**ĐẠI HỌC ĐÀ NẴNG**
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA CÔNG NGHỆ THÔNG TIN**

# ĐỒ ÁN TỐT NGHIỆP

## NGÀNH: CÔNG NGHỆ THÔNG TIN

## CHUYÊN NGÀNH: CÔNG NGHỆ PHẦN MỀM

**ĐỀ TÀI:**

# *XÂY DỰNG HỆ THỐNG HỖ TRỢ DU LỊCH CÁ NHÂN VÀ CHIA SẺ TRẢI NGHIỆM DU LỊCH GIỮA NGƯỜI DÙNG*

Người hướng dẫn: **ThS. NGUYỄN CÔNG DANH**
Sinh viên thực hiện: **NGUYỄN VĂN NGHĨA**
Số thẻ sinh viên: **102200026**
Lớp**:** **20T1**

**Đà Nẵng, 6/2026**

**ĐẠI HỌC ĐÀ NẴNG**
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA CÔNG NGHỆ THÔNG TIN**

# ĐỒ ÁN TỐT NGHIỆP

## NGÀNH: CÔNG NGHỆ THÔNG TIN

## CHUYÊN NGÀNH: CÔNG NGHỆ PHẦN MỀM

**ĐỀ TÀI:**

# *XÂY DỰNG HỆ THỐNG HỖ TRỢ DU LỊCH CÁ NHÂN VÀ CHIA SẺ TRẢI NGHIỆM DU LỊCH GIỮA NGƯỜI DÙNG*

Người hướng dẫn: **ThS. NGUYỄN CÔNG DANH**
Sinh viên thực hiện: **NGUYỄN VĂN NGHĨA**
Số thẻ sinh viên: **102200026**
Lớp**:** **20T1**

**Đà Nẵng, 6/2026**

# TÓM TẮT

Tên đề tài: Xây dựng hệ thống hỗ trợ du lịch cá nhân và chia sẽ trải nghiệm du lịch giữa người dùng

Sinh viên thực hiện: Nguyễn Văn Nghĩa

Số thẻ SV: 102200026 Lớp: 20T1

Du lịch tự túc đang trở thành xu hướng chủ đạo tại Việt Nam, song người dùng phải dùng nhiều ứng dụng rời rạc cho các nhu cầu khác nhau: mạng xã hội để tìm cảm hứng, nền tảng đặt phòng, nhóm Facebook để ghép đoàn, và các kênh riêng để tìm hướng dẫn viên địa phương. Đề tài này xây dựng TripMate — nền tảng hợp nhất ba trụ cột trong cùng một sản phẩm: (1) mạng xã hội chia sẻ trải nghiệm du lịch, (2) tổ chức và tham gia chuyến đi nhóm, (3) thị trường đặt hướng dẫn viên địa phương có thanh toán an toàn — chạy đồng thời trên web và Android

***Công nghệ sử dụng:***

- Frontend: React 18, TypeScript, Vite, TailwindCSS, TanStack Query, Zustand, React Router v6;
- Backend: NestJS 10, TypeORM, PostgreSQL 16, Passport JWT, Socket.IO, Swagger;
- Mobile: Capacitor (đóng gói APK Android từ codebase web);
- AI chat bot: Xây dựng trợ lý ảo với RAG
- Tích hợp : SePay (thanh toán webhook), Firebase Storage (ảnh), Leaflet + OpenStreetMap (bản đồ & định tuyến).

***Kết quả đạt được:***

Hoàn thành hệ thống full-stack với 14 module backend, 80+ REST endpoint, 24 trang giao diện, 31 bảng cơ sở dữ liệu, 4 vai trò người dùng, 36 Use Case. Đóng gói và cài đặt thành công APK Android (~5 MB) chạy trên thiết bị thật. Hệ thống đạt điểm Lighthouse Performance 91/100, API response trung bình dưới 200 ms cho luồng CRUD chính. Không phát hiện lỗ hổng bảo mật nghiêm trọng qua kiểm thử OWAS

ĐẠI HỌC ĐÀ NẴNG
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
KHOA CÔNG NGHỆ THÔNG TIN

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**
Độc lập - Tự do - Hạnh phúc

# NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Nguyễn Văn Nghĩa Số thẻ sinh viên: 102200026

Lớp: 20T1 Khoa: Công nghệ thông tin Ngành: Công nghệ thông tin

1. *Tên đề tài đồ án:*

Xây dựng hệ thống hỗ trợ du lịch cá nhân và chia sẽ trải nghiệm du lịch giữa người dùng

2. *Đề tài thuộc diện: ☐ Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu: Không có*
4. *Nội dung các phần thuyết minh và tính toán:*

*Mở đầu : Lý do chọn đề tài, mục tiêu, phạm vi và giới hạn, đối tượng và phương pháp nghiên cứu*

*Chương 1 : Cơ sở Lý thuyết*

*Chương 2 : Phân tích và thiết kế hệ thống*

*Chương 3 : Triển khai và đánh giá*

*Kết luận, tài liệu tham khảo, phụ lục*

5. *Các bản vẽ, đồ thị : Không có*
6. *Họ tên người hướng dẫn: ThS Nguyễn Công Danh*
7. *Ngày giao nhiệm vụ đồ án: ……/……./2026*
8. *Ngày hoàn thành đồ án: ……/……./2026*

*Đà Nẵng, ngày tháng năm 2026*

**Trưởng Khoa CNTT** **Người hướng dẫn**

# LỜI NÓI ĐẦU

Trước hết, em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến quý thầy cô Khoa Công nghệ Thông tin - trường Đại Học Bách Khoa - Đại Học Đà Nẵng – những người đã tận tâm truyền đạt kiến thức nền tảng và chuyên sâu trong suốt quá trình học tập tại trường. Những bài giảng, định hướng và sự động viên từ thầy cô đã tạo nên hành trang vững chắc giúp em hoàn thành đồ án tốt nghiệp này.

Em xin đặc biệt cảm ơn thầy Nguyễn Công Danh đã trực tiếp hướng dẫn, góp ý và đồng hành trong suốt quá trình thực hiện đề tài. Sự chỉ dẫn tận tình, những gợi ý chuyên môn quý báu cùng tinh thần làm việc nghiêm túc của thầy là động lực để em hoàn thiện sản phẩm theo hướng chuyên nghiệp nhất có thể.

