BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

ĐỒ ÁN MÔN HỌC

LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG

GỢI Ý LỊCH TRÌNH 3N2Đ

Ngành : **KHOA HỌC DỮ LIỆU**

Chuyên ngành: **KHOA HỌC DỮ LIỆU**

Giảng viên hướng dẫn : ThS. Nguyễn Hữu Trung

Sinh viên thực hiện :

2286400022 – Hồ Nguyễn Hoàng Phát

2286400481 – Nguyễn Thị Thu Ngân

2286400001 – Phạm Quốc An

Lớp: 22DKHA1

TP. Hồ Chí Minh, 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

ĐỒ ÁN MÔN HỌC

LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG

GỢI Ý LỊCH TRÌNH 3N2Đ

Ngành : **KHOA HỌC DỮ LIỆU**

Chuyên ngành: **KHOA HỌC DỮ LIỆU**

Giảng viên hướng dẫn : ThS. Nguyễn Hữu Trung

Sinh viên thực hiện :

2286400022 – Hồ Nguyễn Hoàng Phát

2286400481 – Nguyễn Thị Thu Ngân

2286400001 – Phạm Quốc An

Lớp: 22DKHA1

TP. Hồ Chí Minh, 2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

TPHCM, Ngày tháng năm 2025

Giáo viên hướng dẫn

(Ký tên, đóng dấu)

LỜI CAM ĐOAN

Chúng tôi Phạm Quốc An, Nguyễn Thị Thu Ngân, Hồ Nguyễn Hoàng Phát xin cam đoan rằng:

Tất cả thông tin và phân tích trình bày trong báo cáo này được thực hiện một cách chính xác và trung thực.

Mọi dữ liệu, nhận định hoặc ý kiến được trích dẫn từcác nguồn khác đều đã được nêu rõ nguồn gốc và trích dẫn đúng quy định. Chúng tôi cam đoan rằng không có bất kỳ hành vi sao chép hoặc sử dụng thông tin không hợp pháp nào từ các nguồn khác.

Bài báo cáo này là kết quả của công trình nghiên cứu độc lập của chúng tôi và chưa từng được công bố tại bất kỳ nơi nào khác. Tôi cam đoan đã tuân thủ nghiêm ngặt các quy tắc và quy định của môn học, bao gồm việc tham khảo và áp dụng các công cụ nghiên cứu một cách hợp lệ. Nếu phát hiện có bất kỳ sựgian lận nào, chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung bài báo cáo của mình.

Hy vọng rằng bài báo cáo này sẽ cung cấp những thông tin hữu ích cho các nhà nghiên cứu, doanh nghiệp.

TP. Hồ Chí Minh, Ngày _ tháng _ năm 2026

MỤC LỤC

Danh sách bảng

DANH MỤC HÌNH VẼ

DANH MỤC KÝ HIỆU VÀ CHỮ VIẾT TẮT

Ký hiệu / Viết tắt	**Diễn giải**
3N2Đ	3 ngày 2 đêm (hình thức du lịch ngắn ngày)
SDK	Software Development Kit – bộ công cụ phát triển (ví dụ Flutter SDK)
UI	User Interface – giao diện người dùng
UX	User Experience – trải nghiệm người dùng
API	Application Programming Interface – giao diện lập trình ứng dụng
DB	Database – cơ sở dữ liệu (ví dụ: *travel_guides.db*)
JSON	JavaScript Object Notation – định dạng dữ liệu
CRUD	Create, Read, Update, Delete – các thao tác cơ bản với dữ liệu

CHƯƠNG 1: TỔNG QUAN

1.1 Giới thiệu đề tài

Trong những năm gần đây, du lịch ngắn ngày đang trở thành xu hướng phổ biến, đặc biệt là các chuyến đi kéo dài 3 ngày 2 đêm (3N2Đ). Hình thức du lịch này phù hợp với sinh viên và người đi làm vì không tốn quá nhiều thời gian nhưng vẫn đảm bảo được trải nghiệm nghỉ ngơi và khám phá.

Tuy nhiên, việc tự lên lịch trình du lịch thường gặp nhiều khó khăn như sắp xếp thời gian chưa hợp lý, lựa chọn địa điểm chưa phù hợp với ngân sách hoặc sở thích cá nhân. Điều này khiến nhiều người mất nhiều thời gian tìm kiếm thông tin hoặc có trải nghiệm chưa tối ưu.

Với sự phát triển mạnh mẽ của thiết bị di động, các ứng dụng di động đã trở thành công cụ hỗ trợ hiệu quả trong đời sống hằng ngày. Việc xây dựng một ứng dụng GoGo sẽ giúp người dùng dễ dàng tiếp cận các kế hoạch du lịch phù hợp, tiết kiệm thời gian và nâng cao trải nghiệm.

Bên cạnh đó, Flutter với ngôn ngữ Dart là một công cụ hiện đại, cho phép phát triển ứng dụng di động đa nền tảng với hiệu suất cao và giao diện đẹp. Việc áp dụng Flutter vào đề tài không chỉ mang ý nghĩa thực tiễn mà còn giúp sinh viên rèn luyện kỹ năng lập trình ứng dụng di động.

Chính vì những lý do trên, nhóm lựa chọn đề tài "Ứng dụng GoGo" để nghiên cứu và thực hiện trong môn học Lập trình trên thiết bị di động.

1.2 Nhiệm vụ của đề tài

Nhiệm vụ của đề tài là nghiên cứu và tìm hiểu tổng quan về nhu cầu du lịch ngắn ngày, đặc biệt là các chuyến đi 3 ngày 2 đêm, từ đó phân tích yêu cầu người dùng và xác định các chức năng cần thiết cho ứng dụng. Đồng thời, đề tài tập trung tìm hiểu framework Flutter và ngôn ngữ lập trình Dart nhằm xây dựng một ứng dụng di động có

giao diện thân thiện, dễ sử dụng và phù hợp với mục tiêu gợi ý lịch trình du lịch.

Bên cạnh đó, đề tài tiến hành thiết kế giao diện người dùng, xây dựng mô hình dữ liệu và cài đặt chức năng gợi ý lịch trình du lịch 3N2Đ theo từng ngày. Cuối cùng, ứng dụng được kiểm thử ở mức cơ bản để đánh giá khả năng hoạt động, từ đó rút ra nhận xét, chỉ ra các hạn chế còn tồn tại và đề xuất hướng phát triển trong tương lai.

1.2.1 Tính cấp thiết

Hiện nay, việc lên kế hoạch cho các chuyến du lịch ngắn ngày chủ yếu dựa vào kinh nghiệm cá nhân hoặc thông tin rời rạc trên Internet, gây mất thời gian và thiếu tính hệ thống. Đối với các chuyến đi 3N2Đ, nếu không có lịch trình hợp lý, người dùng dễ gặp tình trạng quá tải hoạt động hoặc lãng phí thời gian.

Do đó, việc xây dựng một ứng dụng có khả năng gợi ý lịch trình du lịch 3N2Đ một cách nhanh chóng và tiện lợi là cần thiết. Ứng dụng giúp người dùng tiếp cận lịch trình mẫu phù hợp với nhu cầu, góp phần nâng cao chất lượng trải nghiệm du lịch.

1.2.2 Ý nghĩa khoa học

Đề tài giúp vận dụng các kiến thức lý thuyết đã học trong môn Lập trình trên thiết bị di động, bao gồm thiết kế giao diện, quản lý trạng thái và xử lý dữ liệu trong Flutter. Thông qua quá trình thực hiện đề tài, sinh viên có cơ hội tiếp cận quy trình phát triển một ứng dụng di động hoàn chỉnh.

Ngoài ra, đề tài còn góp phần nghiên cứu cách tổ chức và biểu diễn dữ liệu lịch trình du lịch dưới dạng cấu trúc, tạo nền tảng cho việc mở rộng sang các hệ thống gợi ý thông minh trong tương lai.

1.2.3 Tính thực tiễn

Ứng dụng GoGo có khả năng áp dụng trực tiếp vào thực tế, hỗ trợ người dùng trong việc lập kế hoạch du lịch cá nhân. Với giao diện đơn giản và dễ sử dụng, ứng dụng phù hợp với nhiều đối tượng người dùng khác nhau.

Bên cạnh đó, sản phẩm của đề tài có thể tiếp tục được phát triển, tích hợp thêm các tính năng như bản đồ, đánh giá địa điểm hoặc gợi ý thông minh dựa trên sở thích người dùng.

1.3 Mục tiêu đề tài

Mục tiêu của đề tài là xây dựng một ứng dụng di động gợi ý lịch trình du lịch 3 ngày 2 đêm nhằm hỗ trợ người dùng trong việc lập kế hoạch cho các chuyến du lịch ngắn ngày một cách thuận tiện và hiệu quả. Ứng dụng được thiết kế để giúp người dùng dễ dàng tiếp cận các lịch trình du lịch mẫu, từ đó tiết kiệm thời gian tìm kiếm và sắp xếp kế hoạch.

Bên cạnh đó, ứng dụng cho phép người dùng lựa chọn các thông tin cơ bản như điểm đến du lịch, mức ngân sách dự kiến và sở thích cá nhân. Dựa trên các thông tin này, hệ thống sẽ gợi ý lịch trình phù hợp và hiển thị chi tiết kế hoạch cho từng ngày trong chuyến đi 3N2Đ, bao gồm các hoạt động tham quan và thời gian thực hiện.

Thông qua việc thực hiện đề tài, sinh viên có cơ hội rèn luyện và nâng cao kỹ năng lập trình Flutter với ngôn ngữ Dart, đồng thời áp dụng kiến thức đã học vào việc xây dựng một ứng dụng di động hoàn chỉnh, từ khâu thiết kế giao diện đến xử lý logic và tổ chức mã nguồn.

1.4 Phạm vi nghiên cứu

Trong khuôn khổ của môn học Lập trình trên thiết bị di động, đề tài được triển khai trong phạm vi nhất định nhằm đảm bảo phù hợp với thời gian và yêu cầu học phần. Ứng dụng được phát triển bằng Flutter Dart và chạy trên nền tảng Android, phục vụ mục đích học tập và nghiên cứu.

Chức năng gợi ý lịch trình du lịch được xây dựng theo hướng cố định hoặc bán tự động, dựa trên các mẫu lịch trình được thiết kế sẵn. Dữ liệu về địa điểm du lịch, hoạt động và lịch trình chủ yếu mang tính minh họa, được lưu trữ cục bộ trong ứng dụng và chưa kết nối với các cơ sở dữ liệu lớn hoặc dịch vụ bên ngoài.

Đề tài chưa tập trung vào việc tích hợp các chức năng nâng cao như bản đồ trực tuyến, đặt vé hay thanh toán, mà chủ yếu hướng đến việc xây dựng một ứng dụng có cấu trúc rõ ràng, giao diện thân thiện và đáp ứng được yêu cầu cơ bản của bài toán gợi ý lịch trình du lịch 3 ngày 2 đêm.

1.5 Cấu trúc báo cáo

Nội dung báo cáo được chia thành các chương như sau:

- **Chương 1**: Giới thiệu đề tài.
- **Chương 2**: Cơ sở lý thuyết.
- **Chương 3**: Phân tích và thiết kế hệ thống.
- **Chương 4**: Xây dựng và cài đặt ứng dụng.
- **Chương 5**: Kết quả thực hiện và đánh giá.
- **Chương 6**: Kết luận và hướng phát triển.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về ứng dụng di động

Ứng dụng di động là các phần mềm được thiết kế để chạy trên các thiết bị di động như điện thoại thông minh và máy tính bảng. Với sự phát triển mạnh mẽ của công nghệ và Internet, ứng dụng di động ngày càng đóng vai trò quan trọng trong đời sống hằng ngày, hỗ trợ người dùng trong nhiều lĩnh vực như học tập, giải trí, mua sắm và du lịch.

Trong lĩnh vực du lịch, ứng dụng di động giúp người dùng dễ dàng tra cứu thông tin, lập kế hoạch chuyến đi và quản lý lịch trình. Việc xây dựng một ứng dụng gợi ý lịch trình du lịch 3 ngày 2 đêm góp phần nâng cao trải nghiệm người dùng, đồng thời đáp ứng nhu cầu sử dụng thiết bị di động ngày càng phổ biến hiện nay.

2.2 Giới thiệu về Flutter và Dart

2.2.1 Khái niệm Flutter

Hình 2.1 Flutter

Flutter là một framework mã nguồn mở do Google phát triển, được sử dụng để xây dựng các ứng dụng đa nền tảng từ một mã nguồn duy nhất. Flutter cho phép phát triển ứng dụng chạy trên Android, iOS và một số nền tảng khác với hiệu suất cao và giao diện đẹp.

Một trong những ưu điểm nổi bật của Flutter là khả năng xây dựng giao diện dựa trên hệ thống widget linh hoạt. Flutter hỗ trợ cơ chế hot reload, giúp lập trình viên dễ dàng chỉnh sửa và quan sát kết quả ngay lập tức, từ đó nâng cao hiệu quả phát triển ứng dụng. Nhờ những ưu điểm này, Flutter là công cụ phù hợp để phát triển ứng dụng trong khuôn khổ môn Lập trình trên thiết bị di động.

