.Giao diện yêu thích sản phẩm



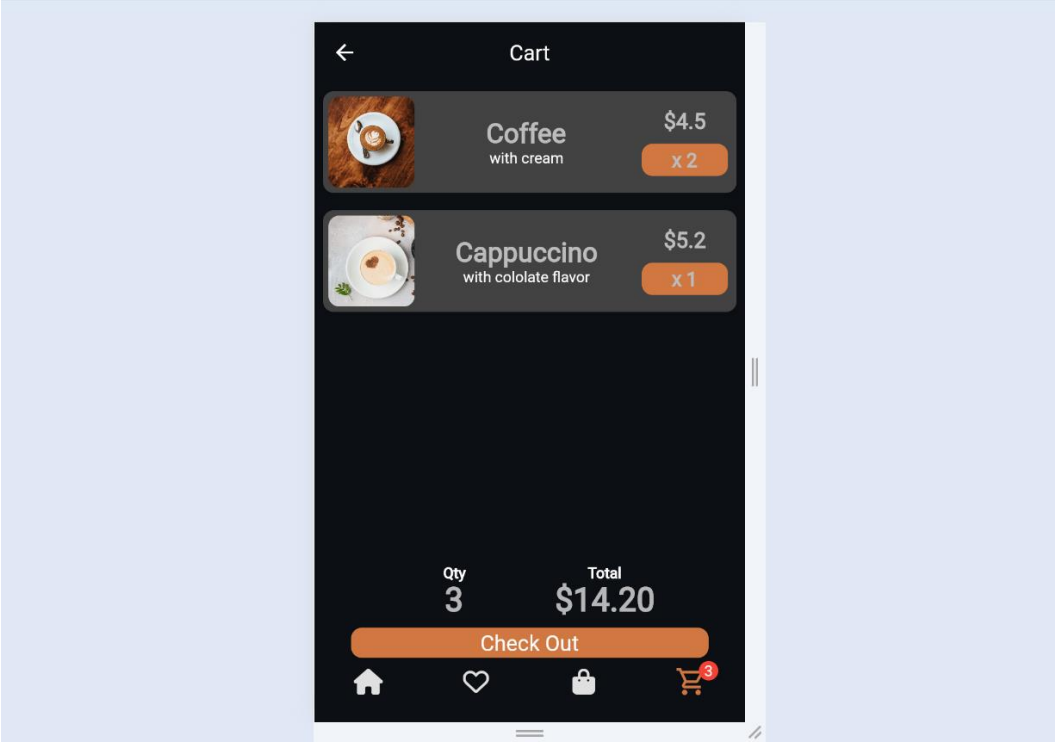

*Hình 4. Giao diện yêu thích sản phẩm*

## 5.Giao diện lịch sử mua hàng



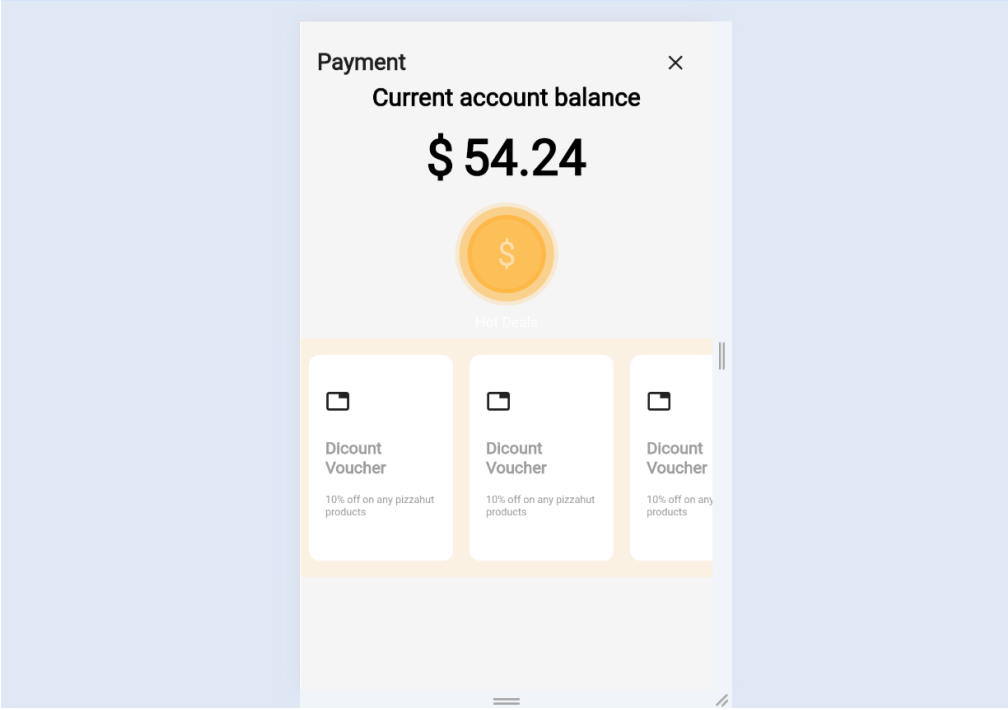

*Hình 5 Giao diện lịch sử mua hàng*

## 6.Giao diện giỏ hàng



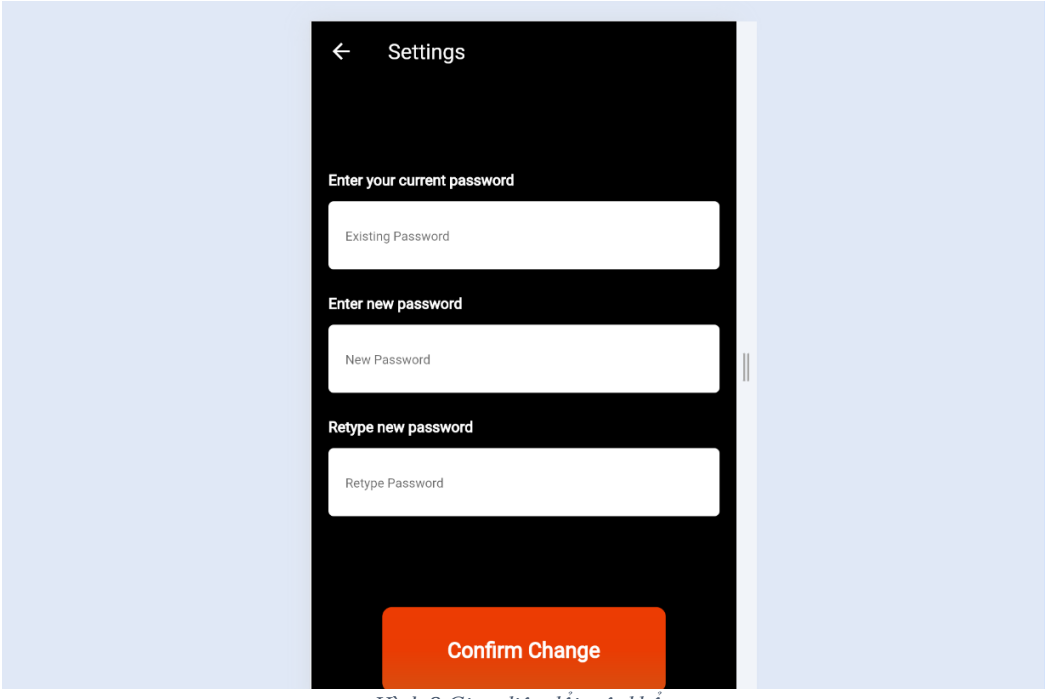

*Hình 6 Giao diện giỏ hàng*

## 7.Giao diện ví tài khoản




*Hình 7 Giao diện ví tài khoản*

## 8.Giao diện Đổi mật khẩu




*Hình 8 Giao diện đổi mật khẩu*

# Chương 3.XÂY DỰNG HỆ THỐNG

## 1.Code quản lý state bằng Change Notifier

```dart
import 'package:cofeeshop/provider/coffee.dart';
import 'package:flutter/material.dart';

class CofeeData extends ChangeNotifier {
  final List<Coffee> _coffeeList = [];

  List<Coffee> get coffeeList {
    return [..._coffeeList];
  }

  set coffeeList(List<Coffee> value){
    _coffeeList.clear();
    for(Coffee cf in value){
      _coffeeList.add(cf);
    }
    notifyListeners();
  }

  List<Coffee> get favoriteItems {
    return _coffeeList.where((item) => item.isFavarate).toList();
```

*Hình 9 Ảnh code minh họa*

```dart
    for(var order in orders){
      _orders.insert(0, order);
    }
    notifyListeners();
  }

  void addOrders(List<CartItems> cartProducts, double total) {
    List<CartItem> products = cartProducts
        .map((cartItem) => CartItem(
              id: cartItem.id,