Cuối cùng, em xin gửi lời cảm ơn đến gia đình, bạn bè và những người đã đồng hành, hỗ trợ em cả về vật chất lẫn tinh thần để em có điều kiện tốt nhất hoàn thành đồ án.

Mặc dù đã cố gắng hết sức nhưng do thời gian và kiến thức còn hạn chế, đồ án không tránh khỏi những thiếu sót. Em rất mong nhận được những đóng góp từ quý thầy cô và bạn đọc để đề tài được hoàn thiện hơn.

Em xin chân thành cảm ơn!

# CAM ĐOAN

Em xin cam đoan đề tài đồ án tốt nghiệp " Xây dựng hệ thống hỗ trợ du lịch cá nhân và chia sẽ trải nghiệm du lịch giữa người dùng " là công trình nghiên cứu của riêng em dưới sự hướng dẫn của giảng viên ThS. Nguyễn Công Danh

Toàn bộ mã nguồn, tài liệu thiết kế và nội dung trình bày trong báo cáo này đều do em tự thực hiện. Các đoạn trích dẫn, hình ảnh, số liệu tham khảo từ nguồn khác đều được ghi rõ trong phần Tài liệu tham khảo. Em xin chịu hoàn toàn trách nhiệm trước Hội đồng Khoa và Nhà trường nếu có bất kỳ vi phạm nào về tính trung thực của đồ án.

Sinh viên thực hiện

# MỤC LỤC

TÓM TẮT ..... 5
LỜI NÓI ĐẦU ..... i
CAM ĐOAN ..... ii
MỤC LỤC ..... iii
DANH SÁCH CÁC HÌNH VẼ ..... vi
DANH SÁCH CÁC BẢNG ..... ix
DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT ..... x
MỞ ĐẦU ..... 11
1. Lý do chọn đề tài ..... 11
2. Mục tiêu đề tài ..... 12
2.1. Mục tiêu ..... 12
2.2. Mục tiêu cụ thể : ..... 12
3. Phạm vi và giới hạn ..... 12
3.1. Phạm vi thực hiện ..... 12
3.2. Giới hạn : ..... 13
4. Đối tượng nghiên cứu : ..... 13
5. Phương pháp nghiên cứu : ..... 13
CHƯƠNG 1: CƠ SỞ LÝ THUYẾT ..... 14
1.1. Kiến trúc client-server và REST API : ..... 14
1.1.1. Mô hình client-server ..... 14
1.1.2. Giới thiệu RESTful API ..... 15
1.2. Single Page Application với ReactJS: ..... 16
1.2.1. Giới thiệu về ReactJS ..... 16
1.2.2. Giới thiệu về Vite : ..... 18
1.2.3. Giới thiệu về TanStack Query (React Query) ..... 18

1.3. Framework NestJS và mô hình module hoá : 19
1.4. Hệ quản trị PostgreSQL và TypeORM: 20
1.4.1. Giới thiệu về PostgreSQL 20
1.4.2. TypeORM 22
1.5. Xác thực bằng JWT (Access + Refresh) 22
1.6. Realtime với Socket.IO 23
1.7. Hybrid mobile app với Capacitor 23
1.8. Xây dựng trợ lý ảo với RAG 24
1.8.1. Khái niệm RAG 25
1.8.2. Chunking 25
1.8.3. Embedding 26
1.8.4. Vector Database ( pgvector ) 27
1.8.5. Truy xuất tài liệu ( Retrivel ) 27
1.8.6. Agent & ReAct Agent 31
1.9. Xây dựng hệ thống gợi ý chuyến du lịch 31
1.9.1. Content-based 31
1.9.2. Collaborative Filtering 32
1.9.3. Popularity-based Filtering 32
1.9.4. Weighted Hybrid System 32
1.10. Cổng thanh toán SePay 32
CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 34
2.1. Phân tích yêu cầu 34
2.1.1 Xác định tác nhân của hệ thống 34
2.1.2. Xác định yêu cầu chức năng 34
2.1.3. Yêu cầu phi chức năng 37
2.2. Phân tích hệ thống 38
2.2.1. Sơ đồ nguyên lý hoạt động 38

2.2.2. Sơ đồ ca sử dụng ..... 42
2.3. Thiết kế hệ thống ..... 76
2.3.1. Kiến trúc tổng thể ..... 76
2.3.2. Thiết kế cơ sở dữ liệu ..... 76
2.4. Thiết kế API và phân quyền ..... 81
2.4.1. Convention API ..... 81
2.4.2. Phân quyền ..... 81
2.4.3. Một số endpoint tiêu biểu ..... 82
2.5. Thiết kế giao diện (UI/UX) ..... 82
2.5.1. Hệ thống thiết kế ..... 82
2.5.2. Cấu trúc trang chính ..... 83
Chương 3: TRIỂN KHAI VÀ ĐÁNH GIÁ ..... 84
3.1.Môi trường và công cụ triển khai ..... 84
3.1.1.Môi trường phát triển ..... 84
3.1.2.Công nghệ và thư viện sử dụng ..... 84
3.2 Mô tả kết quả thực tế đạt được ..... 86
3.2.1. Giao diện chức năng chung ..... 86
3.2.2 Giao diện chức quản lý chuyến đi ..... 88
3.2.3 .Chức năng quản lý bài viết ..... 92
3.2.4. Tính năng hướng dẫn viên ..... 93
3.2.5. Giao diện Admin ..... 94
TÀI LIỆU THAM KHẢO ..... 101
PHỤ LỤC ..... 102