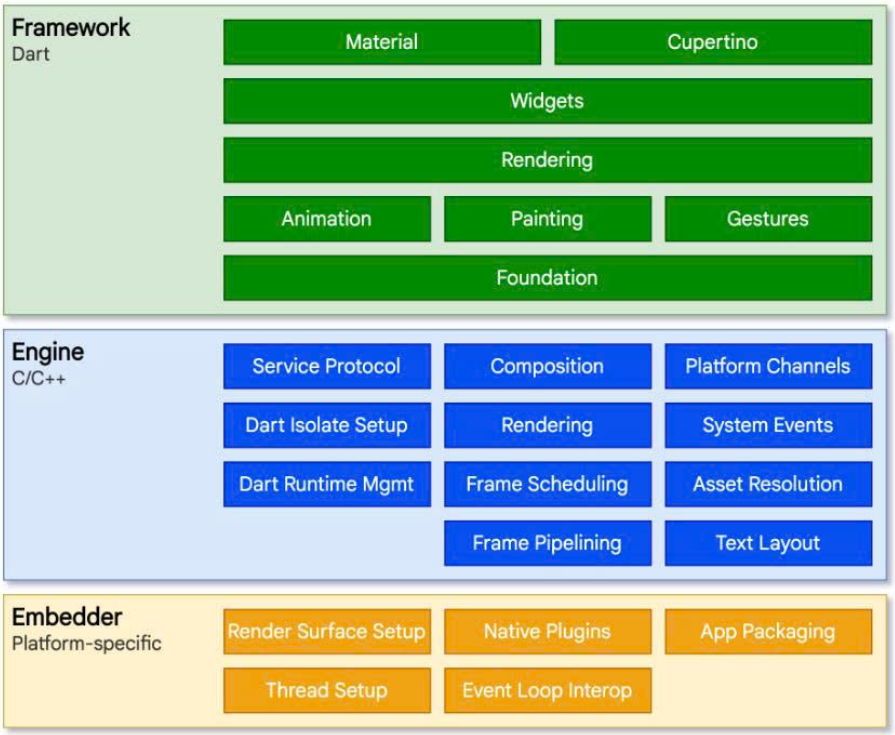
2.2.2 Ngôn ngữ lập trình Dart

Hình 2.2 Dart

Dart là ngôn ngữ lập trình được Google phát triển và sử dụng chính trong framework Flutter. Dart là ngôn ngữ hướng đối tượng, có cú pháp đơn giản, dễ học và phù hợp cho việc phát triển ứng dụng di động.

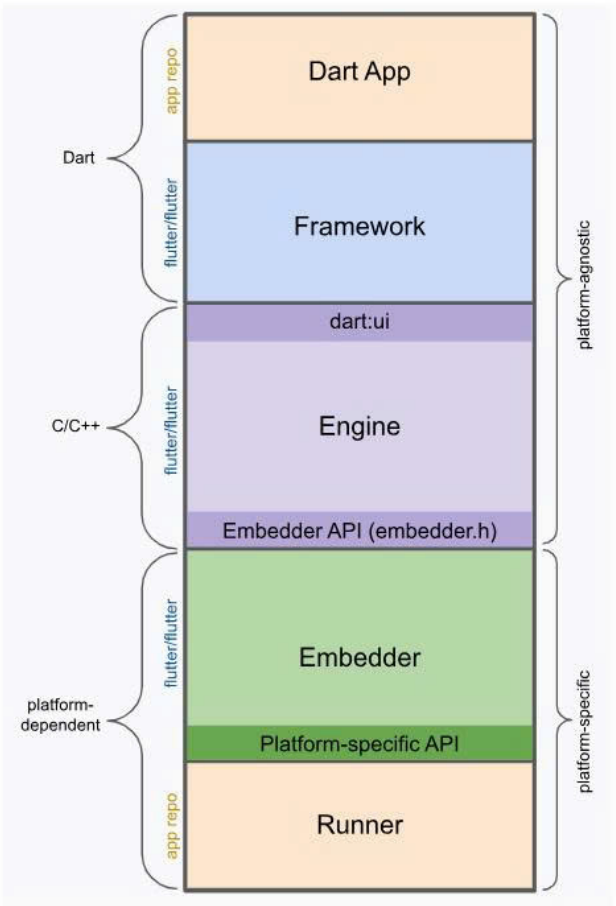
Dart hỗ trợ lập trình bất đồng bộ thông qua Future và async await, giúp xử lý các tác vụ như tải dữ liệu hoặc xử lý logic một cách hiệu quả. Việc sử dụng Dart trong đề tài giúp sinh viên làm quen với một ngôn ngữ lập trình hiện đại và nâng cao kỹ năng lập trình ứng dụng di động.

2.3 Kiến trúc ứng dụng Flutter



Hình 2.3 lớp kiến trúc

Flutter được thiết kế như một hệ thống phân lớp, có khả năng mở rộng. Nó tồn tại dưới dạng một chuỗi các thư viện độc lập, mỗi thư viện phụ thuộc vào lớp nền bên dưới. Không có lớp nào có quyền truy cập đặc quyền vào lớp bên dưới, và mọi phần của khung phần mềm đều được thiết kế để tùy chọn và có thể thay thế.



Hình 2.4 Kiến trúc phân lớp của Flutter

Đối với hệ điều hành nền tảng, các ứng dụng Flutter được đóng gói theo cùng một cách như bất kỳ ứng dụng gốc nào khác. Một trình nhúng dành riêng cho nền tảng cung cấp điểm truy cập; phối hợp với hệ điều hành nền tảng để truy cập các dịch vụ như bề mặt hiển thị, khả năng truy cập và nhập liệu; và quản lý vòng lặp sự kiện thông báo. Trình nhúng được viết bằng ngôn ngữ phù hợp với nền tảng: hiện tại là Java và C++ cho Android, Swift và Objective-C/Objective-C++ cho iOS và macOS, và C++ cho Windows và Linux. Sử dụng trình nhúng, mã Flutter có thể được tích hợp vào một ứng dụng hiện có dưới dạng một mô-đun, hoặc mã đó có thể là toàn bộ nội dung của ứng dụng. Flutter bao gồm một số trình nhúng cho các nền tảng mục tiêu phổ biến, nhưng cũng tồn tại các trình nhúng khác.

Cốt lõi của Flutter là công cụ Flutter (Flutter engine), chủ yếu được viết bằng C++ và hỗ trợ các thành phần cơ bản cần thiết để vận hành tất cả các ứng dụng Flutter. Công cụ này chịu trách nhiệm tạo ảnh bitmap cho các cảnh ghép nối mỗi khi cần vẽ một khung

hình mới. Nó cung cấp triển khai cấp thấp của API cốt lõi của Flutter, bao gồm bố cục văn bản đồ họa, nhập/xuất tệp và mạng, môi trường chạy Dart và chuỗi công cụ biên dịch.

2.4 Công cụ và môi trường phát triển

Trong quá trình thực hiện đề tài **ứng dụng GoGo**, nhóm đã sử dụng nhiều công cụ và môi trường phát triển nhằm hỗ trợ hiệu quả cho việc xây dựng, kiểm thử và hoàn thiện ứng dụng di động. Việc lựa chọn các công cụ phù hợp đóng vai trò quan trọng, giúp quá trình phát triển diễn ra thuận lợi, tiết kiệm thời gian và đảm bảo chất lượng sản phẩm.

Công cụ chính được sử dụng là **Flutter SDK**, đây là bộ công cụ phát triển do Google cung cấp, hỗ trợ xây dựng ứng dụng di động đa nền tảng từ một mã nguồn duy nhất. Flutter SDK cung cấp đầy đủ các thư viện, widget và công cụ cần thiết để thiết kế giao diện, xử lý logic và biên dịch ứng dụng. Nhờ cơ chế hot reload, lập trình viên có thể nhanh chóng quan sát sự thay đổi của giao diện khi chỉnh sửa mã nguồn, từ đó nâng cao hiệu suất làm việc.

Ngôn ngữ lập trình **Dart** được sử dụng xuyên suốt trong quá trình phát triển ứng dụng. Dart có cú pháp rõ ràng, dễ tiếp cận và hỗ trợ lập trình hướng đối tượng, giúp việc tổ chức mã nguồn trở nên khoa học và dễ bảo trì. Ngoài ra, Dart còn hỗ trợ lập trình bất đồng bộ hiệu quả, phù hợp với các ứng dụng di động cần xử lý nhiều tác vụ song song.

Về môi trường lập trình, **Android Studio** và **Visual Studio Code** được sử dụng làm công cụ viết mã nguồn. Android Studio cung cấp hệ thống quản lý dự án, trình giả lập thiết bị Android và các công cụ hỗ trợ gỡ lỗi mạnh mẽ. Trong khi đó, Visual Studio Code có ưu điểm là nhẹ, dễ sử dụng và hỗ trợ nhiều tiện ích mở rộng dành riêng cho Flutter và Dart, giúp quá trình lập trình trở nên linh hoạt hơn.

Để kiểm thử ứng dụng trong quá trình phát triển, nhóm sử dụng **Android Emulator** nhằm mô phỏng các thiết bị Android với nhiều kích thước màn hình và phiên bản hệ điều hành khác nhau. Việc sử dụng trình giả lập giúp kiểm tra giao diện, chức năng và khả năng tương thích của ứng dụng trước khi triển khai thực tế.

Ngoài ra, một số công cụ hỗ trợ khác như Git được sử dụng để quản lý mã nguồn, giúp theo dõi quá trình thay đổi và dễ dàng khôi phục khi xảy ra lỗi. Các tài liệu hướng dẫn chính thức từ Flutter và Dart cũng được tham khảo nhằm đảm bảo việc triển khai đúng chuẩn và hiệu quả.

Nhìn chung, việc sử dụng các công cụ và môi trường phát triển phù hợp đã góp phần quan trọng trong việc hoàn thành đề tài, đồng thời giúp sinh viên làm quen với quy trình phát triển một ứng dụng di động thực tế bằng Flutter và Dart.

Bên cạnh các công cụ và môi trường phát triển chính, ứng dụng còn sử dụng một số package quan trọng nhằm hỗ trợ triển khai các chức năng cốt lõi. Trong đó, package `sqflite` được sử dụng để quản lý cơ sở dữ liệu SQLite, phục vụ cho việc lưu trữ và xử lý dữ liệu lịch trình du lịch trong ứng dụng. Package `path` hỗ trợ xác định và quản lý đường dẫn lưu trữ cơ sở dữ liệu trên thiết bị di động một cách chính xác và an toàn.

Ngoài ra, package `shared_preferences` được sử dụng để lưu trữ các thông tin cấu hình và lựa chọn của người dùng dưới dạng dữ liệu cục bộ đơn giản. Cách tiếp cận này giúp ứng dụng duy trì trạng thái và thông tin cần thiết giữa các lần sử dụng mà không cần phụ thuộc vào kết nối Internet. Bên cạnh đó, ứng dụng còn sử dụng các package hỗ trợ xác thực người dùng và xử lý ngoại lệ nhằm nâng cao tính ổn định và độ tin cậy của hệ thống.

Trong quá trình xác thực và xử lý dữ liệu, ứng dụng được thiết kế với cơ chế bắt và xử lý ngoại lệ nhằm đảm bảo hệ thống không bị gián đoạn khi xảy ra lỗi. Các ngoại lệ phổ biến như lỗi đăng nhập, lỗi truy xuất cơ sở dữ liệu hoặc lỗi dữ liệu đầu vào không hợp lệ đều được xử lý và thông báo phù hợp đến người dùng. Việc xử lý ngoại lệ giúp nâng cao trải nghiệm người dùng và đảm bảo ứng dụng hoạt động ổn định trong nhiều tình huống khác nhau.

2.5 Công nghệ và các thành phần kỹ thuật sử dụng trong ứng dụng

2.5.1 Thư viện và package sử dụng

Trong quá trình phát triển ứng dụng GoGo, đề tài sử dụng các thư viện và package cơ bản của Flutter nhằm phục vụ việc xây dựng giao diện, xử lý dữ liệu và quản lý luồng hoạt động của ứng dụng. Việc sử dụng các thư viện phù hợp giúp giảm thời gian phát triển, đồng thời đảm bảo ứng dụng hoạt động ổn định và dễ mở rộng.

Thư viện **material** được sử dụng làm nền tảng chính để xây dựng giao diện người dùng theo chuẩn Material Design. Thư viện này cung cấp đầy đủ các widget cần thiết như AppBar, Button, Card, ListView và TextField, giúp việc thiết kế giao diện trở nên nhất quán và thân thiện với người dùng.

Ngoài ra, một số package hỗ trợ khác như intl được sử dụng để định dạng dữ liệu hiển thị, trong khi **shared preferences** có thể được sử dụng để lưu trữ các thông tin đơn giản của người dùng. Việc lựa chọn các package này phù hợp với quy mô và mục tiêu của đề tài, đảm bảo tính đơn giản nhưng vẫn đáp ứng yêu cầu chức năng.

2.5.2 Quản lý giao diện và điều hướng trong Flutter

Quản lý giao diện và điều hướng giữa các màn hình là một nội dung quan trọng trong quá trình xây dựng ứng dụng di động, đặc biệt đối với các ứng dụng có nhiều màn hình chức năng. Flutter cung cấp cơ chế điều hướng thông qua lớp Navigator, cho phép quản lý các màn hình dưới dạng các route được lưu trữ trong một ngăn xếp. Cơ chế này giúp việc chuyển đổi giữa các màn hình trở nên linh hoạt và dễ kiểm soát.

Trong đề tài, cơ chế điều hướng được sử dụng để chuyển đổi giữa các màn hình chính của ứng dụng như màn hình nhập thông tin chuyến đi, màn hình gợi ý lịch trình và màn hình hiển thị chi tiết lịch trình theo từng ngày. Khi người dùng thực hiện các thao tác như nhấn nút xác nhận hoặc lựa chọn xem chi tiết, hệ thống sẽ điều hướng sang màn hình tương ứng và truyền các dữ liệu cần thiết.

Việc tổ chức điều hướng hợp lý giúp ứng dụng hoạt động mạch lạc, hạn chế nhầm

lẫn trong quá trình sử dụng và nâng cao trải nghiệm người dùng. Ngoài ra, cơ chế điều hướng của Flutter còn cho phép dễ dàng mở rộng thêm các màn hình mới trong tương lai mà không ảnh hưởng đến cấu trúc tổng thể của ứng dụng.