              title: cartItem.title,
              image: cartItem.image,
              amount: cartItem.amount,
              description: cartItem.descritoin,
              qunatity: cartItem.quantity,
              prodcutid: '',
            ))
        .toList();

    _orders.insert(
```

*Hình 10 Ảnh code minh họa*

```
String get key{
  return userSession.jwtKey;
}

set key(String value){
  userSession.jwtKey = value;
}

void login(String jwtKey, String name, String imagePath){
  userSession.jwtKey = jwtKey;
  userSession.imagePath = imagePath;
  userSession.name = name;
  notifyListeners();
}

void signOut(){
  userSession.clear();
  notifyListeners();
}
```

*Hình 11 Ảnh code minh họa*

Đoạn mã trên Lớp CofeeData là một provider trong Flutter để quản lý danh sách sản phẩm cà phê và các thao tác liên quan đến dữ liệu. Các thành phần chính của lớp này là:

Danh sách _coffeeList: Là danh sách chứa các đối tượng Coffee (sản phẩm cà phê).

Getter coffeeList và Setter coffeeList:

Getter coffeeList trả về một bản sao của _coffeeList, giúp đảm bảo tính bất biến của dữ liệu.

Setter coffeeList cập nhật danh sách _coffeeList bằng danh sách mới được cung cấp (value) và sau đó thông báo cho các thành phần khác biết về sự thay đổi thông qua notifyListeners().

Getter favoriteItems: Trả về danh sách các sản phẩm cà phê được đánh dấu là yêu thích từ _coffeeList.