# DANH SÁCH CÁC HÌNH VẼ

Hình 1.1 - Mô hình kiến trúc mạng cleint-server ..... 14
Hình 1.2. RESTful API ..... 15
Hình 1.3. Cách hoạt động của Virtual DOM ..... 17
Hình 1.4. Vite & Webpack ..... 18
Hình 1.5. Kiến trúc layered trong mỗi module ..... 20
Hình 1.6. Tính năng của PostgreSQL ..... 21
Hình 1.7. Thư viện TypeORM ..... 22
Hình 1.8. Cơ chế làm việc Capacitor ..... 24
Hình 1.9. Biểu diễn vector Embedding ..... 26
Hình 1.10. Công thức tính Cosine Similarity ..... 28
Hình 1.11. Công thức tính điểm BM25 ..... 29
Hình 1.12. Công thức tính điểm Ranking Fusion ..... 30
Hình 1.13. Công thức tính điểm Weighted ..... 32
Hình 1.14. Cơ chế thanh toán Sepay ..... 33
Hình 2.1. Luồng đăng nhập ..... 38
Hình 2.2. Luồng hoạt động của khách du lịch ..... 39
Hình 2.3. Luồng hoạt động của hướng dẫn viên ..... 40
Hình 2.4. Luồng hoạt động của quản trị viên ..... 41
Hình 2.5. Sơ đồ ca sử dụng chức năng quản lý hồ sơ cá nhân ..... 45
Hình 2.6. Sơ đồ ca sử dụng chức năng quản lý bài viết ..... 47
Hình 2.7. Sơ đồ ca sử dụng chức năng tương tác bài viết ..... 48
Hình 2.8. Sơ đồ ca sử dụng chức năng quản lý chuyến đi ..... 50
Hình 2.9. Sơ đồ ca sử dụng chức năng gửi yêu cầu tham gia ..... 51
Hình 2.10. Sơ đồ ca sử dụng chức năng xử lý yêu cầu tham gia ..... 53
Hình 2.11. Sơ đồ ca sử dụng chức năng chat nhóm ..... 54

Hình 2.12. Sơ đồ ca sử dụng chức năng đăng ký làm HDV ..... 56
Hình 2.13. Sơ đồ ca sử dụng chức năng tìm kiếm HDV ..... 57
Hình 2.14. Sơ đồ ca sử dụng đặt lịch hướng dẫn viên cho tour ..... 58
Hình 2.15. Sơ đồ ca sử dụng chức năng nạp tiền ..... 60
Hình 2.16. Sơ đồ ca sử dụng chức năng đánh giá ..... 62
Hình 2.17. Sơ đồ ca sử dụng chức năng bot chat ..... 63
Hình 2.18. Sơ đồ ca sử dụng chức năng quản lý booking ..... 65
Hình 2.19. Sơ đồ ca sử dụng chức năng quản lý lịch bận ..... 66
Hình 2.20. Sơ đồ ca sử dụng quản lý doanh thu và rút tiền ..... 67
Hình 2.21. Sơ đồ ca sử dụng chức năng thống kê tổng quan ..... 68
Hình 2.22. Sơ đồ ca sử dụng duyệt yêu cầu làm hướng dẫn viên ..... 69
Hình 2.23. Sơ đồ ca sử dụng duyệt rút tiền ..... 71
Hình 2.24. Sơ đồ ca sử dụng chức năng khóa tài khoản ..... 72
Hình 2.25. Sơ đồ ca sử dụng chức năng quản lý doanh thu ..... 74
Hình 2.26. Sơ đồ kiến trúc hệ thống ..... 76
Hình 2.27. Sơ đồ quan hệ ..... 77
Hình 3.1. Màn hình chức năng đăng nhập ..... 87
Hình 3.2. Màn hình chức năng đăng ký ..... 87
Hình 3.3. Màn hình chức năng chỉnh sửa hồ sơ người dùng ..... 88
Hình 3.4. Màn hình chức năng chat với AI ..... 88
Hình 3.5. Màn hình chức năng xem danh sách chuyến đi ..... 89
Hình 3.6. Màn hình chức năng xem gợi ý về chuyến đi ..... 89
Hình 3.7. Màn hình chức năng tạo chuyến đi ..... 90
Hình 3.8. Màn hình chức năng xem chi tiết chuyến đi ..... 91
Hình 3.9. Màn hình chức năng nhắn tin trong nhóm ..... 91
Hình 310. Màn hình chức năng chỉnh sửa chuyến đi ..... 92
Hình 3.11. Màn hình chức năng quản lý bài viết ..... 92

Hình 3.12. Màn hình chức năng đăng ký làm hướng dẫn viên .....93
Hình 3.13. Màn hình chức năng quản lý đặt lịch .....93
Hình 3.14. Màn hình chức năng chi tiết đặc lịch .....94
Hình 3.15. Màn hình chức năng quản lý doanh thu .....94
Hình 3.16. Màn hình chức năng thống kê của Admin .....95
Hình 3.17. Màn hình chức năng quản lý user .....95
Hình 3.18. Màn hình chức năng duyệt hồ sơ hướng dẫn viên .....96
Hình 3.19. Màn hình chức năng quản lý bài viết (Admin) .....97
Hình 3.20. Màn hình chức năng quản lý địa điểm .....97
Hình 3.21. Màn hình chức năng xử lý rút tiền .....98
Hình 3.23. Kết quả trên giao ứng dụng .....99

# DANH SÁCH CÁC BẢNG

Bảng 1.1 Tham số mặc định của phương pháp chunking ..... 26
Bảng 2.1. Xác định tác nhân hệ thống ..... 34
Bảng 2.2. Xác định yêu cầu khách vãng lai ..... 34
Bảng 2.3. Xác định yêu cầu khách du lịch ..... 35
Bảng 2.4. Xác định yêu cầu hướng dẫn viên ..... 36
Bảng 2.5. Xác định yêu cầu quản trị viên ..... 36
Bảng 2.6. Yêu cầu bảo mật ..... 37
Bảng 2.7. Tổng quan chức năng hệ thống ..... 42
Bảng 2.8. Cấu trúc bảng users ..... 78
Bảng 2.9. Cấu trúc bảng trips ..... 78
Bảng 2.10. Cấu trúc bảng guide_profiles ..... 80
Bảng 2.11. cấu trúc bảng guide_bookings ..... 80
Bảng 2.12. Một số endpoint thể tiêu biểu ..... 82
Bảng 3.1. Môi trường phát triển ..... 84
Bảng 3.2. Thư viên sự dụng trong backend ..... 84
Bảng 3.3. Thư viện sử dụng trong Frontend ..... 85
Bảng 3.4. Thư viện và dịch vụ bên ngoài ..... 86

# DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

| Từ viết tắt | Tên đầy đủ | Ý nghĩa |
|---|---|---|
| FE | Frontend | Giao diện phía người dùng, nơi hiển thị thông tin và tiếp nhận thao tác từ người dùng. |
| BE | Backend | Hệ thống xử lý nghiệp vụ phía máy chủ, quản lý dữ liệu và giao tiếp với các dịch vụ khác. |
| VNĐ | Việt Nam Đồng | Đơn vị tiền tệ chính thức của Việt Nam. |
| QR | Quick Response Code | Mã phản hồi nhanh dùng để mã hóa thông tin thanh toán hoặc dữ liệu khác. |
| MB BANK | Ngân hàng Thương mại Cổ phần Quân đội | Ngân hàng Thương mại Cổ phần Quân đội |
| API | Application Programming Interface | Phương thức trung gian cho phép các phần mềm và hệ thống khác nhau giao tiếp và tương tác với nhau một cách có cấu trúc |
| Tnx | Transaction | Giao dịch phát sinh trong hệ thống |
| JWT | JSON Web Token | Một chuẩn mở dùng để truyền tải thông tin an toàn giữa client và server dưới dạng chuỗi JSON được mã hóa và ký số. |
| CORS | Cross-Origin Resource Sharing | Cơ chế cho phép nhiều tài nguyên khác nhau (fonts, Javascript, v.v…) của một trang web có thể được truy vấn từ domain khác với domain của trang đó. |