2.5.3 Quản lý trạng thái trong ứng dụng Flutter

Quản lý trạng thái là yếu tố quan trọng ảnh hưởng trực tiếp đến tính tương tác và hiệu quả hoạt động của ứng dụng Flutter. Trạng thái trong ứng dụng có thể bao gồm các thông tin người dùng nhập vào, các lựa chọn về điểm đến, ngân sách, sở thích du lịch cũng như lịch trình được hệ thống gợi ý.

Trong phạm vi đề tài, việc quản lý trạng thái được thực hiện bằng cách sử dụng StatefulWidget kết hợp với hàm setState. Phương pháp này phù hợp với quy mô nhỏ và vừa của ứng dụng, giúp sinh viên dễ dàng tiếp cận và triển khai mà không cần sử dụng các thư viện quản lý trạng thái phức tạp.

Khi trạng thái của ứng dụng thay đổi, Flutter sẽ tự động xây dựng lại các widget liên quan và cập nhật giao diện tương ứng. Nhờ đó, dữ liệu hiển thị trên màn hình luôn đồng bộ với dữ liệu xử lý bên trong. Việc quản lý trạng thái đúng cách giúp ứng dụng hoạt động ổn định, giảm lỗi phát sinh và mang lại trải nghiệm tốt cho người dùng.

2.5.4 Thiết kế giao diện người dùng theo Material Design

Material Design là hệ thống thiết kế giao diện do Google phát triển và được tích hợp sẵn trong Flutter. Hệ thống này cung cấp các nguyên tắc và thành phần giao diện nhằm tạo ra trải nghiệm người dùng trực quan, nhất quán và dễ sử dụng.

Trong đề tài, Material Design được áp dụng để xây dựng giao diện cho các màn hình chính của ứng dụng. Các thành phần như thanh tiêu đề, nút bấm, thẻ thông tin và danh sách nội dung được sử dụng một cách hợp lý, đảm bảo giao diện rõ ràng và dễ thao tác.

Bố cục giao diện được thiết kế theo hướng đơn giản, tập trung vào nội dung chính là lịch trình du lịch 3 ngày 2 đêm. Việc áp dụng Material Design giúp giao diện ứng

dụng phù hợp với tiêu chuẩn thiết kế trên Android, đồng thời nâng cao tính thẩm mỹ và khả năng sử dụng của ứng dụng.

2.5.5 Lưu trữ dữ liệu trong ứng dụng

Để phục vụ chức năng gợi ý lịch trình, ứng dụng cần lưu trữ các thông tin liên quan đến địa điểm du lịch, chi phí và các lịch trình mẫu. Trong phạm vi đề tài, dữ liệu được lưu trữ chủ yếu dưới dạng các đối tượng và danh sách trong bộ nhớ ứng dụng.

Ngoài ra, ứng dụng có thể sử dụng cơ chế lưu trữ cục bộ như shared preferences để lưu một số thông tin đơn giản của người dùng, chẳng hạn như lựa chọn điểm đến gần nhất hoặc lịch trình đã xem. Việc lưu trữ cục bộ giúp ứng dụng hoạt động ổn định ngay cả khi không có kết nối Internet.

Cách tiếp cận này phù hợp với mục tiêu học tập của đề tài, giúp giảm độ phức tạp trong triển khai và tạo nền tảng cho việc mở rộng hệ thống lưu trữ dữ liệu trong tương lai.

2.5.6 Đánh giá lựa chọn công nghệ

Việc lựa chọn Flutter và Dart để phát triển ứng dụng mang lại nhiều lợi ích cho đề tài. Flutter cho phép xây dựng giao diện nhanh chóng, dễ tùy biến và phù hợp với nền tảng Android. Ngôn ngữ Dart có cú pháp rõ ràng, dễ tiếp cận đối với sinh viên mới học lập trình di động.

Các công nghệ và thư viện được lựa chọn đều phù hợp với mục tiêu học tập của môn Lập trình trên thiết bị di động. Đồng thời, chúng tạo nền tảng thuận lợi cho việc mở rộng ứng dụng trong tương lai, chẳng hạn như tích hợp cơ sở dữ liệu trực tuyến hoặc các thuật toán gợi ý nâng cao hơn.

2.6 Phân tích yêu cầu hệ thống và chức năng

2.6.1 Phân tích hệ thống

Phân tích yêu cầu hệ thống là bước quan trọng nhằm xác định rõ các chức năng cần có của ứng dụng và các yêu cầu liên quan trước khi tiến hành thiết kế và cài đặt. Việc phân tích đúng yêu cầu giúp hệ thống đáp ứng tốt nhu cầu của người dùng và tránh sai sót trong quá trình phát triển.

Ứng dụng GoGo là một ứng dụng di động được phát triển bằng Flutter, cung cấp các gợi ý lịch trình du lịch 3 ngày 2 đêm cho các thành phố phổ biến tại Việt Nam. Hệ thống được thiết kế với kiến trúc module hóa, bao gồm các thành phần chính sau:

1. **Module xác thực người dùng**: Quản lý toàn bộ quy trình đăng nhập, đăng ký tài khoản và khôi phục mật khẩu. Module này sử dụng cơ chế lưu trữ cục bộ thông qua `SharedPreferences` để duy trì trạng thái phiên làm việc của người dùng.

2. **Module danh sách thành phố**: Hiển thị danh sách các điểm đến du lịch dưới dạng card trực quan, mỗi card bao gồm hình ảnh đại diện và tên thành phố. Người dùng có thể lựa chọn thành phố muốn xem lịch trình chi tiết.

3. **Module chi tiết lịch trình**: Cung cấp giao diện hiển thị thông tin đầy đủ về lịch trình du lịch thông qua 3 tab chính:

   - **Tổng quan**: Thông tin cơ bản, khách sạn, phương tiện di chuyển.
   - **Cụ thể**: Lịch trình chi tiết 3 ngày 2 đêm theo khung giờ.
   - **Chuẩn bị**: Vật dụng cần mang, mẹo du lịch, chi phí ước tính.

4. **Module dữ liệu địa phương**: Lưu trữ và quản lý thông tin về khách sạn, phương tiện di chuyển (xe khách, máy bay) và các điểm tham quan bổ sung. Dữ liệu được tổ chức dưới dạng cấu trúc đối tượng trong bộ nhớ ứng dụng.

Ứng dụng hướng đến việc hỗ trợ người dùng lập kế hoạch du lịch nhanh chóng, đơn giản và dễ sử dụng. Người dùng chủ yếu là sinh viên hoặc người đi làm có nhu cầu du

lịch ngắn ngày và sử dụng thiết bị di động Android. Thiết kế hệ thống tập trung vào tính trực quan, tốc độ phản hồi và khả năng hoạt động ổn định trên nhiều thiết bị khác nhau.

2.6.2 Phân tích chức năng

Dựa trên mục tiêu đề tài, hệ thống cần đáp ứng các yêu cầu chức năng cơ bản sau:

a) Chức năng xác thực người dùng

- **Đăng ký tài khoản**: Cho phép người dùng tạo tài khoản mới bằng cách nhập địa chỉ email và mật khẩu. Hệ thống kiểm tra tính hợp lệ của email và đảm bảo mật khẩu đủ mạnh (tối thiểu 6 ký tự).
- **Đăng nhập**: Xác thực người dùng với thông tin tài khoản đã đăng ký. Hệ thống so sánh thông tin nhập vào với dữ liệu lưu trữ và cấp quyền truy cập nếu thông tin chính xác.
- **Quên mật khẩu**: Cung cấp chức năng khôi phục mật khẩu thông qua email. Trong phiên bản hiện tại, chức năng này hoạt động ở chế độ demo, mô phỏng việc gửi hướng dẫn đặt lại mật khẩu.
- **Đăng xuất**: Kết thúc phiên làm việc của người dùng, xóa thông tin phiên đăng nhập và chuyển hướng về màn hình đăng nhập.

b) Chức năng quản lý du lịch

- **Hiển thị danh sách thành phố**: Cung cấp giao diện hiển thị 3 thành phố du lịch mẫu (Phan Thiết, Đà Lạt, Nha Trang) dưới dạng card trực quan với hình ảnh đại diện và tên thành phố.
- **Xem chi tiết lịch trình**: Triển khai giao diện với 3 tab chính để trình bày thông tin lịch trình:
  - **Tab 1 Tổng quan**: Hiển thị thông tin cơ bản về chuyến đi, gợi ý khách sạn, phương tiện di chuyển (xe khách, máy bay) và các điểm tham quan bổ sung.
  - **Tab 2 Cụ thể**: Trình bày lịch trình chi tiết 3 ngày 2 đêm, phân chia theo từng khung thời gian (sáng, trưa, chiều, tối) với các hoạt động cụ thể.
  - **Tab 3 Chuẩn bị**: Cung cấp thông tin hữu ích cho chuyến đi bao gồm vật dụng cần mang, mẹo du lịch, dự trù chi phí và lưu ý về thời tiết.
- **Xem thông tin bổ sung**: Triển khai cơ chế `BottomSheet` để hiển thị chi tiết về khách sạn, thông tin xe khách và chuyến bay khi người dùng nhấn vào các nút tương ứng.

c) Chức năng quản lý dữ liệu

- **Lưu trữ cục bộ**: Sử dụng `SharedPreferences` để lưu trữ thông tin người dùng, bao gồm email phiên đăng nhập và danh sách tài khoản đã đăng ký. Cơ chế này đảm bảo ứng dụng hoạt động ngay cả khi không có kết nối Internet.
- **Dữ liệu tĩnh**: Các lịch trình mẫu được lưu trữ dưới dạng hard-code trong ứng dụng thông qua cấu trúc dữ liệu đối tượng. Mỗi thành phố có một `TripPlan` hoàn chỉnh với đầy đủ thông tin về khách sạn, phương tiện, lịch trình và các chuẩn bị cần thiết.

Các chức năng trên được thiết kế để hoạt động đồng bộ, tạo thành một hệ thống hoàn chỉnh hỗ trợ người dùng trong việc lập kế hoạch du lịch một cách thuận tiện và hiệu quả. Giao diện được tối ưu cho trải nghiệm người dùng trên thiết bị di động, với các thao tác đơn giản và trực quan.

2.6.3 Phân tích phi chức năng

1. **Giao diện thân thiện và trực quan**

   - Ứng dụng áp dụng nguyên tắc thiết kế **Material Design** do Google phát triển, đảm bảo giao diện nhất quán và dễ sử dụng.
   - Bố cục giao diện được thiết kế đơn giản, tập trung vào nội dung chính, giúp người dùng phổ thông dễ dàng tương tác mà không cần hướng dẫn phức tạp.

2. **Hiệu suất và tốc độ phản hồi**

   - Thời gian phản hồi của ứng dụng được tối ưu nhờ sử dụng dữ liệu cục bộ (hard-coded), loại bỏ thời gian chờ đợi tải dữ liệu từ server.
   - Các thao tác chuyển màn hình diễn ra mượt mà với hiệu ứng chuyển tiếp mặc định của Flutter, không gây gián đoạn trải nghiệm người dùng.
   - Cơ chế `FutureBuilder` được sử dụng để xử lý bất đồng bộ trong quá trình xác thực, đảm bảo giao diện không bị đơ khi chờ dữ liệu.

3. **Khả năng responsive và tương thích**

   - Ứng dụng được thiết kế responsive, hoạt động tốt trên nhiều kích thước màn hình khác nhau từ điện thoại đến máy tính bảng.
   - Sử dụng `ConstrainedBox` với *maxWidth: 420* để giới hạn chiều rộng nội dung trên màn hình lớn, đảm bảo tính thẩm mỹ và dễ đọc.
   - Hỗ trợ đa dạng phiên bản Android thông qua tương thích ngược với các widget cơ bản của Flutter.

4. **Độ tin cậy và ổn định**

   - Ứng dụng xử lý lỗi mạng một cách lịch sự khi tải ảnh từ URL, hiển thị placeholder hoặc icon thay thế khi không thể tải được ảnh.
   - Cơ chế validation được áp dụng cho tất cả các form nhập liệu, ngăn chặn dữ liệu không hợp lệ và cung cấp thông báo lỗi rõ ràng cho người dùng.

- Xử lý ngoại lệ (`try-catch`) được triển khai trong các tác vụ quan trọng như đăng nhập, đăng ký để tránh crash ứng dụng.

5. **Khả năng mở rộng và bảo trì**

   - Mã nguồn được tổ chức theo cấu trúc module rõ ràng, dễ dàng thêm mới thành phố hoặc chỉnh sửa lịch trình hiện có.
   - Sử dụng `SharedPreferences` làm lớp trừu tượng cho lưu trữ cục bộ, có thể dễ dàng thay thế bằng cơ sở dữ liệu SQLite hoặc kết nối backend trong tương lai.
   - Kiến trúc widget độc lập cho phép tái sử dụng component và dễ dàng nâng cấp giao diện mà không ảnh hưởng đến logic nghiệp vụ.

Các yêu cầu phi chức năng trên được thiết kế để đảm bảo ứng dụng không chỉ hoạt động đúng chức năng mà còn mang lại trải nghiệm người dùng tốt, dễ bảo trì và có tiềm năng phát triển lâu dài.

2.7 Thiết kế tổng thể

2.7.1 Thiết kế hệ thống

Hệ thống được thiết kế theo mô hình ứng dụng di động đơn giản, tập trung vào việc đảm bảo tính dễ sử dụng, dễ phát triển và phù hợp với mục tiêu của đề tài. Ứng dụng được xây dựng trên nền tảng Flutter với kiến trúc dựa trên widget, cho phép tái sử dụng các thành phần giao diện và dễ dàng mở rộng trong tương lai.