Phương thức findByid và findById:

findByid: Tìm kiếm và trả về một sản phẩm cà phê từ _coffeeList dựa trên id.

findById: Tương tự như findByid nhưng tìm kiếm trong bản sao coffeeList để đảm bảo tính bất biến.

Phương thức addProduct và refreshUI:

addProduct: Không làm gì ngoài việc thông báo cho các thành phần khác biết về sự thay đổi trong dữ liệu.

refreshUI: Gọi notifyListeners() để thông báo cho các thành phần khác cần làm mới giao diện người dùng.

Lớp CofeeData sử dụng ChangeNotifier để cung cấp tính năng cập nhật dữ liệu và thông báo sự thay đổi đến các thành phần giao diện liên quan trong ứng dụng Flutter.

## 2.Code login , register

```
51    onTap: () async {
52      String usr = email.value.text;
53      String pwd = password.value.text;
54      String name = fullName.value.text;
55
56      bool isRegistered = await Service.gI().register(usr, pwd, name);
57      if(isRegistered){
58        ScaffoldMessenger.of(context).showSnackBar(
59          const SnackBar(
60            content: Text('Đăng ký thành công !'),
61            duration: Duration(seconds: 2),
62          ),  // SnackBar
63        );
64      }
65      else{
66        ScaffoldMessenger.of(context).showSnackBar(
67          const SnackBar(
68            content: Text('Đăng ký thất bại !'),
69            duration: Duration(seconds: 2),
70          ),  // SnackBar
```

*Hình 12 Ảnh code minh họa*

```
52    onTap: () async {
53      String usr = username.value.text;
54      String pwd = password.value.text;
55      bool isSuccess = await Service.gI().checkLogin(usr, pwd, user) ;
56      if(isSuccess) {
57        Service.gI().getCoffees(coffeeData);
58        Service.gI().getUserCart(user, cart);
59        Service.gI().getUserOrder(user, orders);
60
61        Navigator.of(context)
62            .push(MaterialPageRoute(builder: (_) => const PageNavigation()));
63      }
64      else{
65        ScaffoldMessenger.of(context).showSnackBar(
66          const SnackBar(
67            content: Text('Đăng nhập thất bại. Vui lòng thử lại.'),
68            duration: Duration(seconds: 2), // Thời gian hiển thị thông báo
69          ),  // SnackBar
```

*Hình 13 Ảnh code minh họa*

Đoạn mã trên là một hàm callback onTap trong Flutter được sử dụng khi người dùng nhấn vào một widget (ví dụ: nút đăng nhập). Hãy đi qua từng phần của mã để hiểu cách hoạt động:

Lấy giá trị từ các trường nhập liệu:

String usr = username.value.text;

String pwd = password.value.text;

username.value.text và password.value.text lấy giá trị mà người dùng nhập vào các trường tương ứng. Đây giả định là các trường nhập liệu (ví dụ: TextField) trong giao diện người dùng.

Gọi phương thức kiểm tra đăng nhập từ service:

bool isSuccess = await Service.gI().checkLogin(usr, pwd, user);

Service.gI().checkLogin(usr, pwd, user) được gọi để kiểm tra đăng nhập với tên đăng nhập (usr), mật khẩu (pwd), và đối tượng người dùng (user). Phương thức này trả về true nếu đăng nhập thành công và false nếu không.

Xử lý kết quả đăng nhập:

```
if (isSuccess) {
  // Nếu đăng nhập thành công, thực hiện các hành động sau:
  Service.gI().getCoffees(coffeeData);   // Lấy danh sách sản phẩm cà phê từ service
và gán vào biến coffeeData
  Service.gI().getUserCart(user, cart);   // Lấy giỏ hàng của người dùng từ service
và gán vào biến cart
  Service.gI().getUserOrder(user, orders); // Lấy đơn hàng của người dùng từ service
và gán vào biến orders

  // Điều hướng đến màn hình `PageNavigation`
  Navigator.of(context).push(MaterialPageRoute(builder: (_) => const PageNavigation
()));
} else {
  // Nếu đăng nhập không thành công, hiển thị thông báo lỗi
  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(
      content: Text('Đăng nhập thất bại. Vui lòng thử lại.'),
      duration: Duration(seconds: 2), // Thời gian hiển thị thông báo
    ),
  );
}
```

*Hình 14 Ảnh code minh họa*

Nếu đăng nhập thành công (isSuccess == true), thực hiện các hành động sau:

Gọi các phương thức từ service để lấy danh sách sản phẩm cà phê, giỏ hàng và đơn hàng của người dùng và gán vào các biến tương ứng (coffeeData, cart, orders).

Tiếp đó, điều hướng đến màn hình PageNavigation bằng cách sử dụng Navigator.

Nếu đăng nhập không thành công (isSuccess == false), hiển thị một SnackBar thông báo lỗi trong giao diện người dùng với nội dung "Đăng nhập thất bại. Vui lòng thử lại." và thời gian hiển thị là 2 giây.