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Du lịch là một trong những ngành kinh tế trọng điểm của Việt Nam. Theo Tổng cục Du lịch, năm 2024 ngành du lịch đón hơn 17,5 triệu lượt khách quốc tế và phục vụ 110 triệu lượt khách nội địa, đóng góp khoảng 9,2% GDP. Cùng với sự phát triển đó, hành vi tiêu dùng du lịch của người Việt cũng thay đổi mạnh mẽ: người dùng ngày càng tự tổ chức chuyến đi, ưu tiên trải nghiệm cá nhân hoá, tham khảo đánh giá thực tế từ cộng đồng và đặc biệt là ưu chuộng kết nối trực tiếp với người dân địa phương thay vì các tour đóng gói cứng nhắc.

Tuy nhiên, hệ sinh thái ứng dụng du lịch tại Việt Nam vẫn đang bị phân mảnh: người dùng phải sử dụng cùng lúc nhiều nền tảng cho các nhu cầu khác nhau:

- Facebook, Instagram, TikTok: chia sẻ ảnh, kinh nghiệm, gợi ý điểm đến nhưng không có tính năng đặt chuyến đi hay ghép nhóm.
- Traveloka, Booking.com, Agoda: tốt cho đặt khách sạn và vé máy bay nhưng nội dung cộng đồng nghèo nàn, không có chức năng tổ chức chuyến đi nhóm.
- Klook, KKday: bán tour và hoạt động trải nghiệm nhưng theo mô hình bán lẻ, không kết nối được hướng dẫn viên độc lập.
- Group Facebook "Đà Nẵng Ăn Gì ? Ở đâu ?" và các diễn đàn: cộng đồng sôi động nhưng không có công cụ chính thức để quản lý chuyến đi, đặt cọc, đánh giá an toàn.

Khoảng trống thị trường là nền tảng hợp nhất kết hợp ba yếu tố: (1) mạng xã hội du lịch để khám phá và truyền cảm hứng, (2) công cụ tổ chức – tham gia chuyến đi nhóm có quản lý chính thức, (3) thị trường đặt hướng dẫn viên địa phương có thanh toán an toàn. Đây chính là cơ hội để TripMate ra đời.

Mặt khác, từ góc độ học thuật, đề tài này cho phép sinh viên thực hành toàn diện các nội dung đã học: phân tích thiết kế hệ thống, lập trình full-stack hiện đại (React + NestJS + PostgreSQL), kiến trúc REST API, xác thực JWT, realtime, AI sinh nội dung, tích hợp cổng thanh toán, và đặc biệt là đóng gói ứng dụng đa nền tảng (web + mobile) từ một codebase duy nhất – một xu hướng đang được nhiều doanh nghiệp công nghệ áp dụng.

## 2. Mục tiêu đề tài

### *2.1. Mục tiêu*

Xây dựng một nền tảng mạng xã hội du lịch thế hệ mới kết hợp đặt tour và hướng dẫn viên trực tuyến, hoạt động đồng thời trên web và Android, đáp ứng đầy đủ luồng nghiệp vụ từ khám phá – kết nối – đặt dịch vụ – thanh toán – đánh giá trong cùng một sản phẩm.

### *2.2. Mục tiêu cụ thể :*

- Phân tích và thiết kế hệ thống đáp ứng 4 nhóm vai trò: Khách (Guest), Du khách (Traveler), Hướng dẫn viên (Guide), Quản trị viên (Admin).
- Xây dựng backend NestJS với 14 module quản lý đầy đủ nghiệp vụ, tài liệu API qua Swagger, xác thực JWT chuẩn (access + refresh).
- Xây dựng frontend ReactJS với 24 trang, hỗ trợ responsive đa thiết bị, tận dụng React Query cho quản lý dữ liệu server-side.
- Xây dựng trợ lý AI du lịch theo kiến trúc pipeline (hiểu yêu cầu → truy xuất dữ liệu → sinh câu trả lời) dùng mô hình ngôn ngữ tự host + RAG, gợi ý lộ trình theo ngôn ngữ tự nhiên mà không phụ thuộc cứng vào dịch vụ AI bên ngoài.
- Tích hợp cổng thanh toán SePay cho phép nạp tiền qua chuyển khoản ngân hàng tự động (webhook).
- Tích hợp Firebase Storage cho lưu trữ ảnh, Leaflet + OpenStreetMap cho bản đồ và định tuyến
- Đóng gói ứng dụng Android qua Capacitor, build được file APK chạy trên thiết bị thật.
- Đảm bảo bảo mật cơ bản: hash mật khẩu (bcrypt), token rotation, CORS, rate limit, input validation.

## 3. Phạm vi và giới hạn

### *3.1. Phạm vi thực hiện*

Đối tượng người dùng: thị trường du lịch Việt Nam, ngôn ngữ chính tiếng Việt

**Tính năng triển khai :**

- Đăng ký, đăng nhập, quản lý hồ sơ người dùng (cá nhân, sở thích du lịch, ảnh đại diện, ảnh bìa).
- Mạng xã hội: đăng bài, like, bình luận (reply), chia sẻ, theo dõi người dung
- Quản lý chuyến đi: tạo, chỉnh sửa, gửi yêu cầu tham gia, duyệt yêu cầu, lịch trình hằng ngày, gallery ảnh, chat nhóm tự động cho thành viên chuyến đi.
- Hướng dẫn viên: đăng ký hồ sơ, được admin duyệt, nhận booking, quản lý lịch bận, dashboard doanh thu.
- Booking tour: vòng đời 5 trạng thái (pending → accepted → paid → completed/cancelled), kèm chính sách hoàn tiền.