Về mặt tổng thể, hệ thống bao gồm các thành phần chính:

- **Thành phần dữ liệu**: Bao gồm các class định nghĩa cấu trúc dữ liệu như `Place`, `City`, `Trip Plan`, `Hotel Info`, `Bus Info`, `Flight Info`, `Day Schedule` và `Prep Section` được khai báo trong file `Gogo.dart`. Các class này đóng vai trò mô hình hóa thông tin về địa điểm, thành phố và lịch trình du lịch.
- **Thành phần giao diện**: Bao gồm các widget Flutter như `Home Screen`, `Trip Detail Screen`, `Login Screen`, `Register Screen`, `Edit Guide Screen`, và

`Forgot Password Screen`. Các thành phần này chịu trách nhiệm hiển thị thông tin và tiếp nhận thao tác từ người dùng.

- **Thành phần xử lý nghiệp vụ**: Được triển khai thông qua class `Auth Service` quản lý toàn bộ logic xác thực người dùng bao gồm đăng nhập, đăng ký, quên mật khẩu và quản lý phiên làm việc.
- **Thành phần lưu trữ**: Sử dụng `Shared Preferences` để lưu trữ thông tin người dùng và trạng thái phiên làm việc một cách cục bộ trên thiết bị.

Dữ liệu dữ lịch được lưu trữ dưới dạng hằng số (hard-coded) trong ứng dụng thông qua các collection trong file `Gogo.dart`, đảm bảo ứng dụng hoạt động ngay cả khi không có kết nối mạng. Cách thiết kế này giúp hệ thống hoạt động ổn định, logic rõ ràng và dễ bảo trì trong quá trình phát triển.

2.7.2 Thiết kế các màn hình chức năng

Dựa trên yêu cầu hệ thống, ứng dụng được thiết kế với các màn hình chức năng chính nhằm đảm bảo trải nghiệm người dùng liền mạch và thuận tiện.

Màn hình đăng nhập (Login Screen): Cung cấp form nhập email và mật khẩu để xác thực người dùng. Giao diện bao gồm các trường nhập liệu với validation, nút hiển thị/ẩn mật khẩu, cùng các liên kết đến chức năng đăng ký tài khoản mới và quên mật khẩu.

Màn hình đăng ký (Register Screen): Cho phép người dùng tạo tài khoản mới bằng cách nhập email, mật khẩu và xác nhận mật khẩu. Hệ thống kiểm tra tính hợp lệ của email và đảm bảo mật khẩu đủ mạnh trước khi tạo tài khoản.

Màn hình quên mật khẩu (Forgot Password Screen): Hỗ trợ người dùng khôi phục quyền truy cập bằng cách nhập địa chỉ email đã đăng ký để nhận hướng dẫn đặt lại mật khẩu (chế độ demo).

Màn hình chính (Home Screen): Hiển thị danh sách các thành phố du lịch dưới dạng card trực quan. Mỗi card bao gồm hình ảnh đại diện của thành phố và tên thành

phố, cho phép người dùng lựa chọn điểm đến muốn xem lịch trình chi tiết.

Màn hình chỉnh sửa lịch trình (Edit Guide Screen): Ứng dụng cung cấp màn hình chỉnh sửa lịch trình cho phép người dùng tùy chỉnh nội dung lịch trình đã tạo. Tại màn hình này, người dùng có thể chỉnh sửa thông tin chung của lịch trình, cập nhật các hoạt động theo từng ngày và thay đổi các nội dung liên quan đến chuyến đi.

Màn hình chi tiết lịch trình (Trip Detail Screen): Cung cấp giao diện hiển thị đầy đủ thông tin về lịch trình du lịch 3 ngày 2 đêm. Màn hình sử dụng Sliver AppBar với ảnh hero co giãn và TabBarView với 3 tab chính: (1) Tổng quan thông tin cơ bản, khách sạn, phương tiện, (2) Cụ thể lịch trình chi tiết theo từng ngày, (3) Chuẩn bị vật dụng cần mang, mẹo du lịch, chi phí.

2.7.3 Thiết kế dữ liệu

Dữ liệu trong ứng dụng được tổ chức dưới dạng các đối tượng và danh sách nhằm đảm bảo việc xử lý và hiển thị thông tin được thuận tiện. Các đối tượng dữ liệu chính bao gồm thông tin địa điểm du lịch, thông tin lịch trình và danh sách các hoạt động theo từng ngày.

Thông tin địa điểm du lịch (Place): Mô tả các điểm tham quan với các thuộc tính: tên địa điểm (name), mô tả (desc), và URL hình ảnh (image Url).

Thông tin thành phố (City): Đại diện cho mỗi điểm đến du lịch, bao gồm: tên thành phố (name), URL ảnh đại diện (heroImageUrl), và danh sách các địa điểm tham quan (places).

Thông tin lịch trình du lịch (TripPlan): Cấu trúc dữ liệu chính cho lịch trình 3 ngày 2 đêm, bao gồm: thời lượng (duration), phong cách du lịch (style), danh sách khách sạn (hotels), danh sách xe khách (buses), danh sách chuyến bay (flights), danh sách điểm tham quan thêm (extras), lịch trình theo ngày (schedule), và thông tin chuẩn bị (prep).

Thông tin chi tiết theo ngày (Day Schedule): Mô tả lịch trình cụ thể cho mỗi ngày, phân chia theo các khung thời gian: buổi sáng (morning), buổi trưa (noon), buổi chiều

(afternoon), và buổi tối (evening).

Mỗi lịch trình bao gồm ba ngày, tương ứng với lịch trình du lịch 3 ngày 2 đêm. Trong mỗi ngày, dữ liệu được chia thành các hoạt động cụ thể theo từng khung thời gian, giúp hệ thống dễ dàng quản lý và hiển thị. Việc thiết kế dữ liệu theo cấu trúc rõ ràng không chỉ giúp hệ thống hoạt động ổn định mà còn tạo điều kiện thuận lợi cho việc mở rộng chức năng trong tương lai.

2.7.4 Thiết kế luồng hoạt động

Luồng hoạt động của hệ thống được xây dựng theo hướng đơn giản và trực quan, phù hợp với thói quen sử dụng ứng dụng di động của người dùng.

- **Luồng khởi động và xác thực**: Khi người dùng mở ứng dụng, hệ thống kiểm tra trạng thái đăng nhập thông qua `Auth Gate`. Nếu chưa đăng nhập, ứng dụng chuyển hướng đến màn hình đăng nhập (`Login Screen`). Người dùng có thể đăng nhập với tài khoản có sẵn hoặc chuyển sang màn hình đăng ký (`Register Screen`) để tạo tài khoản mới.
- **Luồng chính sau đăng nhập**: Sau khi xác thực thành công, người dùng được chuyển đến màn hình chính (`Home Screen`) hiển thị danh sách các thành phố du lịch. Tại đây, người dùng có thể lựa chọn thành phố muốn xem lịch trình chi tiết.
- **Luồng xem chi tiết lịch trình**: Khi chọn một thành phố, hệ thống chuyển đến màn hình chi tiết (`Trip Detail Screen`) với 3 tab thông tin: Tổng quan, Cụ thể, và Chuẩn bị. Trong tab Tổng quan, người dùng có thể nhấn vào các icon để xem thông tin chi tiết về khách sạn, xe khách, máy bay thông qua `BottomSheet`.
- **Luồng quay lại và đăng xuất**: Trong quá trình sử dụng, người dùng có thể quay lại màn hình trước bằng nút back của thiết bị hoặc nút back trên app bar. Để đăng xuất, người dùng nhấn nút đăng xuất trên màn hình chính, hệ thống sẽ xóa phiên đăng nhập và chuyển hướng về màn hình đăng nhập.

2.7.5 Đánh giá thiết kế hệ thống

Thiết kế hệ thống của ứng dụng đáp ứng đầy đủ các yêu cầu chức năng và phi chức năng đã đề ra trong giai đoạn phân tích. Cấu trúc hệ thống rõ ràng, giao diện thân thiện và dễ sử dụng đối với người dùng phổ thông.

Ưu điểm về kiến trúc: Ứng dụng sử dụng hiệu quả các thành phần của Flutter như StatefulWidget cho các màn hình cần thay đổi trạng thái (đăng nhập, đăng ký), kết hợp với phương thức *setState()* để cập nhật giao diện khi dữ liệu thay đổi.

Quản lý trạng thái hiệu quả: Hệ thống sử dụng *Shared Preferences* để lưu trữ session người dùng một cách an toàn và *Future Builder* để xử lý các tác vụ bất đồng bộ trong quá trình xác thực, đảm bảo giao diện không bị đơ khi chờ dữ liệu.

Tính module hóa: Cấu trúc mã nguồn được tổ chức theo từng màn hình và chức năng riêng biệt, giúp việc bảo trì và nâng cấp trở nên dễ dàng. Các thành phần giao diện được tái sử dụng cao thông qua các widget chuyên biệt.

Khả năng mở rộng: Mặc dù ứng dụng chưa tích hợp các chức năng nâng cao như bản đồ trực tuyến hoặc gợi ý thông minh dựa trên trí tuệ nhân tạo, thiết kế hiện tại vẫn phù hợp với mục tiêu học tập của đề tài. Đồng thời, cấu trúc hệ thống tạo nền tảng vững chắc cho việc mở rộng và nâng cấp ứng dụng trong các nghiên cứu và phát triển sau này.

2.7.6 Các bước thực hiện thuật toán gợi ý lịch trình

Thuật toán gợi ý lịch trình du lịch 3 ngày 2 đêm được thực hiện theo các bước sau:

- Bước 1: Hệ thống tiếp nhận dữ liệu đầu vào từ người dùng, bao gồm điểm đến, ngân sách và sở thích du lịch.
- Bước 2: Dựa trên điểm đến đã chọn, hệ thống truy xuất danh sách các lịch trình mẫu tương ứng từ dữ liệu cục bộ.
- Bước 3: Hệ thống lọc các lịch trình phù hợp với ngân sách và sở thích của người dùng.

- Bước 4: Lịch trình được phân chia thành ba ngày, mỗi ngày bao gồm các hoạt động chính theo từng khung thời gian.
- Bước 5: Hệ thống tổng hợp dữ liệu và hiển thị lịch trình hoàn chỉnh cho người dùng.

Mặc dù chưa áp dụng các thuật toán gợi ý thông minh hay học máy, phương pháp này vẫn đáp ứng tốt yêu cầu của đề tài và có thể mở rộng trong tương lai để nâng cao độ chính xác và tính cá nhân hóa.

2.8 Cơ chế hoạt động ứng dụng

Bảng dưới đây mô tả các chức năng chính của ứng dụng GoGo.

Bảng 2.1: Bảng mô tả chức năng hệ thống GoGo

STT	Chức năng	Mô tả
1	Đăng nhập	Nhập email và mật khẩu để xác thực. Kiểm tra dữ liệu và cấp quyền truy cập nếu hợp lệ.
2	Đăng ký tài khoản	Tạo tài khoản mới bằng email, mật khẩu và xác nhận. Kiểm tra định dạng và độ mạnh của mật khẩu.
3	Quên mật khẩu	Nhập email để nhận hướng dẫn khôi phục (demo). Kiểm tra sự tồn tại của email trong hệ thống.
4	Đăng xuất	Kết thúc phiên làm việc, xóa thông tin đăng nhập và chuyển về màn hình đăng nhập.
5	Danh sách thành phố	Hiển thị 3 thành phố mẫu dưới dạng card: Phan Thiết, Đà Lạt, Nha Trang.
6	Chi tiết lịch trình	Giao diện gồm 3 tab: Tổng quan, Cụ thể, Chuẩn bị. Hiển thị lịch trình 3 ngày 2 đêm.
7	Thông tin khách sạn	Hiển thị qua BottomSheet: tên, giá, khu vực và liên kết đặt phòng.
8	Thông tin phương tiện	Hiển thị xe khách và máy bay qua BottomSheet: hãng, giá vé, tuyến đường.
9	Điểm tham quan thêm	Hiển thị các địa điểm bổ sung ngoài lịch trình chính, kèm mô tả ngắn.
10	Quản lý phiên đăng nhập	Duy trì trạng thái đăng nhập giữa các lần mở ứng dụng bằng SharedPreferences.
11	Chỉnh sửa lịch trình	Thay đổi thời lượng, phong cách, hoạt động, checklist và lưu vào SQLite.
12	Lịch trình theo ngày	Trình bày theo khung giờ (sáng, trưa, chiều, tối) bằng các card minh họa.
13	Checklist chuẩn bị	Hiển thị vật dụng, thời tiết, chi phí, mẹo du lịch. Cho phép chỉnh sửa và lưu.
14	Chia sẻ lịch trình (demo)	Giao diện có nút chia sẻ giúp hình dung cách gửi kế hoạch cho người khác.

CHƯƠNG 3: XÂY DỰNG VÀ TRIỂN KHAI ỨNG DỤNG

3.1 Cấu trúc thư mục và tổ chức mã nguồn

3.1.1 Cấu trúc thư mục

Ứng dụng GoGo được xây dựng tuân thủ theo cấu trúc dự án chuẩn của Flutter, nhằm đảm bảo tính khoa học, rõ ràng và dễ dàng bảo trì. Mã nguồn chính của ứng dụng được tập trung trong thư mục `lib`, đóng vai trò là trung tâm chứa đựng toàn bộ logic nghiệp vụ và thành phần giao diện.