## 3.Code xử lý backend

```
    def get_user(self, username: str, password: str) -> User:
        if validate_string(username) == False or validate_string(password) == False:
            return None
        query = f"SELECT * FROM User WHERE username = '{username}' AND password = '{password}'"
        result = self.sql.execute_query(query)
        if len(result) > 0:
            return User(result[0])
        return None

    def add_user(self, username: str, password: str, fullname: str) -> bool:
        if validate_string(username) == False or validate_string(password) == False:
            return False
        if self.get_user(username, password) != None:
            return False
        try:
            query = f"INSERT INTO User(username, password, imagePath, name) VALUES('{username}', '{password}','http://localhost/guest.pr
            self.sql.execute_insert_query(query)
            return True
        except:
            return False
```

*Hình 15 Ảnh code minh họa*

```
def login():
    data = request.json
    username = data["username"]
    password = data["password"]
    user = service.check_login(username, password)[0]
    token = service.check_login(username, password)[1]

    if token != None:
        print("User: ", user)
        return jsonify({"isSuccess": True, "name": user.name, "image" : user.image ,"token": token})
    return jsonify({"error": "Invalid username or password"})

@app.route("/register", methods=["POST"])
def register():
    data = request.json
    username = data["username"]
    password = data["password"]
    fullname = data["fullName"]
    return jsonify({"isSuccess": service.addAccount(username, password, fullname)})


@app.route("/get_coffees", methods=["GET"])
def get_coffees():
    return jsonify(service.get_coffes())
```

*Hình 16 Ảnh code minh họa*

```python
    def __init__(self, data: tuple):
        self.id = data[0]
        self.id_user = data[1]
        self.amount = data[2]
        self.date_time = data[3]
        self.order_items = []

    def datetime_converter(self, o):
        if isinstance(o, datetime):
            return o.__str__()

    def to_dict(self):
        return {
            "id": self.id,
            "id_user": self.id_user,
            "amount": self.amount,
            "date_time": json.dumps(self.date_time, default= self.datetime_converter),
            "order_items": [order_item.to_dict() for order_item in self.order_items]
        }

    def add_order_item(self, order_item: CartItem):
        self.order_items.append(order_item)
```

*Hình 15 Ảnh code minh họa*

```python
class Security:
    def __init__(self):
        self.secret_key = 'coffeshopXYZ'
        pass

    # Generate JWT key
    def generate_jwt_key(self,id: int, username: str, image : str, name : str):
        claims = {
            "id" : id,
            "username": username,
            "image" : image,
            "name" : name,
            "exp": int(time.time() * 1000) +  120 * 60 * 1000 # Token expires in 120 minutes
        }

        jwt_token = jwt.encode(claims, self.secret_key, algorithm='HS256')
        return jwt_token


    def validate_jwt_key(self, token):
        try:
            # Verify and decode the token
            decoded_token = jwt.decode(token, self.secret_key, algorithms=['HS256'])
            if time.time() * 1000 > decoded_token['exp']:
                return None
            return decoded_token
        except:
            return None
```

*Hình 16 Ảnh code minh họa*

```python
    def check_jwt(self, token: str) -> bool:
        userInfo = sc.validate_jwt_key(token)
        return sc.validate_jwt_key(token) != None

    def addAccount(self, username: str, password: str, fullname: str) -> bool:
        return self.ds.add_user(username, password, fullname)

    #Product
    def get_coffes(self) -> list:
        query = "select * from Coffe"
        coffees = []
        for cf_data in self.ds.sql.execute_query(query):
            cf = Coffe(cf_data)
            coffees.append(cf.to_dict())
        return coffees

    def addCoffeToCart(self, id_user, id_coffe):
        query = f"insert into UserCart(id_user, id_coffe, quantity) values('{id_user}', '{id_coffe}', 1)"
        self.ds.sql.execute_insert_query(query)
        pass

    def updateQuantityCoffeInCart(self, id_user, id_coffe):
        query = f"update UserCart set quantity = quantity + 1 where id_user = '{id_user}' and id_coffe = '{id_coffe}'"
        self.ds.sql.execute_insert_query(query)
        pass
```

*Hình 17 Ảnh code minh họa*

**Datasource**

Đoạn mã Python trên định nghĩa hai lớp SqlConnector và DataSource để tương tác với cơ sở dữ liệu MySQL và thực hiện các thao tác liên quan đến người dùng.

Hàm validate_string(string: str) -> bool:

Hàm này được sử dụng để kiểm tra xem một chuỗi string chỉ chứa các ký tự chữ cái và số.

Phương thức string.isalnum() trả về True nếu chuỗi string chỉ chứa các ký tự chữ cái và số, ngược lại trả về False.

Lớp SqlConnector:

Lớp này đại diện cho kết nối đến cơ sở dữ liệu MySQL và thực hiện các thao tác truy vấn SQL.

Phương thức __init__:

Phương thức khởi tạo này thiết lập kết nối đến cơ sở dữ liệu MySQL sử dụng thông tin về host, user, password, và database được cung cấp.

Sau khi kết nối thành công, nó in ra thông báo "Connected to database".

Tiếp theo, nó tạo một đối tượng con trỏ (cursor) để thực hiện các truy vấn SQL.

Trong ví dụ này, sau khi khởi tạo SqlConnector, nó thực hiện một truy vấn SQL SELECT để lấy tất cả các dòng từ bảng User và in ra kết quả.