- Ví điện tử: số dư khả dụng + đóng băng, lịch sử giao dịch, nạp tiền qua SePay, rút tiền có duyệt.
- Đánh giá đa hình thức: chuyến đi, hướng dẫn viên, địa điểm, thành viên cùng chuyến.
- Đánh giá đa hình thức: chuyến đi, hướng dẫn viên, địa điểm, thành viên cùng chuyến.
- AI assistant: chat bong bóng, gợi ý chuyến đi mới
- Gợi ý chuyến đi cá nhân hoá theo User.preferences (style du lịch, ngân sách, trải nghiệm, ngôn ngữ).
- Bản đồ điểm đến qua Google Maps Embed, click để xem đường đi.
- Bookmark/lưu nhanh: bài viết, chuyến đi, hướng dẫn viên.
- Thông báo realtime: khi có yêu cầu tham gia, like, comment, follow, đặt tour, xác nhận thanh toán.
- Quản trị viên: thống kê tổng quan, khoá/mở user, duyệt hồ sơ HDV, duyệt rút tiền, kiểm duyệt nội dung.

### *3.2. Giới hạn :*

- Chưa triển khai phiên bản iOS (do hạn chế tài khoản Apple Developer).
- Chưa triển khai gateway thanh toán quốc tế (Visa/Mastercard); chỉ hỗ trợ chuyển khoản nội địa qua SePay/MB Bank.
- Chưa triển khai bản dịch sang ngôn ngữ khác (i18n).
- Realtime chat hiện đang ở mức cơ bản qua Socket.IO; chưa hỗ trợ video call.

## 4. Đối tượng nghiên cứu :

- Đối tượng người dùng cuối: du khách trẻ Việt Nam (18–40 tuổi) yêu thích du lịch tự túc, thích chia sẻ trải nghiệm; hướng dẫn viên địa phương muốn có kênh marketing độc lập.
- Đối tượng kỹ thuật: kiến trúc microservice nhẹ qua module hoá NestJS, mô hình Backend-for-Frontend, TypeORM với PostgreSQL, hybrid app qua Capacitor, AI sinh nội dung, cổng thanh toán webhook.

## 5. Phương pháp nghiên cứu :

- Phương pháp khảo sát: quan sát hành vi sử dụng các nền tảng cạnh tranh (Klook, Traveloka, Facebook Group), thu thập case study về tổ chức tour nhóm.
- Phương pháp phân tích thiết kế: sử dụng UML cho Use Case, kiến trúc layered (Controller – Service – Repository), thiết kế CSDL chuẩn 3NF.
- Phương pháp lập trình thực nghiệm: áp dụng Agile lặp ngắn (1-2 ngày/feature), test đi kèm tính năng, version control bằng Git.
- Phương pháp đánh giá: kiểm thử chức năng thủ công theo Use Case, kiểm thử trên Android và thiết bị thật, ghi nhận feedback và iteration.

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## 1.1. Kiến trúc client-server và REST API :

### *1.1.1. Mô hình client-server*

Mô hình Client–Server là một kiến trúc mạng phổ biến trong các hệ thống phần mềm hiện đại, đặc biệt là trong các ứng dụng web và ứng dụng di động. Trong mô hình này, hệ thống được tổ chức thành hai thành phần chính là client (máy khách) và server (máy chủ), mỗi thành phần đảm nhận một vai trò riêng biệt nhưng có sự phối hợp chặt chẽ với nhau. [8]




*Hình 1.1 - Mô hình kiến trúc mạng cleint-server*

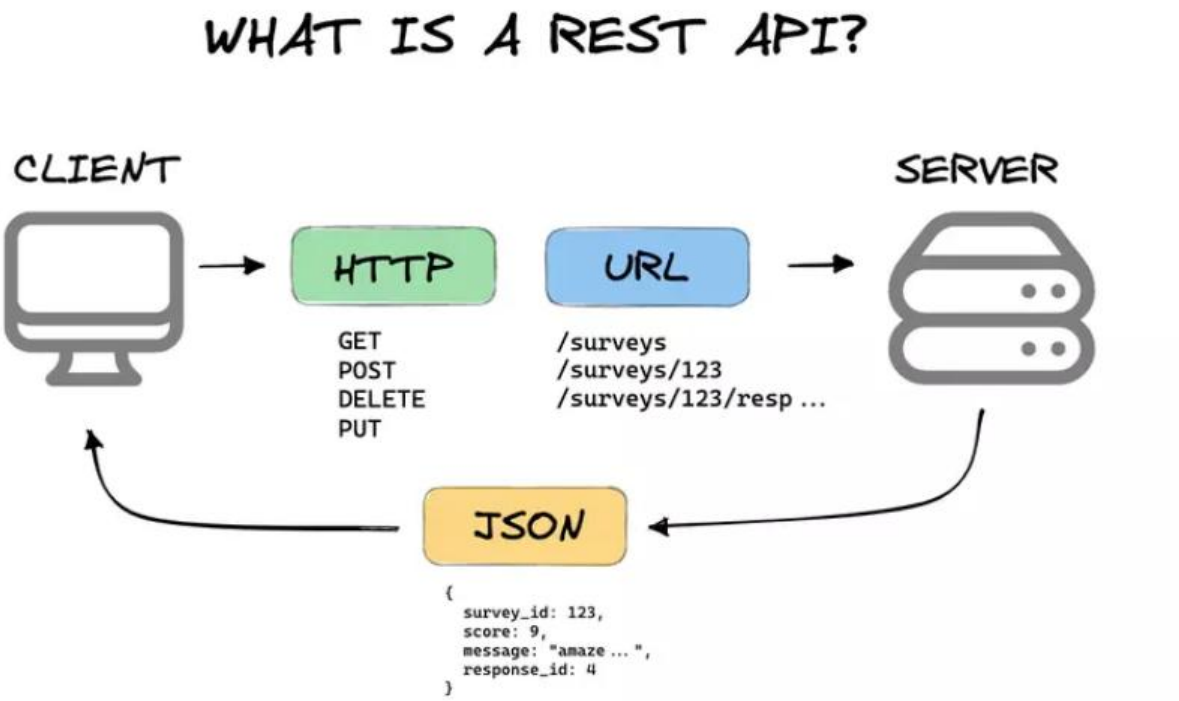
Client (máy khách) : có nhiệm vụ gửi các yêu cầu (request) đến server thông qua giao thức mạng. Client có thể là trình duyệt web, ứng dụng di động hoặc ứng dụng desktop. Về mặt chức năng, client thường đảm nhiệm phần giao diện người dùng (UI), cho phép người dùng tương tác với hệ thống một cách trực quan. Tuy nhiên, client không trực tiếp xử lý các nghiệp vụ phức tạp hoặc truy cập cơ sở dữ liệu mà phụ thuộc vào server.