Bên trong thư mục `lib`, hệ thống được tổ chức thành các thư mục con chuyên biệt, mỗi thư mục đảm nhiệm một nhóm chức năng cụ thể, tuân theo nguyên tắc tách biệt trách nhiệm (Separation of Concerns). Cấu trúc chi tiết được mô tả như sau:

- Thư mục `screens/` chứa các tệp định nghĩa giao diện chính của từng màn hình, như màn hình nhập liệu, màn hình hiển thị kết quả và màn hình chi tiết.
- Thư mục `models/` bao gồm các lớp đối tượng dữ liệu (DTO - Data Transfer Objects) dùng để biểu diễn thông tin như Điểm đến, Lịch trình và Hoạt động.
- Thư mục `services` tập trung các lớp xử lý logic nghiệp vụ cốt lõi, đặc biệt là lớp chứa thuật toán gợi ý và xử lý lọc lịch trình.
- Thư mục `utils/` cung cấp các hàm tiện ích, hằng số và công cụ hỗ trợ được sử dụng xuyên suốt ứng dụng.
- Thư mục `widgets/` chứa các thành phần giao diện con có khả năng tái sử dụng cao, giúp nhất quán về mặt hình ảnh và giảm thiểu sự trùng lặp mã nguồn.
- Tệp `main.dart` đóng vai trò là điểm khởi động của ứng dụng, chịu trách nhiệm khởi tạo, cấu hình và thiết lập các luồng điều hướng cơ bản.

3.1.2 Nguyên tắc tổ chức mã nguồn

Mã nguồn được tổ chức dựa trên các nguyên tắc sau để đảm bảo tính rõ ràng, dễ bảo trì và mở rộng:

- **Tách biệt Model-View:** Các lớp `Model` (trong thư mục `models/`) đại diện cho cấu trúc dữ liệu (Điểm đến, Lịch trình, Hoạt động). Các `Screen` và `Widget` (trong thư mục `screens/`, `widgets/`) chỉ chịu trách nhiệm hiển thị.
- **Đóng gói logic nghiệp vụ:** Toàn bộ thuật toán gợi ý và xử lý dữ liệu được đặt trong các lớp `Service` (trong thư mục `services/`), giúp mã giao diện gọn nhẹ và dễ kiểm thử.
- **Tái sử dụng Widget:** Các thành phần giao diện được sử dụng nhiều lần (như slider chọn ngân sách, thẻ hiển thị một ngày) được tách thành các `Widget` độc lập trong thư mục `widgets/`.
- **Quản lý phụ thuộc rõ ràng:** Tất cả các thư viện bên ngoài (package) được khai báo tập trung trong tệp `pubspec.yaml`.

3.2 Xây dựng giao diện và xử lý dữ liệu nhập liệu

3.2.1 Giao diện màn hình nhập thông tin

Màn hình nhập thông tin đóng vai trò là điểm tiếp xúc đầu tiên giữa người dùng và hệ thống, được thiết kế với triết lý **đơn giản, trực quan và thân thiện**. Mục tiêu của giao diện này là cho phép người dùng khai báo các yêu cầu cốt lõi cho chuyến đi một cách nhanh chóng và không gặp trở ngại.

Giao diện được xây dựng bằng các widget chuẩn của Flutter theo nguyên tắc Material Design, bao gồm:

- **DropdownButtonFormField:** Cho phép người dùng lựa chọn điểm đến từ danh sách các địa điểm du lịch phổ biến (ví dụ: Đà Lạt, Nha Trang, Phú Quốc).
- **Slider kết hợp với Text:** Cho phép người dùng thiết lập ngân sách dự kiến thông

qua thao tác kéo thả trực quan. Giá trị ngân sách được hiển thị động và định dạng tiền tệ rõ ràng.

- **FilterChip/Wrap:** Hiển thị danh sách các sở thích du lịch (ví dụ: "Ẩm thực", "Thiên nhiên", "Lịch sử văn hóa") dưới dạng các chip có thể chọn/bỏ chọn, tạo trải nghiệm nhập liệu linh hoạt.
- **ElevatedButton:** Nút hành động "Gợi ý lịch trình" được thiết kế nổi bật, chỉ được kích hoạt khi người dùng đã cung cấp đầy đủ thông tin bắt buộc.

Bố cục màn hình sử dụng `Column`, `Padding` và `SizedBox` để tạo không gian âm hợp lý, đảm bảo nội dung không bị dồn nén và dễ dàng tương tác trên nhiều kích thước màn hình khác nhau.

3.2.2 Xử lý dữ liệu nhập liệu

Quy trình xử lý dữ liệu từ giao diện được thực hiện một cách có hệ thống để đảm bảo tính toàn vẹn và sẵn sàng cho bước gợi ý.

1. **Thu thập dữ liệu:** Khi người dùng nhấn nút xác nhận, các controller tương ứng (như `TextEditingController` cho ô nhập, `ValueNotifier` cho Slider) sẽ thu thập giá trị hiện tại.
2. **Validation (Kiểm tra hợp lệ):** Dữ liệu được kiểm tra ngay lập tức thông qua thuộc tính `validator` của `FormField` hoặc logic tùy chỉnh. Các trường bắt buộc (điểm đến, ngân sách) phải được điền đầy đủ trước khi tiếp tục.
3. **Đóng gói dữ liệu:** Các giá trị hợp lệ được đóng gói vào một đối tượng DTO (Data Transfer Object) có tên `UserPreference`, chứa các thuộc tính: `destination: String`, `budget: double`, `preferences: List<String>`.
4. **Lưu trữ tạm thời và truyền dữ liệu:** Đối tượng `UserPreference` được lưu trong trạng thái (`State`) của widget và được truyền sang màn hình kết quả thông qua phương thức `Navigator.pushNamed()` kèm theo `arguments`.

3.3 Quản lý dữ liệu cục bộ trong ứng dụng

3.3.1 Sử dụng SQLite và package sqflite

Trong ứng dụng, cơ sở dữ liệu cục bộ được xây dựng dựa trên SQLite nhằm lưu trữ thông tin các lịch trình du lịch do người dùng tạo và chỉnh sửa. Flutter hỗ trợ làm việc với SQLite thông qua package `sqflite`, cho phép thực hiện các thao tác truy vấn dữ liệu một cách hiệu quả trên thiết bị di động.

Việc sử dụng SQLite giúp ứng dụng hoạt động ổn định ngay cả khi không có kết nối Internet, đồng thời phù hợp với mục tiêu học tập và phạm vi của đề tài.

3.3.2 Cấu trúc cơ sở dữ liệu

Cơ sở dữ liệu được lưu trữ trong file `travel_guides.db`, bao gồm bảng lưu thông tin lịch trình du lịch. Mỗi bản ghi trong bảng đại diện cho một lịch trình hoàn chỉnh, chứa các thông tin như tên lịch trình, điểm đến, thời gian, danh sách hoạt động và các thông tin mở rộng khác.

Cấu trúc cơ sở dữ liệu được thiết kế đơn giản nhưng đủ để hỗ trợ các chức năng thêm, sửa, xóa và truy vấn lịch trình trong ứng dụng.

3.3.3 Áp dụng Repository Pattern

Ứng dụng áp dụng Repository Pattern để tách biệt logic truy cập dữ liệu khỏi các thành phần giao diện. Lớp Repository đóng vai trò trung gian giữa cơ sở dữ liệu SQLite và các màn hình giao diện, giúp mã nguồn rõ ràng và dễ bảo trì.

Cách tiếp cận này giúp các thao tác với cơ sở dữ liệu như thêm, cập nhật, xóa và truy vấn dữ liệu được quản lý tập trung, đồng thời giảm sự phụ thuộc giữa các thành phần trong hệ thống.

3.3.4 JSON Serialization

Dữ liệu lịch trình được ánh xạ giữa đối tượng trong Dart và dữ liệu lưu trữ thông qua các phương thức `toJson()` và `fromJson()`. Việc chuyển đổi này giúp dữ liệu được lưu trữ và truy xuất một cách linh hoạt, đồng thời tạo tiền đề cho việc mở rộng ứng dụng sang các hệ thống lưu trữ hoặc API trong tương lai.

3.3.5 Các thao tác CRUD

Hệ thống hỗ trợ đầy đủ các thao tác CRUD (Create, Read, Update, Delete), bao gồm:

- Tạo mới lịch trình du lịch
- Đọc và hiển thị danh sách lịch trình
- Cập nhật nội dung lịch trình đã tồn tại
- Xóa lịch trình theo yêu cầu người dùng

3.4 Cài đặt thuật toán gợi ý lịch trình

3.4.1 Mô tả thuật toán

Thuật toán gợi ý lịch trình du lịch 3 ngày 2 đêm trong đề tài được xây dựng theo hướng đơn giản và trực quan, dựa trên các lịch trình mẫu đã được chuẩn bị sẵn từ trước. Cách tiếp cận này giúp giảm độ phức tạp của hệ thống, đồng thời đảm bảo thuật toán dễ triển khai và phù hợp với mục tiêu học tập của môn học.

Thuật toán sử dụng các thông tin đầu vào do người dùng cung cấp, bao gồm điểm đến và ngân sách dự kiến cho chuyến đi. Dựa trên các tiêu chí này, hệ thống sẽ thực hiện việc đối chiếu với tập các lịch trình mẫu, từ đó lựa chọn ra lịch trình phù hợp nhất. Trong trường hợp có nhiều lịch trình thỏa mãn điều kiện, thuật toán sẽ ưu tiên lịch trình có mức chi phí gần với ngân sách của người dùng.

Sau khi lựa chọn được lịch trình phù hợp, dữ liệu lịch trình sẽ được tổ chức lại và

chia thành ba ngày tương ứng với thời gian chuyến đi. Mỗi ngày bao gồm các hoạt động chính như tham quan, ăn uống và nghỉ ngơi, được sắp xếp theo trình tự hợp lý nhằm giúp người dùng dễ theo dõi và sử dụng. Cách xây dựng thuật toán này đảm bảo ứng dụng hoạt động ổn định và mang lại trải nghiệm nhất quán cho người dùng.

3.4.2 Sơ đồ và luồng xử lý thuật toán

Luồng xử lý của thuật toán gợi ý lịch trình bắt đầu từ việc tiếp nhận dữ liệu đầu vào của người dùng thông qua giao diện nhập thông tin. Sau khi dữ liệu được thu thập, hệ thống tiến hành kiểm tra tính hợp lệ của các thông tin nhằm đảm bảo dữ liệu đầu vào đáp ứng các yêu cầu cơ bản.

Tiếp theo, thuật toán thực hiện quá trình tìm kiếm và lọc các lịch trình mẫu dựa trên các tiêu chí đã xác định. Các lịch trình không phù hợp sẽ bị loại bỏ, trong khi các lịch trình thỏa mãn điều kiện sẽ được đưa vào danh sách lựa chọn. Từ danh sách này, hệ thống chọn ra một lịch trình tối ưu để hiển thị cho người dùng.

Kết quả cuối cùng của thuật toán là một lịch trình du lịch hoàn chỉnh 3 ngày 2 đêm, được trình bày rõ ràng theo từng ngày và từng hoạt động cụ thể. Việc xây dựng luồng xử lý rõ ràng và logic giúp thuật toán dễ hiểu, dễ triển khai, đồng thời tạo nền tảng thuận lợi cho việc mở rộng và nâng cấp trong tương lai, chẳng hạn như áp dụng

3.5 Quản lý trạng thái và điều hướng trong ứng dụng

Quản lý trạng thái là một trong những yếu tố quan trọng ảnh hưởng trực tiếp đến khả năng tương tác và tính ổn định của ứng dụng. Trong đề tài, việc quản lý trạng thái được thực hiện bằng cách sử dụng *StatefulWidget* kết hợp với hàm *setState*. Trạng thái của ứng dụng bao gồm các thông tin người dùng nhập vào, các lựa chọn về điểm đến, ngân sách cũng như kết quả lịch trình du lịch được hệ thống gợi ý.

Khi trạng thái của ứng dụng thay đổi, Flutter sẽ tự động cập nhật lại giao diện để phản ánh dữ liệu mới. Cơ chế này giúp đảm bảo dữ liệu hiển thị trên giao diện luôn đồng bộ với dữ liệu xử lý bên trong ứng dụng, từ đó nâng cao trải nghiệm người dùng và hạn

chế các lỗi không mong muốn.

Chức năng điều hướng giữa các màn hình được triển khai thông qua cơ chế Navigator của Flutter. Khi người dùng thực hiện các thao tác như nhấn nút xác nhận, quay lại hoặc xem chi tiết lịch trình, ứng dụng sẽ chuyển sang màn hình tương ứng và truyền các dữ liệu cần thiết. Việc kết hợp chặt chẽ giữa quản lý trạng thái và điều hướng giúp luồng hoạt động của ứng dụng diễn ra mạch lạc, dễ sử dụng và ổn định trong quá trình vận hành.

3.6 Lưu trữ và xử lý dữ liệu trong ứng dụng

3.6.1 Cấu trúc và phương thức lưu trữ

Hệ thống quản lý dữ liệu của ứng dụng được thiết kế theo ba cấp độ lưu trữ khác nhau, tương ứng với tính chất và yêu cầu sử dụng của từng loại dữ liệu. Cách tiếp cận này đảm bảo sự cân bằng giữa hiệu suất, tính tiện dụng và độ phức tạp của hệ thống.

- **Dữ liệu tĩnh/mẫu:** Các thông tin cố định như danh sách điểm đến, lịch trình mẫu và hoạt động du lịch được lưu trữ dưới dạng tệp `JSON` trong thư mục `assets/data/`. Dữ liệu được tải một lần khi ứng dụng khởi động thông qua phương thức `rootBundle.loadStr` và chuyển đổi thành đối tượng Dart để sử dụng trong suốt phiên làm việc. Phương pháp này giúp ứng dụng hoạt động độc lập, không cần kết nối mạng và đảm bảo tốc độ truy xuất nhanh chóng.
- **Dữ liệu người dùng tạm thời:** Trong quá trình tương tác, các thông tin do người dùng nhập vào (điểm đến, ngân sách, sở thích) được lưu giữ trong bộ nhớ RAM dưới dạng các đối tượng Dart. Dữ liệu này tồn tại xuyên suốt phiên làm việc hiện tại, được truyền giữa các màn hình và sử dụng làm đầu vào cho thuật toán gợi ý. Cách lưu trữ tạm thời này tối ưu hiệu suất xử lý và phản hồi tức thì cho người dùng.
- **Lưu trữ cục bộ đơn giản:** Để nâng cao trải nghiệm người dùng giữa các lần sử dụng, ứng dụng áp dụng cơ chế lưu trữ bền vững thông qua package `shared_preferences`. Các lựa chọn gần nhất của người dùng (ví dụ: điểm đến ưa thích, mức ngân sách thường dùng) được lưu lại dưới dạng cặp khóa-giá trị. Khi ứng dụng được mở lại,

hệ thống tự động khôi phục các thiết lập này, giúp người dùng tiết kiệm thời gian nhập liệu và cá nhân hóa trải nghiệm sử dụng.