Phương thức execute_query(self, query: str) -> list:

Phương thức này được sử dụng để thực hiện một truy vấn SQL được cung cấp qua biến query.

Nó sử dụng phương thức self.cursor.execute(query) để thực hiện truy vấn.

Sau đó, nó sử dụng self.cursor.fetchall() để lấy tất cả các kết quả của truy vấn và trả về dưới dạng một danh sách các tuple.

Phương thức execute_insert_query(self, query: str):

Phương thức này được sử dụng để thực hiện một truy vấn SQL INSERT được cung cấp qua biến query.

Sau khi thực hiện truy vấn INSERT bằng cách gọi self.cursor.execute(query), nó gọi self.conn.commit() để xác nhận việc thêm dữ liệu vào cơ sở dữ liệu.

Phương thức stop(self):

Phương thức này được sử dụng để đóng các tài nguyên kết nối và con trỏ của cơ sở dữ liệu.

Nó gọi self.cursor.close() để đóng con trỏ và self.conn.close() để đóng kết nối.

Lớp DataSource:

Lớp này là một nguồn dữ liệu để thực hiện các thao tác liên quan đến người dùng trong cơ sở dữ liệu.

Phương thức __init__:

Trong hàm khởi tạo này, lớp DataSource khởi tạo một đối tượng SqlConnector để sử dụng cho các hoạt động truy vấn cơ sở dữ liệu.

Phương thức get_user(self, username: str, password: str) -> User:

Phương thức này được sử dụng để lấy thông tin của một người dùng từ cơ sở dữ liệu với username và password cung cấp.

Trước khi thực hiện truy vấn, nó kiểm tra tính hợp lệ của username và password bằng cách gọi hàm validate_string.

Sau đó, nó thực hiện một truy vấn SQL SELECT để tìm người dùng với username và password tương ứng.

Nếu tìm thấy người dùng, nó tạo một đối tượng User từ thông tin người dùng và trả về.

Nếu không tìm thấy người dùng, nó trả về None.

Phương thức add_user(self, username: str, password: str, fullname: str) -> bool:

Phương thức này được sử dụng để thêm một người dùng mới vào cơ sở dữ liệu với thông tin username, password, và fullname cung cấp.

Trước khi thêm người dùng mới, nó kiểm tra tính hợp lệ của username và password bằng cách gọi hàm validate_string và xem xét xem người dùng đã tồn tại chưa bằng cách gọi phương thức get_user.

Nếu thông tin người dùng hợp lệ và người dùng chưa tồn tại trong cơ sở dữ liệu, nó thực hiện truy vấn SQL INSERT để thêm người dùng mới và trả về True.

Nếu gặp lỗi trong quá trình thêm người dùng, nó trả về False.

**Main**

Đoạn mã trên là một ứng dụng web backend được xây dựng bằng Flask để cung cấp các API RESTful cho ứng dụng frontend. Hãy đi qua từng phần của mã để hiểu cách hoạt động:

Cài đặt và Import các thư viện:

Flask: Framework web để xây dựng ứng dụng backend.

jsonify: Để trả về dữ liệu JSON từ các endpoint API.

request: Để nhận và xử lý các yêu cầu HTTP gửi đến các endpoint API.

CORS: Để xử lý CORS (Cross-Origin Resource Sharing), cho phép truy cập từ các nguồn khác nhau.

Service: Được import từ Service module để tương tác với dịch vụ (service) cung cấp các chức năng xử lý logic của ứng dụng.

Security: Được import từ Security module để xử lý các vấn đề bảo mật như xác thực người dùng.

Khởi tạo ứng dụng Flask và cài đặt CORS:

app = Flask(__name__)

CORS(app)

app = Flask(__name__): Khởi tạo một đối tượng Flask để xây dựng ứng dụng web.

CORS(app): Kích hoạt CORS để cho phép các yêu cầu từ các nguồn khác nhau truy cập vào API.

Khởi tạo Service và Endpoint API:

service = Service()

service = Service(): Khởi tạo một đối tượng của lớp Service để sử dụng trong các endpoint API.

Các Endpoint API:

Endpoint /login (POST):

Endpoint này được sử dụng để xử lý yêu cầu đăng nhập.

Lấy dữ liệu từ yêu cầu JSON (username và password).

Gọi phương thức check_login của service để kiểm tra đăng nhập.

Trả về kết quả thành công hoặc thất bại dưới dạng JSON.

Endpoint /register (POST):

Endpoint này được sử dụng để xử lý yêu cầu đăng ký tài khoản mới.

Lấy dữ liệu từ yêu cầu JSON (username, password và fullName).

Gọi phương thức addAccount của service để thêm tài khoản mới.

Trả về kết quả thành công hoặc thất bại dưới dạng JSON.

Endpoint /get_coffees (GET):

Endpoint này được sử dụng để lấy danh sách các mặt hàng cà phê.

Gọi phương thức get_coffees của service để lấy danh sách cà phê.

Trả về danh sách các mặt hàng cà phê dưới dạng JSON.

Endpoint /get_user_cart (POST):

Endpoint này được sử dụng để lấy giỏ hàng của người dùng.