Server là hệ thống trung tâm chịu trách nhiệm tiếp nhận, xử lý và phản hồi các yêu cầu từ client. Server thường đảm nhận các nhiệm vụ như xử lý logic nghiệp vụ, truy vấn cơ sở dữ liệu, xác thực người dùng và quản lý tài nguyên hệ thống. Sau khi xử lý xong yêu cầu, server sẽ trả kết quả về cho client dưới dạng dữ liệu hoặc thông báo.

Trong dự án TripMate, mô hình này được áp dụng theo dạng stateless: server không lưu phiên người dùng trên RAM mà sử dụng JWT token để xác định danh tính. Điều này cho phép horizontal scaling dễ dàng – có thể chạy nhiều instance backend song song mà không gặp vấn đề session affinity.

### *1.1.2. Giới thiệu RESTful API*

RESTful API (Representational State Transfer API) là một phong cách kiến trúc và phương pháp tiếp cận được sử dụng rộng rãi trong việc xây dựng các dịch vụ web. RESTful API dựa trên giao thức HTTP để thực hiện các hoạt động CRUD (Create, Read, Update, Delete) trên các tài nguyên [1]. Sự phổ biến của RESTful API đến từ tính đơn giản, dễ hiểu và khả năng mở rộng dễ dàng.




*Hình 1.2. RESTful API*

Nguyên tắc cơ bản:

- Client–Server phân tách rõ ràng giữa client và server, trong đó client đảm nhiệm giao diện người dùng, còn server xử lý dữ liệu và logic nghiệp vụ.

- Stateless quy định mỗi yêu cầu từ client phải chứa đầy đủ thông tin để server xử lý mà không cần lưu trạng thái trước đó.

- Cacheable cho phép phản hồi từ server được lưu tạm thời nhằm tăng hiệu suất và giảm tải hệ thống.

- Uniform Interface đảm bảo giao tiếp thống nhất thông qua các URL và định dạng dữ liệu chuẩn như JSON.

- Layered System cho phép hệ thống được tổ chức theo nhiều lớp trung gian như bảo mật hoặc cân bằng tải để tăng khả năng mở rộng. Cuối cùng, Code on Demand là tùy chọn, cho phép server cung cấp mã thực thi cho client nhằm tăng tính linh hoạt của hệ thống

Các phương thức HTTP cơ bản được sử dụng trong RESTful API bao gồm:

- GET: Truy xuất thông tin của một hoặc nhiều tài nguyên.

- POST: Tạo mới một tài nguyên.

- PUT: Cập nhật toàn bộ thông tin của một tài nguyên đã tồn tại.

- PATCH: Cập nhật một phần thông tin của tài nguyên.

- DELETE: Xóa một tài nguyên khỏi hệ thống.

Trong TripMate, toàn bộ API tuân thủ REST với prefix /api, ví dụ:

- GET /api/trips : lấy danh sách chuyến đi

- POST /api/trips : tạo chuyến đi mới

- PATCH /api/trips/:id — cập nhật chuyến đi

- DELETE /api/trips/:id : xoá chuyến đi

## 1.2. Single Page Application với ReactJS:

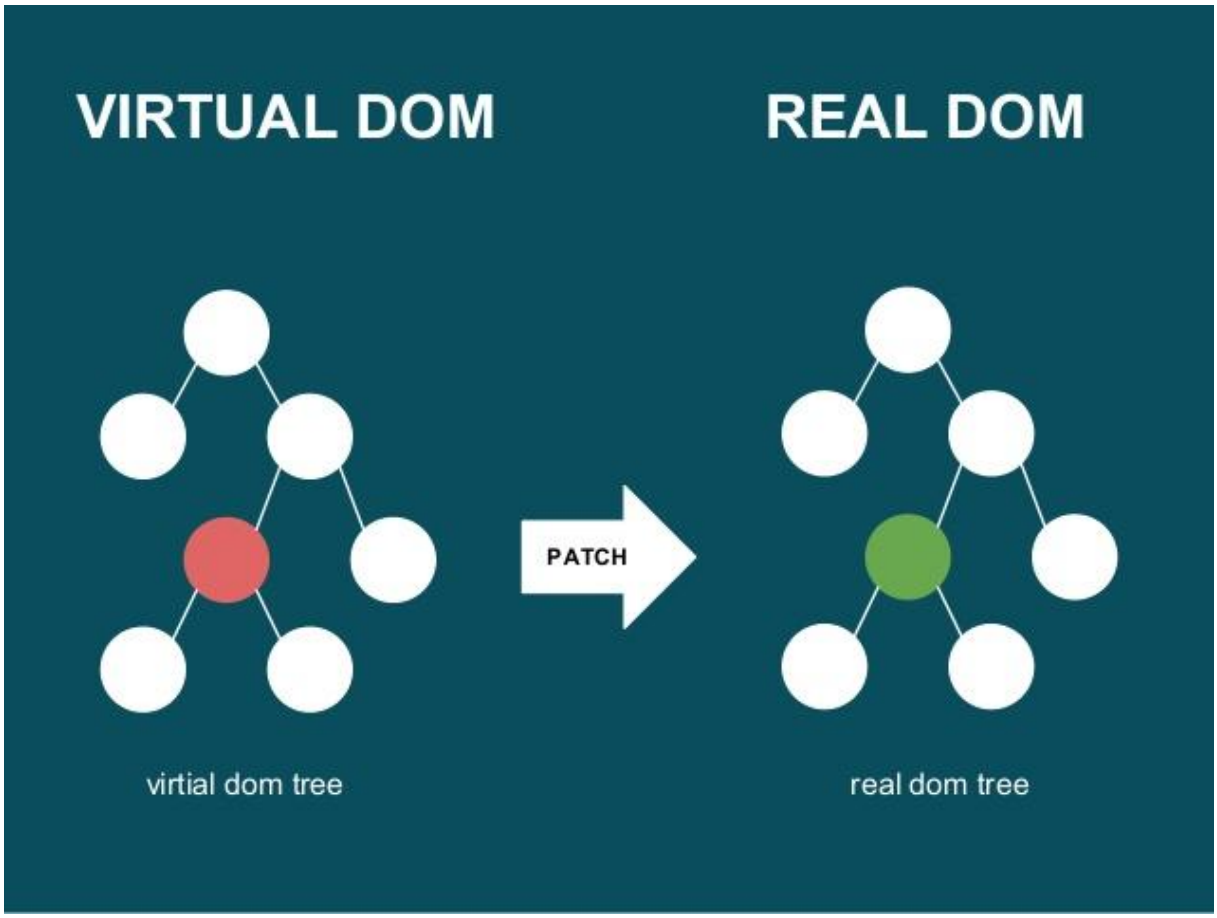
### *1.2.1. Giới thiệu về ReactJS*

React là một thư viện JavaScript dùng để xây dựng giao diện người dùng (UI), được phát triển bởi Meta Platforms nhằm hỗ trợ tạo ra các thành phần giao diện có tính tương tác cao, có trạng thái và có thể tái sử dụng [2]. React hiện đang được sử dụng trong các hệ thống thực tế tại Facebook, và các sản phẩm như Instagram cũng được xây dựng dựa trên thư viện này.

Một trong những ưu điểm nổi bật của React là khả năng hoạt động linh hoạt ở cả phía client và server, cho phép render giao diện hiệu quả và tối ưu trải nghiệm người dùng. Bên cạnh đó, React hỗ trợ cơ chế kết nối giữa các component, giúp việc xây dựng ứng dụng trở nên có cấu trúc và dễ mở rộng hơn.

Ngoài ra, React sử dụng khái niệm Virtual DOM (DOM ảo) để tối ưu quá trình cập nhật giao diện. Khi trạng thái của ứng dụng thay đổi, React sẽ so sánh Virtual DOM với DOM thật và chỉ cập nhật những phần tử cần thiết, từ đó giúp giảm thiểu thao tác trực tiếp lên DOM và tăng hiệu năng của ứng dụng

**Cách hoạt động của Virtual DOM**




Hình 1.3. Cách hoạt động của Virtual DOM

Tưởng tượng bạn có một vật thể được thiết kế dựa trên một người. Nó có mọi bộ phận một người có thể có, và phản ánh lại trạng thái hiện tại của người đó. Đây là điều cơ bản React làm với DOM. Bây giờ thử nghĩ rằng nếu bạn lấy vật thể đó và thay đổi một vài bộ phận. Thêm ria, bắp tay và đôi mắt xanh. Trong React, khi chúng ta tạo ra thay đổi, sẽ diễn ra 2 việc. Đầu tiên, React chạy một thuật toán so sánh sự khác biệt để

phát hiện ra thay đổi. Bước thứ 2 là điều hòa bằng cách cập nhật DOM với kết qủa của thuật toán ở bước 1. Cách React hoạt động là thay vì lấy một người thật và tái tạo lại từ đầu, React chỉ thay đổi khuôn mặt và tay. Điều này có nghĩa là khi bạn nhập vào 1 đoạn text và render xong thì đoạn text này sẽ không bị ảnh hưởng cho tới khi parent node được sắp xếp lại.

### *1.2.2. Giới thiệu về Vite :*

Vite hay Vitejs là một công cụ build và phát triển ứng dụng web được tạo ra bởi Evan You, người đã phát triển Vue.js, Vite ra đời mục đích khắc phục những hạn chế về tốc độ và hiệu suất[3]




*Hình 1.4. Vite & Webpack*

Vite có hiệu năng phát triển nhanh hơn Webpack đáng kể nhờ cách tiếp cận hiện đại trong quá trình xử lý và đóng gói mã nguồn. Cụ thể, trong chế độ development, Vite tận dụng trực tiếp native ES Modules (ESM) của trình duyệt, nghĩa là các module JavaScript được trình duyệt tải theo nhu cầu thay vì phải bundle toàn bộ ứng dụng ngay từ đầu như Webpack. Điều này giúp thời gian khởi động server phát triển gần như tức thì và các thay đổi trong code cũng được cập nhật nhanh hơn thông qua cơ chế HMR (Hot Module Replacement).

TripMate sử dụng Vite vì:

- Cold start dưới 1 giây
- HMR (Hot Module Replacement) gần như tức thì
- Build production tối ưu, hỗ trợ code splitting tự động qua dynamic import.

### *1.2.3. Giới thiệu về TanStack Query (React Query)*

React Query là thư viện quản lý server state. Khác với Redux quản lý client state, React Query đặc biệt cho dữ liệu fetch từ API[4]:

- Cache thông minh: tự động dedupe request giống nhau.
- Background refetch: dữ liệu luôn fresh khi user quay lại tab.
- Optimistic updates: UI cập nhật ngay khi user bấm, rollback nếu API fail.
- Pagination/infinite scroll built-in.

Trong TripMate, mọi API call qua useQuery / useMutation, giúp giảm boilerplate so với việc tự viết useEffect + useState.

### 1.3. Framework NestJS và mô hình module hoá :

NestJS là framework Node.js tham khảo từ Angular, hỗ trợ TypeScript first-class. Triết lý chính của Nest:

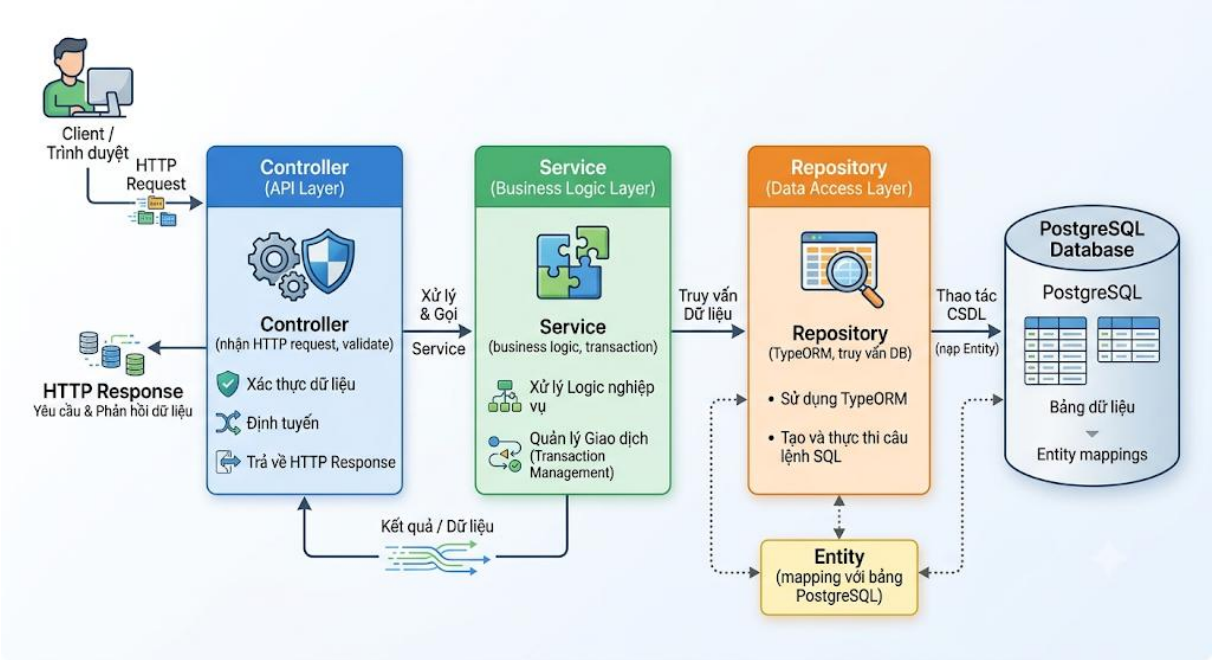
Module hoá: mỗi nghiệp vụ là một module : ( AuthModule , UserModule, TripModule...) gồm controller + service + entity.