3.6.2 Quy trình xử lý dữ liệu

Quá trình xử lý dữ liệu trong ứng dụng được thực hiện theo một luồng tuần tự và logic, bắt đầu từ việc thu thập thông tin người dùng cho đến khi đưa ra kết quả gợi ý cuối cùng.

1. **Khởi tạo dữ liệu:** Ứng dụng tải các dữ liệu mẫu từ tệp `JSON` trong thư mục `assets/data/` ngay khi khởi động. Đồng thời, hệ thống kiểm tra và khôi phục các thiết lập đã lưu từ `shared_preferences` (nếu có).

2. **Thu thập và xác thực dữ liệu người dùng:** Thông qua giao diện nhập liệu, dữ liệu từ người dùng được thu thập và kiểm tra tính hợp lệ. Các giá trị không đúng định dạng hoặc thiếu thông tin bắt buộc sẽ được yêu cầu nhập lại.

3. **Xử lý và lọc dữ liệu:** Dữ liệu đã xác thực được sử dụng làm đầu vào cho thuật toán gợi ý. Hệ thống thực hiện việc đối chiếu và lọc các lịch trình mẫu dựa trên các tiêu chí như điểm đến, ngân sách và sở thích. Các lịch trình không phù hợp sẽ bị loại bỏ, trong khi các lịch trình thỏa mãn điều kiện sẽ được đưa vào danh sách lựa chọn.

4. **Lưu trữ và truy xuất:** Trong quá trình xử lý, dữ liệu trung gian và kết quả cuối cùng được lưu tạm trong bộ nhớ để đảm bảo tốc độ truy cập. Đồng thời, các lựa chọn quan trọng của người dùng có thể được lưu vào `shared_preferences` để sử dụng cho các phiên làm việc sau.

5. **Hiển thị kết quả:** Lịch trình được chọn cuối cùng được hiển thị chi tiết trên giao diện người dùng, phân chia theo từng ngày và hoạt động cụ thể. Dữ liệu được trình bày rõ ràng, giúp người dùng dễ dàng theo dõi và sử dụng.

Việc tổ chức lưu trữ và xử lý dữ liệu một cách hệ thống không chỉ giúp ứng dụng hoạt động ổn định, chính xác mà còn tạo điều kiện thuận lợi cho việc mở rộng và nâng

cấp hệ thống trong tương lai, chẳng hạn như tích hợp cơ sở dữ liệu trực tuyến hoặc các thuật toán gợi ý phức tạp hơn.

CHƯƠNG 4: KẾT QUẢ VÀ ĐÁNH GIÁ

4.1 Nội dung kiểm thử

Quá trình kiểm thử ứng dụng **GoGo** được thực hiện toàn diện trên ba cấp độ chính nhằm đảm bảo chất lượng sản phẩm từ gốc đến ngọn:

1. **Kiểm thử đơn vị (Unit Testing):** Tập trung vào kiểm tra tính đúng đắn của các thành phần logic nghiệp vụ độc lập, đặc biệt là thuật toán lọc và gợi ý lịch trình trong lớp `ItineraryService`. Các test case được thiết kế để bao phủ các tình huống biên như ngân sách tối đa/tối thiểu, không có lịch trình phù hợp, hoặc danh sách sở thích rỗng.
2. **Kiểm thử giao diện (Widget Testing):** Kiểm tra từng màn hình và widget tùy chỉnh để đảm bảo chúng được xây dựng đúng với các thuộc tính đầu vào khác nhau. Ví dụ: kiểm tra xem `BudgetSlider` có hiển thị đúng giá trị định dạng tiền tệ không, hoặc `DayCard` có hiển thị đúng danh sách hoạt động không.
3. **Kiểm thử tích hợp và thủ công (Integration & Manual Testing):** Đây là bước quan trọng nhất, được thực hiện trực tiếp trên thiết bị Android. Quá trình này mô phỏng hành vi người dùng thực tế, kiểm tra toàn bộ luồng hoạt động từ khi mở ứng dụng, nhập thông tin, nhận gợi ý, xem chi tiết đến điều hướng giữa các màn hình.

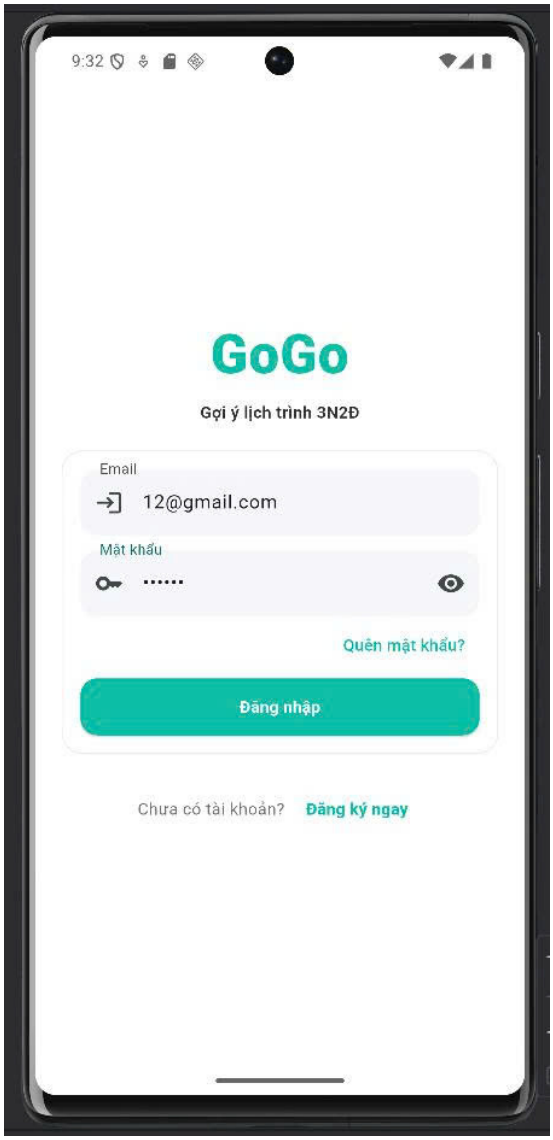
4.2 Kết quả

4.2.1 Kết quả về chức năng và giao diện

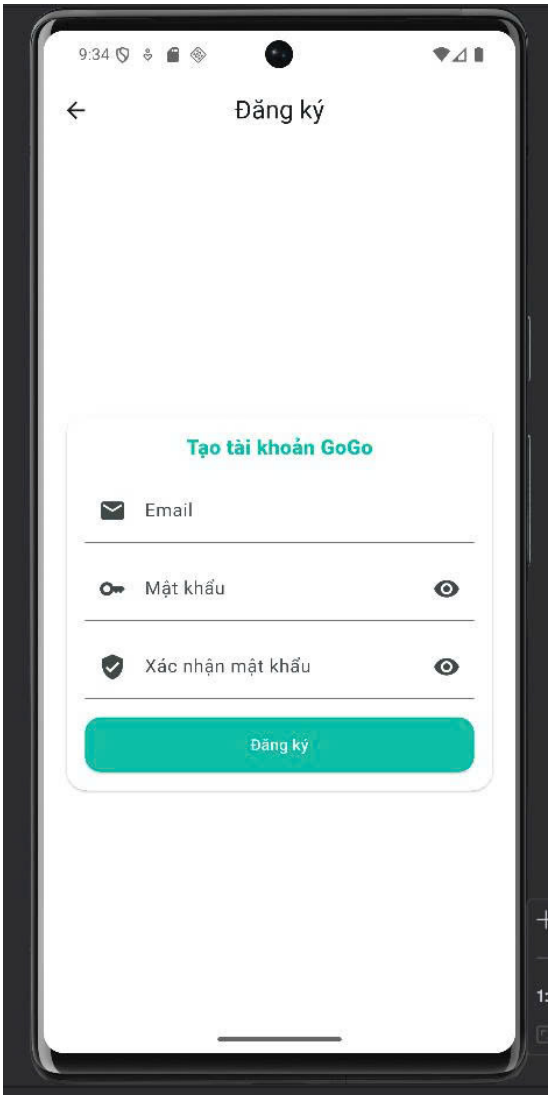
Giao diện ứng dụng được đánh giá cao về tính thẩm mỹ và sự thân thiện. Việc tuân thủ Material Design tạo nên sự nhất quán và quen thuộc cho người dùng Android. Bố cục rõ ràng, khoảng cách hợp lý và màu sắc hài hòa (lấy từ bảng màu chính của ứng dụng) giúp tập trung vào nội dung chính là thông tin lịch trình.

Ứng dụng đã triển khai thành công tất cả các chức năng cốt lõi như đã đề ra trong bảng mô tả chức năng hệ thống (Mục 2.9). Các chức năng chính bao gồm:

Nhập thông tin chuyến đi: Giao diện cho phép lựa chọn điểm đến, thiết lập ngân sách và đánh dấu sở thích một cách trực quan. Dữ liệu được xác thực chặt chẽ trước khi chuyển sang bước xử lý.



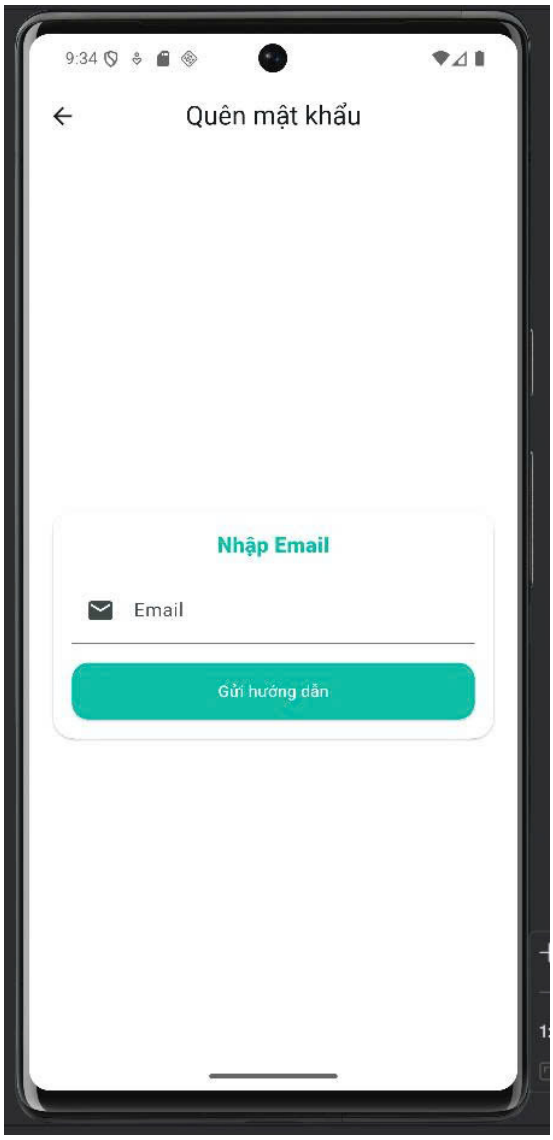
Hình 4.5 Màn hình nhập thông tin người dùng



Hình 4.6 Màn hình đăng ký tài khoản

Hình 4.5 và 4.6 minh họa hai màn hình tiêu biểu trong quá trình sử dụng ứng dụng. Hình 4.5 thể hiện giao diện nhập thông tin người dùng với các thành phần chính như lựa chọn điểm đến, thiết lập ngân sách và sở thích. Hình 4.6 trình bày màn hình đăng ký tài khoản, cho phép người dùng tạo tài khoản để sử dụng các chức năng của ứng dụng.