Lấy jwtKey từ yêu cầu JSON để xác thực người dùng.

Gọi phương thức getUserCart của service để lấy giỏ hàng của người dùng.

Trả về thông tin giỏ hàng dưới dạng JSON.

Endpoint /get_user_order (POST):

Endpoint này được sử dụng để lấy đơn hàng của người dùng.

Lấy jwtKey từ yêu cầu JSON để xác thực người dùng.

Gọi phương thức getUserOrder của service để lấy đơn hàng của người dùng.

Trả về thông tin đơn hàng dưới dạng JSON.

Khởi chạy ứng dụng Flask:

```
if __name__ == '__main__':
    app.run(debug=True, port=12121)
```

Dòng này kiểm tra xem script được chạy trực tiếp hay là được import từ một module khác.

Nếu được chạy trực tiếp, Flask sẽ khởi chạy ứng dụng với chế độ debug và trên cổng 12121.

**Models**

Mã trên định nghĩa một số lớp trong Python để đại diện cho các đối tượng trong hệ thống, bao gồm User, Coffee, CartItem, Order, và OrderItem. Dưới đây là các giải thích chi tiết về mỗi lớp:

Lớp User:

Lớp này đại diện cho đối tượng người dùng trong hệ thống.

Thuộc tính:

id: ID của người dùng.

username: Tên đăng nhập của người dùng.

password: Mật khẩu của người dùng.

image: Đường dẫn đến hình ảnh của người dùng.

name: Tên của người dùng.

Phương thức:

generate_jwt(): Phương thức này được sử dụng để tạo mã JWT (JSON Web Token) dựa trên id và username của người dùng bằng cách gọi phương thức generate_jwt_key từ đối tượng sc (được khởi tạo từ lớp Security).

Lớp Coffee:

Lớp này đại diện cho đối tượng sản phẩm cà phê trong hệ thống.

Thuộc tính:

id: ID của sản phẩm cà phê.

title: Tiêu đề của sản phẩm.

image: Đường dẫn đến hình ảnh của sản phẩm.

description: Mô tả của sản phẩm.

amount: Số lượng sản phẩm.

Phương thức:

to_dict(): Phương thức này chuyển đổi đối tượng Coffee thành một từ điển (dictionary) để dễ dàng serialize thành JSON.

Lớp CartItem:

Lớp này đại diện cho một mục trong giỏ hàng của người dùng.

Thuộc tính:

id_user: ID của người dùng.

id_coffee: ID của sản phẩm cà phê.

quantity: Số lượng của sản phẩm trong giỏ hàng.

coffee: Đối tượng Coffee tương ứng với sản phẩm trong giỏ hàng.

Phương thức:

to_dict(): Phương thức này chuyển đổi đối tượng CartItem thành một từ điển (dictionary) để dễ dàng serialize thành JSON.

Lớp Order:

Lớp này đại diện cho đối tượng đơn hàng trong hệ thống.

Thuộc tính:

id: ID của đơn hàng.

id_user: ID của người dùng.

amount: Tổng số tiền của đơn hàng.

date_time: Thời điểm đặt đơn hàng (dưới dạng đối tượng datetime).

order_items: Danh sách các mục trong đơn hàng (OrderItem).

Phương thức:

datetime_converter(o): Phương thức này được sử dụng để chuyển đổi đối tượng datetime thành một định dạng chuỗi.

to_dict(): Phương thức này chuyển đổi đối tượng Order thành một từ điển (dictionary) để dễ dàng serialize thành JSON.

add_order_item(order_item): Phương thức này được sử dụng để thêm một mục vào đơn hàng (order_items).

Lớp OrderItem:

Lớp này đại diện cho một mục trong đơn hàng.

id_order: ID của đơn hàng.

id_coffee: ID của sản phẩm cà phê.

quantity: Số lượng sản phẩm trong đơn hàng.

price: Giá của sản phẩm.

coffee: Đối tượng Coffee tương ứng với sản phẩm trong đơn hàng.

to_dict(): Phương thức này chuyển đổi đối tượng OrderItem thành một từ điển (dictionary) để dễ dàng serialize thành JSON.

**Security**

Đoạn mã trên định nghĩa một lớp Security trong Python để xử lý việc tạo và xác thực JSON Web Token (JWT). Dưới đây là giải thích chi tiết về từng phần của mã:

Lớp Security:

Lớp này chứa các phương thức để tạo và xác thực JWT (JSON Web Token).

Phương thức __init__(self):

Phương thức khởi tạo của lớp Security.

Nó khởi tạo secret_key là một chuỗi đại diện cho khóa bí mật được sử dụng để ký và giải mã JWT.

Trong trường hợp này, secret_key được đặt là 'coffeshopXYZ'.

Phương thức generate_jwt_key(self, id: int, username: str, image: str, name: str) -> str:

Phương thức này được sử dụng để tạo JWT với các claims (tuyên bố) chứa các thông tin của người dùng.

Tham số:

id: ID của người dùng.

username: Tên đăng nhập của người dùng.

image: Đường dẫn đến hình ảnh của người dùng.

name: Tên của người dùng.