Hình 4.7 và Hình 4.8 là chức năng quên mật khẩu hiển thị biểu mẫu nhập email và gửi hướng dẫn khôi phục. Sau khi đăng nhập, người dùng được chuyển đến màn hình

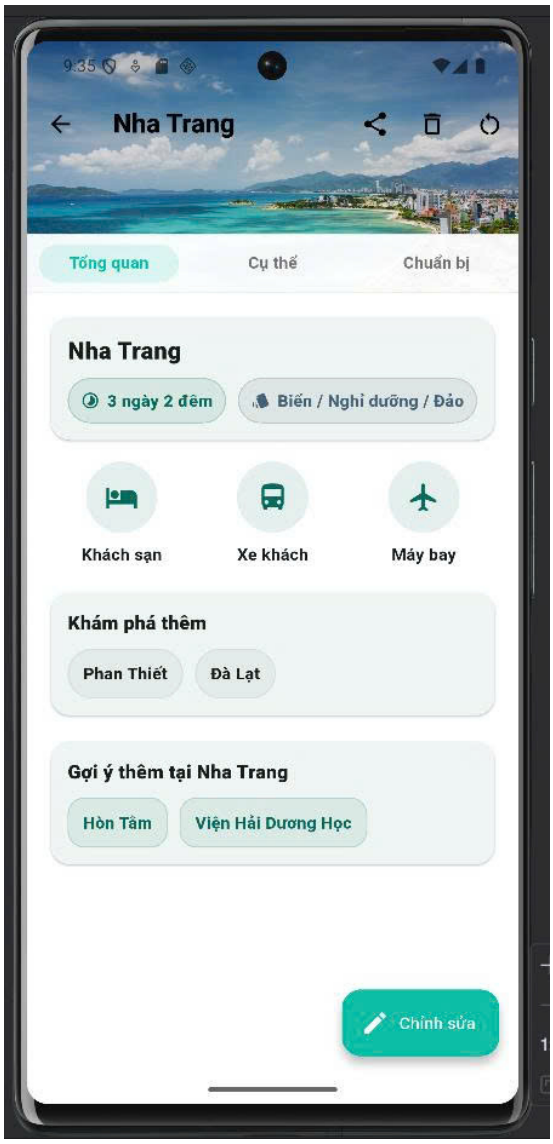


Hình 4.7 Màn hình quên mật khẩu

Hình 4.8 Màn hình chọn điểm đến

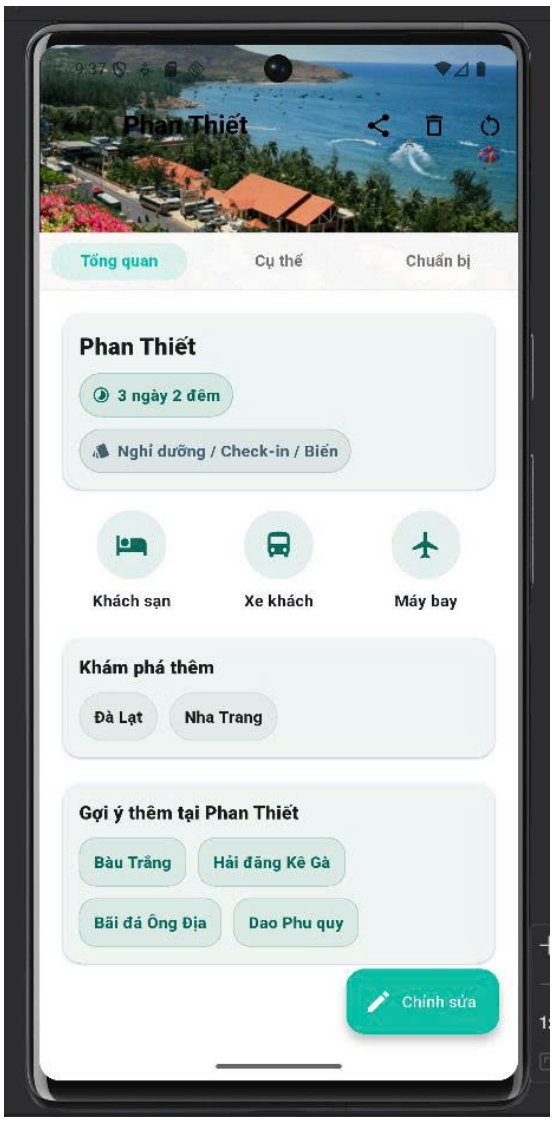
chính hiển thị danh sách thành phố du lịch.

Gợi ý lịch trình tự động: Thuật toán lọc và khớp hoạt động ổn định, trả về lịch trình phù hợp nhất trong thời gian dưới 1 giây.

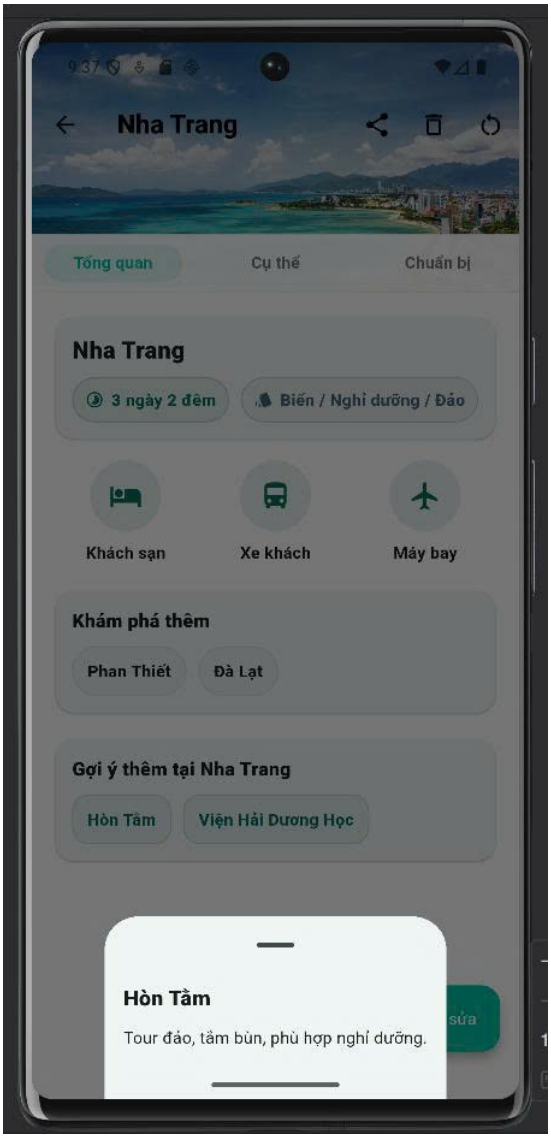


Hình 4.9 Màn hình chi tiết lịch trình Nha Trang

Hình 4.10 Màn hình chi tiết lịch trình Đà Lạt



Hình 4.11 Màn hình chi tiết lịch trình Phan Thiết



Hình 4.12 Màn hình gợi ý

Mỗi thành phố có lịch trình mặc định theo phong cách riêng. Người dùng có thể xem thông tin tổng quan, phương tiện di chuyển, và các địa điểm gợi ý thêm.

Hiển thị và điều hướng: Ứng dụng cung cấp cả hai chế độ xem: tổng quan 3 ngày và chi tiết từng ngày. Việc chuyển đổi giữa các màn hình diễn ra mượt mà, không bị gián đoạn.

Hình 4.13 Màn hình chi tiết lịch trình Nha Trang

Hình 4.14 Màn hình chi tiết lịch trình Đà Lạt

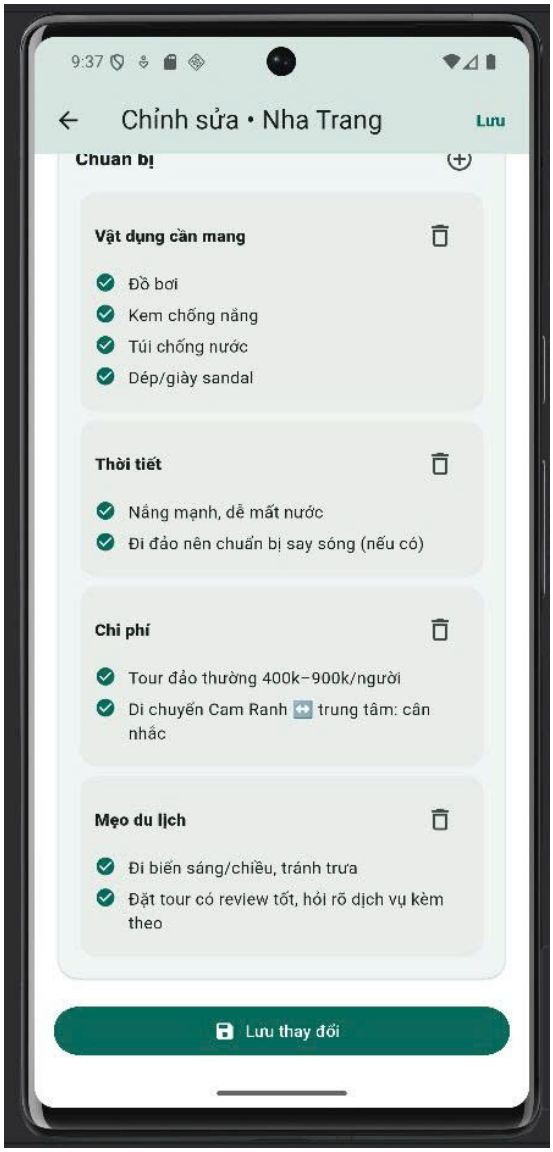
Hình 4.15 Màn hình chi tiết lịch trình Phan Thiết

Hình 4.13, hình 4.14 và Hình 4.15 minh họa giao diện chi tiết của một ngày trong lịch trình du lịch được gợi ý. Màn hình này được thiết kế với bố cục rõ ràng theo trục dọc, hiển thị tuần tự các hoạt động từ sáng đến tối. Mỗi hoạt động được trình bày trong một `Card` riêng biệt, bao gồm các thông tin cốt lõi: khung giờ cụ thể, tên hoạt động chính, mô tả ngắn gọn và biểu tượng minh họa phù hợp. Việc sử dụng màu sắc nhẹ nhàng, khoảng cách hợp lý giữa các card và font chữ dễ đọc giúp người dùng nhanh chóng nắm bắt lộ trình trong ngày, từ đó dễ dàng lên kế hoạch di chuyển và chuẩn bị cho chuyến đi thực tế.

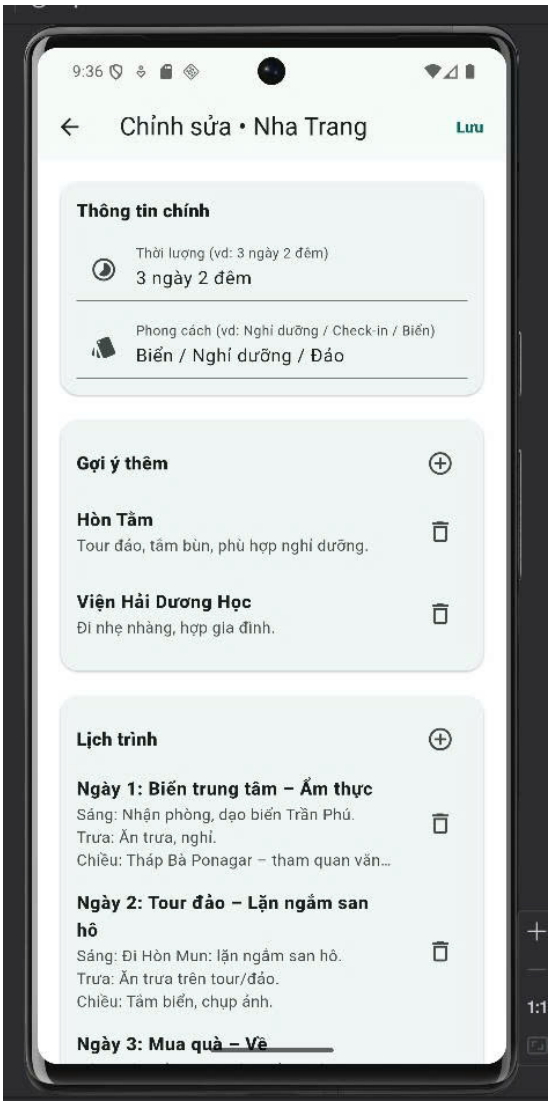
Chia sẻ lịch trình: ứng dụng hỗ trợ chức năng chia sẻ lịch trình, cho phép người

dùng gửi nội dung lịch trình du lịch đến các ứng dụng khác trên thiết bị như tin nhắn, email hoặc các nền tảng mạng xã hội. Chức năng này giúp người dùng dễ dàng chia sẻ kế hoạch du lịch với bạn bè và người thân, đồng thời tăng tính tương tác và khả năng sử dụng thực tế của ứng dụng.

Xác nhận chỉnh sửa dữ liệu: là khi người dùng thực hiện thao tác xóa một lịch trình đã lưu, hệ thống sẽ hiển thị hộp thoại xác nhận nhằm hạn chế việc xóa nhầm dữ liệu quan trọng. Chỉ khi người dùng đồng ý xác nhận, lịch trình mới được xóa khỏi cơ sở dữ liệu cục bộ. Cơ chế này góp phần đảm bảo an toàn dữ liệu và nâng cao độ tin cậy của ứng dụng.



Hình 4.16 Màn hình chỉnh sửa lịch trình Nha Trang



Hình 4.17 Màn hình chỉnh sửa lịch trình Nha Trang đã lưu

Ngoài ra, ứng dụng còn cung cấp chức năng khôi phục lịch trình mặc định, cho phép người dùng đưa lịch trình trở về trạng thái ban đầu trong trường hợp cần thiết. Chức năng này đặc biệt hữu ích khi người dùng đã chỉnh sửa nhiều nội dung và muốn quay lại cấu hình gốc của hệ thống.

Hình 4.18 Màn hình checklist chuẩn bị Đà Lạt

Hình 4.19 Màn hình checklist chuẩn bị Nha Trang

Hình 4.20 Màn hình checklist chuẩn bị Phan Thiết

Checklist hiển thị các vật dụng cần mang, thời tiết, chi phí và mẹo du lịch. Người dùng có thể chỉnh sửa và lưu lại theo từng điểm đến.