Các bước:

Tạo claims là một từ điển chứa các thông tin của người dùng và thời gian hết hạn của token.

Trong claims, exp (expiration time) được tính toán là thời điểm hiện tại cộng với 120 phút (2 giờ) tính bằng millisecond, là thời gian hết hạn của token.

Sử dụng thư viện jwt để mã hóa claims thành JWT bằng cách sử dụng thuật toán HS256 (HMAC with SHA-256) và secret_key.

Trả về JWT đã mã hóa dưới dạng chuỗi.

Phương thức validate_jwt_key(self, token: str) -> dict:

Phương thức này được sử dụng để xác thực JWT đã mã hóa.

token: Chuỗi JWT cần xác thực.

Sử dụng thư viện jwt để giải mã và xác thực token bằng cách sử dụng secret_key và thuật toán HS256.

Nếu token hợp lệ và chưa hết hạn (exp chưa qua thời gian hiện tại), phương thức trả về các tuyên bố (claims) của token dưới dạng một từ điển.

Nếu token không hợp lệ hoặc đã hết hạn, phương thức trả về None.

Lưu ý rằng nếu xảy ra bất kỳ lỗi nào trong quá trình giải mã hoặc xác thực, phương thức cũng trả về None.

**Service**

Đoạn mã trên định nghĩa một lớp Service trong Python để cung cấp các chức năng cho ứng dụng của bạn, bao gồm xác thực người dùng, quản lý sản phẩm và giỏ hàng, và truy xuất dữ liệu người dùng và đơn hàng từ cơ sở dữ liệu. Dưới đây là giải thích chi tiết về từng phần của mã:

Lớp Service:

Lớp này chứa các phương thức để thực hiện các hoạt động liên quan đến xác thực, sản phẩm và người dùng.

Phương thức __init__(self):

Phương thức khởi tạo của lớp Service.

Nó khởi tạo một đối tượng DataSource để truy xuất dữ liệu từ cơ sở dữ liệu.

Phương thức check_login(self, username: str, password: str) -> list:

Phương thức này được sử dụng để xác thực người dùng dựa trên tên đăng nhập và mật khẩu.

Nó sử dụng DataSource để lấy thông tin người dùng từ cơ sở dữ liệu.

Nếu người dùng tồn tại, nó trả về một danh sách gồm người dùng và mã JWT được tạo dựa trên thông tin của người dùng.

Nếu không tìm thấy người dùng, nó trả về danh sách rỗng.

Phương thức check_jwt(self, token: str) -> bool:

Phương thức này được sử dụng để xác thực mã JWT.

Nó sử dụng đối tượng Security để xác thực mã JWT dựa trên token.

Nếu mã JWT hợp lệ, nó trả về True, ngược lại trả về False.

Phương thức addAccount(self, username: str, password: str, fullname: str) -> bool:

Phương thức này được sử dụng để thêm tài khoản người dùng mới vào cơ sở dữ liệu.

Nó sử dụng DataSource để thực hiện thêm người dùng mới.

Nếu thêm thành công, nó trả về True, ngược lại trả về False.

Phương thức get_coffes(self) -> list:

Phương thức này được sử dụng để lấy danh sách các sản phẩm cà phê từ cơ sở dữ liệu.

Nó thực hiện một truy vấn SQL để lấy thông tin của các sản phẩm cà phê.

Sau đó, nó tạo các đối tượng Coffee từ dữ liệu và chuyển chúng thành danh sách từ điển để trả về.

Các phương thức khác:

addCoffeToCart(self, id_user, id_coffe), updateQuantityCoffeInCart(self, id_user, id_coffe), removeCoffeFromCart(self, id_user, id_coffe): Các phương thức này được sử dụng để thêm, cập nhật hoặc xóa sản phẩm từ giỏ hàng của người dùng trong cơ sở dữ liệu.

getUserCart(self, jwtKey: str) -> list: Phương thức này được sử dụng để lấy thông tin giỏ hàng của người dùng dựa trên mã JWT.

getUserOrder(self, jwtKey: str) -> list: Phương thức này được sử dụng để lấy thông tin đơn hàng của người dùng dựa trên mã JWT, bao gồm cả các mục trong đơn hàng.

# Chương 4.Ý TƯỞNG ỨNG DỤNG, PHÁT TRIỂN

Việc phát triển ứng dụng bán cafe trên Android Studio là một ý tưởng thú vị và tiềm năng, nhưng đòi hỏi bạn cần xem xét nhiều yếu tố để có một ứng dụng hiệu quả và thu hút người dùng. Dưới đây là một bài luận vắn tắt về ý tưởng và phát triển ứng dụng bán cafe trên nền tảng Android:

## 1.Ý tưởng ứng dụng

**Mục tiêu chính:**

- Tạo ra một ứng dụng di động trực quan và dễ sử dụng cho người dùng để đặt đồ uống café trước.
- Cung cấp trải nghiệm đơn giản và thuận tiện để lựa chọn các loại đồ uống và đặt hàng từ xa.