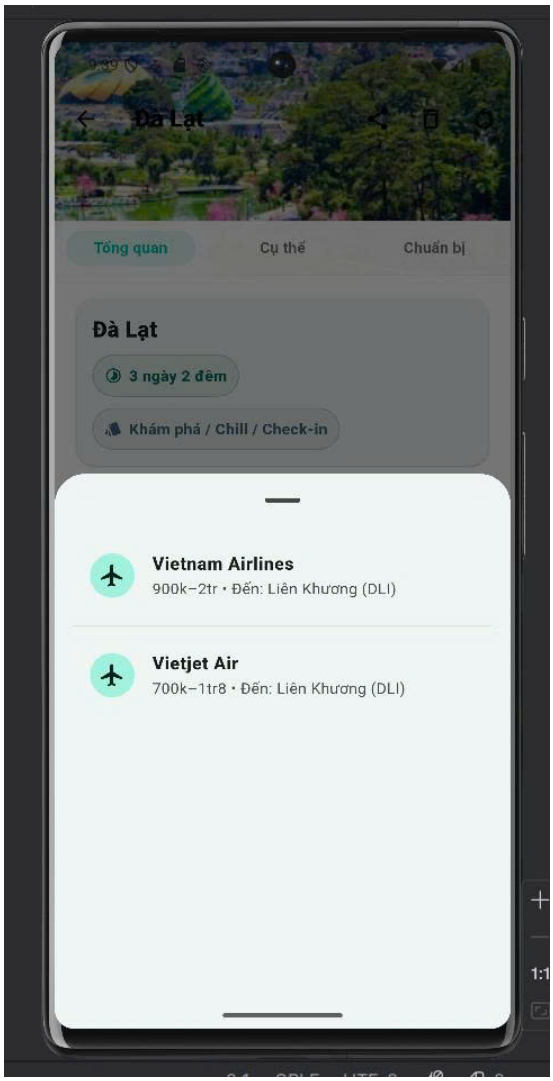
4.2.2 Kết quả về trải nghiệm người dùng

Trải nghiệm người dùng được tối ưu qua các điểm sau:

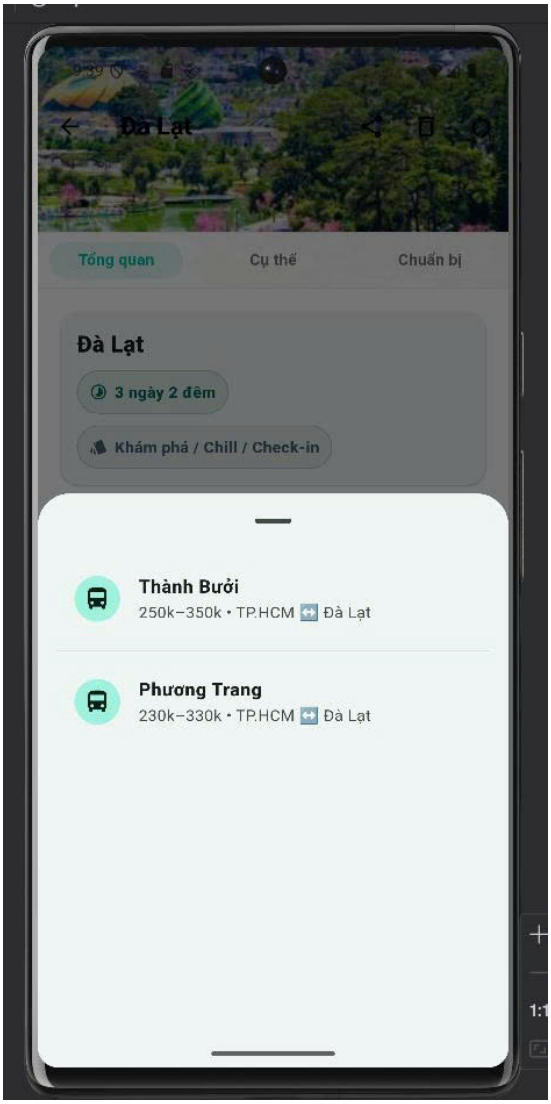
- **Luồng thao tác đơn giản:** Người dùng chỉ cần thực hiện ba bước chính: chọn điểm đến, nhập ngân sách dự kiến, và nhận gợi ý lịch trình phù hợp.
- **Phản hồi trực quan:** Các nút bấm và thành phần giao diện có hiệu ứng phản hồi rõ ràng; thanh trượt (slider) cho cảm giác điều khiển trực tiếp; thông báo lỗi hiển

thị cụ thể giúp người dùng dễ dàng nhận biết và xử lý.

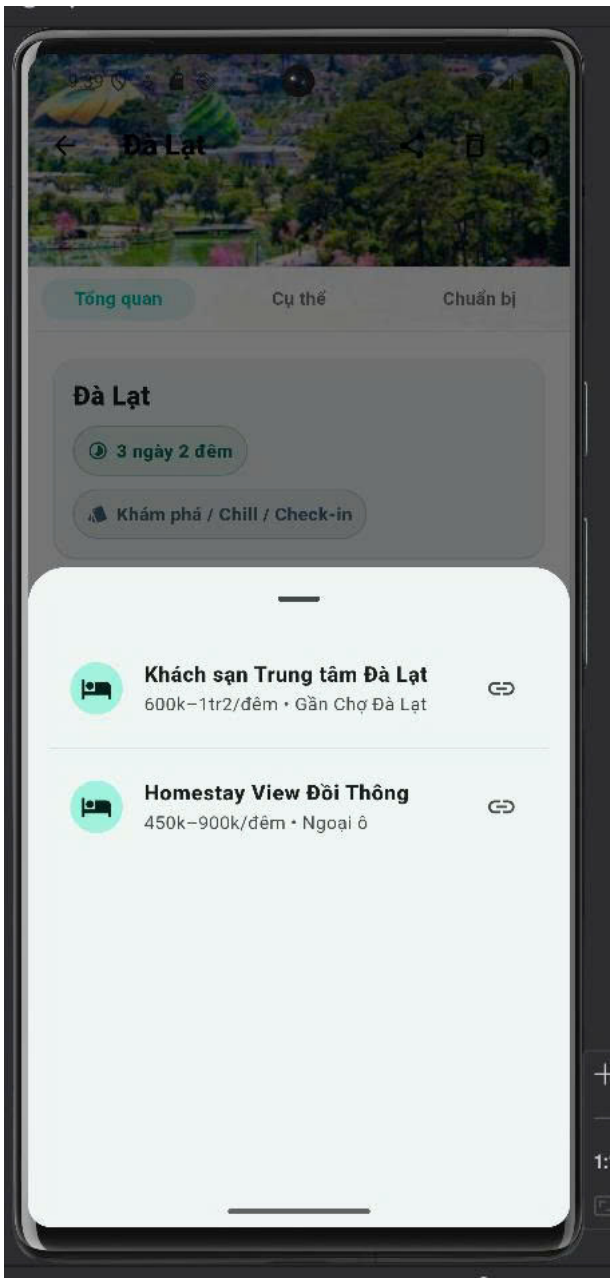
- **Tính cá nhân hóa cơ bản:** Ứng dụng sử dụng `SharedPreferences` để lưu lại lựa chọn trước đó, giúp người dùng tiết kiệm thời gian và mang lại trải nghiệm thuận tiện hơn khi quay lại sử dụng.



Hình 4.21 Bottom sheet hiển thị thông tin chuyến đi Đà Lạt



Hình 4.22 Bottom sheet hiển thị thông tin chuyến đi



Hình 4.23 Bottom sheet thông tin chuyến đi Đà Lạt

kết quả giao diện bottom sheet hiển thị thông tin phương tiện di chuyển cho từng điểm đến. Các thành phần như khách sạn, xe khách và máy bay được trình bày rõ ràng, có biểu tượng minh họa và mức giá tham khảo. Thiết kế bottom sheet giúp người dùng dễ dàng tiếp cận thông tin cần thiết mà không làm gián đoạn trải nghiệm chính. Việc bố trí hợp lý, font chữ dễ đọc và cách hiển thị trực quan giúp người dùng nhanh chóng đưa ra lựa chọn phù hợp cho chuyến đi.

4.3 Thảo luận kết quả

4.3.1 Đánh giá ưu điểm

Ứng dụng GoGo đã đạt được những ưu điểm nổi bật sau:

- **Đáp ứng đúng mục tiêu học tập:** Ứng dụng hoàn thành xuất sắc nhiệm vụ là một sản phẩm minh họa cho việc áp dụng Flutter/Dart vào một bài toán thực tế, tích hợp đầy đủ các kiến thức về widget, điều hướng, quản lý trạng thái và xử lý dữ liệu.
- **Kiến trúc mã nguồn rõ ràng:** Việc áp dụng mô hình phân tách thư mục (screens, models, services, widgets) giúp mã nguồn dễ đọc, dễ bảo trì và là nền tảng tốt cho việc mở rộng.
- **Giao diện thân thiện và chuyên nghiệp:** Ứng dụng không chỉ hoạt động tốt mà còn có giao diện đẹp mắt, tạo ấn tượng tốt và thể hiện sự đầu tư nghiêm túc vào trải nghiệm người dùng.
- **Tính ổn định cao:** Qua quá trình kiểm thử kỹ lưỡng, ứng dụng không gặp phải lỗi crash (dừng đột ngột) hay hoạt động sai lệch nghiêm trọng nào.

4.3.2 Hạn chế của ứng dụng

Bên cạnh những kết quả đạt được, ứng dụng vẫn còn một số hạn chế cần được ghi nhận và cải thiện trong tương lai:

- **Dữ liệu tĩnh và hạn chế:** Toàn bộ dữ liệu điểm đến và lịch trình mẫu đều được lưu cứng trong ứng dụng. Điều này khiến ứng dụng khó cập nhật thông tin mới, không có tính thời sự và chỉ phù hợp với phạm vi demo.
- **Thuật toán gợi ý đơn giản:** Thuật toán hiện tại chỉ dừng ở mức lọc cơ bản dựa trên quy tắc (rule-based). Nó thiếu khả năng "học" từ hành vi người dùng hoặc đề xuất thông minh hơn dựa trên các yếu tố phức tạp như khoảng cách địa lý, đánh giá, hoặc xu hướng.

- **Thiếu tích hợp dịch vụ bên ngoài:** Ứng dụng chưa kết nối với các API bản đồ (Google Maps), dịch vụ đặt phòng/ vé, hay mạng xã hội để chia sẻ, làm hạn chế tính hữu dụng trong một kịch bản du lịch thực tế.
- **Phạm vi nền tảng:** Hiện tại ứng dụng mới chỉ được phát triển và kiểm thử cho Android. Cần thêm bước build và tối ưu để chạy được trên iOS.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

Trong quá trình thực hiện học phần Lập trình trên thiết bị di động, đề tài đã tập trung nghiên cứu và xây dựng ứng dụng GoGo trên nền tảng Flutter với ngôn ngữ lập trình Dart. Ứng dụng được thiết kế nhằm hỗ trợ người dùng trong việc lập kế hoạch du lịch ngắn ngày thông qua việc cho phép lựa chọn các thông tin cơ bản như điểm đến, ngân sách và sở thích cá nhân. Dựa trên các thông tin này, hệ thống sẽ đưa ra lịch trình du lịch phù hợp và được trình bày chi tiết theo từng ngày, giúp người dùng dễ dàng theo dõi và sử dụng.

Quá trình thực hiện đề tài đã tạo điều kiện cho sinh viên vận dụng tổng hợp các kiến thức và kỹ năng đã được học, bao gồm lập trình ứng dụng di động bằng Flutter, thiết kế giao diện người dùng theo Material Design, quản lý trạng thái ứng dụng và điều hướng giữa các màn hình. Bên cạnh đó, sinh viên còn được rèn luyện kỹ năng phân tích yêu cầu, xây dựng thuật toán gợi ý, thiết kế hệ thống và tổ chức mã nguồn một cách khoa học, góp phần nâng cao tư duy lập trình và khả năng triển khai các ứng dụng thực tế.

Kết quả đạt được cho thấy ứng dụng hoạt động ổn định trên nền tảng Android, các chức năng chính được triển khai đúng theo yêu cầu đề tài và giao diện thân thiện với người dùng. Mặc dù thuật toán gợi ý còn ở mức cơ bản và dữ liệu chủ yếu được lưu trữ cục bộ, ứng dụng vẫn đáp ứng tốt các mục tiêu đề ra ban đầu. Nhìn chung, đề tài đã hoàn thành tốt yêu cầu của học phần Lập trình trên thiết bị di động, đồng thời tạo tiền đề cho việc tiếp tục nghiên cứu và phát triển ứng dụng trong tương lai.

5.2 Hướng phát triển tương lai

Trong thời gian tới, ứng dụng có thể được tiếp tục phát triển và hoàn thiện với nhiều chức năng nâng cao hơn. Trước hết, hệ thống có thể tích hợp các API du lịch để cập nhật dữ liệu địa điểm, khách sạn và nhà hàng một cách phong phú và chính xác hơn.

Ngoài ra, việc kết hợp bản đồ trực tuyến và định vị GPS sẽ giúp người dùng dễ

dàng theo dõi và di chuyển theo lịch trình đã gợi ý. Thuật toán gợi ý cũng có thể được cải tiến bằng cách áp dụng các phương pháp học máy nhằm cá nhân hóa lịch trình theo hành vi và thói quen của người dùng.

Cuối cùng, ứng dụng có thể được mở rộng để hỗ trợ nhiều nền tảng khác như iOS và bổ sung các tính năng quản lý chuyến đi, chia sẻ lịch trình và đánh giá địa điểm. Những hướng phát triển này sẽ góp phần nâng cao giá trị ứng dụng và mở ra tiềm năng triển khai trong thực tế.

TÀI LIỆU THAM KHẢO

[1] Google. *Flutter Documentation*. https://docs.flutter.dev.

[2] Google. *Dart Programming Language Documentation*. https://dart.dev, 2025.

[3] Google. *Material Design 3*. https://m3.material.io, 2025.

[4] Google. *Flutter Cookbook*. https://docs.flutter.dev/cookbook, truy cập năm 2025.

[5] Android Developers. *Android Developers Documentation*. https://developer.android.com, truy cập năm 2025.

[6] Ian Sommerville. *Software Engineering*. Pearson Education, 10th Edition, 2016.

[7] Alan Cooper, Robert Reimann, David Cronin, Christopher Noessel. *About Face: The Essentials of Interaction Design*. Wiley Publishing, 4th Edition, 2014.

[8] FilledStacks. *Flutter Architecture Guide*. https://www.filledstacks.com, 2025.

[9] Nguyễn Văn A. *Cơ sở lý thuyết về du lịch và xây dựng lịch trình*. Nhà xuất bản Lao động – Xã hội, 2018.