**Tính năng chính:**

- Đăng nhập và đăng ký: Người dùng có thể đăng nhập bằng tài khoản của họ hoặc đăng ký tài khoản mới.
- Danh mục sản phẩm: Hiển thị danh sách các loại đồ uống (cà phê, trà, nước ép, kem, vv.) với hình ảnh và mô tả.
- Giỏ hàng: Cho phép người dùng xem lại các mặt hàng đã chọn và chỉnh sửa số lượng.
- Thanh toán: Cung cấp nhiều phương thức thanh toán an toàn và thuận tiện.
- Quản lý đơn hàng: Người dùng có thể xem lịch sử đơn hàng và tình trạng đơn hàng.
- Ví tiền : Người dùng có thể xem trong ví tiền điện tử có bao nhiêu tiền trong ứng dụng.
- Thay đổi mật khẩu : Người dùng có thể thay đổi mật khẩu .

**Tiềm năng mở rộng :**

- Tích hợp tính năng định vị để tìm kiếm cửa hàng gần nhất.
- Kết hợp với chương trình khuyến mãi và thẻ thành viên.
- Tạo một hệ thống tích điểm thưởng cho khách hàng thân thiết.

- Phát triển tính năng đặt hàng trước và giao hàng tận nơi.

## 2.Ý tưởng phát triển

**Công nghệ và công cụ:**

- Sử dụng Android Studio để phát triển ứng dụng Android.
- Sử dụng Java hoặc Kotlin làm ngôn ngữ lập trình chính.

**Các bước cơ bản:**

- Thiết kế giao diện người dùng tốt hơn: Tạo các màn hình và cài đặt layout cho các mục đích khác nhau của ứng dụng.
- Lập trình logic: Xử lý sự kiện người dùng, truy xuất dữ liệu và quản lý luồng ứng dụng.
- Tích hợp API và dịch vụ: Kết nối với các API thanh toán và dịch vụ vị trí nếu cần.
- Kiểm thử và triển khai: Kiểm tra ứng dụng trên nhiều thiết bị và triển khai lên Google Play Store.

## 3.Ưu điểm và lợi ích

**Ưu điểm:**

- Giúp cửa hàng cafe tiếp cận khách hàng một cách dễ dàng và thuận tiện hơn.
- Tăng trải nghiệm người dùng và khả năng tương tác với thương hiệu.
- Tăng doanh thu bằng việc mở rộng kênh bán hàng và tạo mối quan hệ khách hàng tốt hơn.

**Nhược điểm:**

- Yêu cầu kỹ năng lập trình và thiết kế đồ họa để phát triển một ứng dụng chất lượng.
- Cần đầu tư thời gian và nguồn lực cho việc triển khai, duy trì và hỗ trợ ứng dụng sau khi ra mắt.

# Chương 5.KẾT LUẬN

Phát triển ứng dụng bán cafe trên nền tảng Android là một ý tưởng sáng tạo và mang lại nhiều lợi ích cho cả doanh nghiệp và người dùng. Quá trình phát triển cần được tiếp cận một cách toàn diện từ thiết kế, lập trình cho đến triển khai và quản lý sau khi ra mắt để đạt được hiệu quả tốt nhất. Nếu có sự đầu tư và nỗ lực phát triển, ứng dụng bán cafe có thể trở thành một công cụ quan trọng trong việc mở rộng kinh doanh và tăng cường trải nghiệm khách hàng.

Việc phát triển ứng dụng bán cafe trên nền tảng Android là một ý tưởng có tiềm năng mang lại nhiều lợi ích cho cả người dùng và các doanh nghiệp trong ngành dịch vụ ẩm thực. Dưới đây là một bản tóm tắt và kết luận về ứng dụng này:

**Lợi ích của ứng dụng bán cafe trên Android:**

- Tiện lợi và tiết kiệm thời gian
- Tăng trải nghiệm khách hàng
- Mở rộng thị trường và doanh số
- Quản lý đơn hàng hiệu quả
- Tích hợp các tính năng tiện ích

**Nhược điểm và thách thức:**

- Chi phí và thời gian: Phát triển ứng dụng di động đòi hỏi đầu tư chi phí và thời gian để nghiên cứu, thiết kế, lập trình và kiểm thử.
- Cạnh tranh thị trường: Thị trường ứng dụng di động đầy cạnh tranh, vì vậy cần có một chiến lược tiếp thị tốt để thu hút người dùng.
- Bảo mật và tính ổn định: Đảm bảo tính bảo mật và ổn định của ứng dụng là một thách thức đối với các nhà phát triển. Cần phải đảm bảo rằng dữ liệu của khách hàng được bảo vệ và giao dịch diễn ra một cách an toàn.

Tóm lại, việc phát triển một ứng dụng bán cafe trên nền tảng Android mang lại nhiều lợi ích và tiềm năng kinh doanh. Tuy nhiên, để thành công, cần có sự đầu tư và nỗ lực trong quá trình phát triển và quản lý ứng dụng.

# DANH MỤC TÀI LIỆU THAM KHẢO

*https://viblo.asia/p/gioi-thieu-ngon-ngu-dart-ORNZqdv3K0n*

*https://viblo.asia/p/tim-hieu-ve-providers-trong-flutter-4dbZNXrn5YM*

*https://aws.amazon.com/vi/what-is/flutter/*

*https://developer.android.com/studio?hl=vi*