Dependency Injection (DI): Nest tự inject dependency qua decorator @Injectable() , dễ test và mock.

Decorator-based routing: @Get('/users'), @Body(), @Param('id') rõ ràng và dễ đọc.

Guards / Interceptors / Pipes: middleware mạnh mẽ. TripMate dùng JwtAuthGuard để bảo vệ route, ValidationPipe để validate DTO bằng class-validator.

Kiến trúc layered trong mỗi module:




*Hình 1.5. Kiến trúc layered trong mỗi module*

## 1.4. Hệ quản trị PostgreSQL và TypeORM:

### *1.4.1. Giới thiệu về PostgreSQL*

PostgreSQL là hệ quản trị cơ sở dữ liệu quan hệ hướng đối tượng (ORDBMS) mã nguồn mở, miễn phí và được đánh giá là một trong những hệ quản trị cơ sở dữ liệu mạnh mẽ nhất hiện nay. PostgreSQL được phát triển bởi PostgreSQL Global Development Group và phát hành lần đầu vào năm 1996. Hệ thống có khả năng hoạt động trên nhiều nền tảng như Windows, Linux và macOS [5].




*Hình 1.6. Tính năng của PostgreSQL*

PostgreSQL nổi bật nhờ tính ổn định cao, khả năng mở rộng tốt và hỗ trợ xử lý lượng lớn dữ liệu cũng như nhiều người dùng truy cập đồng thời. Hệ quản trị này cung cấp nhiều tính năng như truy vấn phức tạp, Trigger, View, Transaction và cơ chế kiểm soát đồng thời đa phiên bản (MVCC) giúp đảm bảo tính nhất quán dữ liệu.

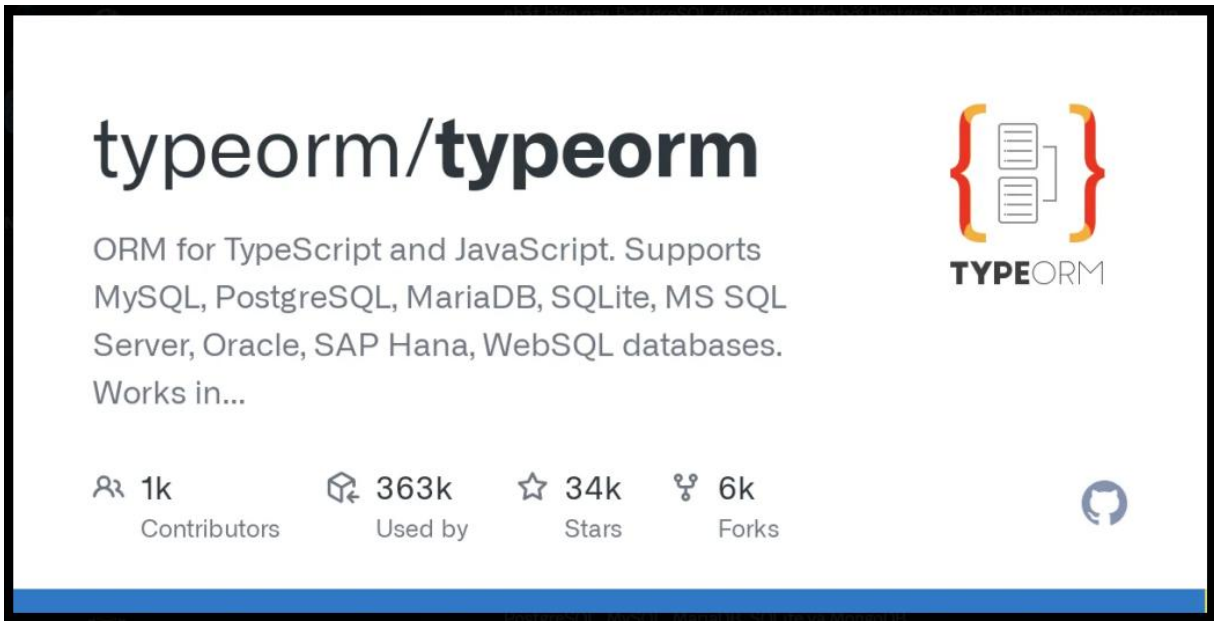
Ngoài ra, PostgreSQL hỗ trợ nhiều kiểu dữ liệu đa dạng như số, chuỗi, ngày giờ, UUID, JSON/JSONB và XML; đồng thời cung cấp các cơ chế đảm bảo toàn vẹn dữ liệu như Primary Key, Foreign Key, UNIQUE và NOT NULL. Hệ thống còn hỗ trợ sao lưu, phục hồi dữ liệu, sao chép dữ liệu (Replication), bảo mật nhiều lớp và nhiều phương pháp lập chỉ mục giúp tối ưu hiệu suất truy vấn.

TripMate chọn PostgreSQL thay vì MySQL vì:

- Hỗ trợ kiểu JSONB (dùng cho cột User.preferences, Notification.meta).
- Hỗ trợ enum native (dùng cho Booking.status, WalletTxnType).
- Khả năng full-text search built-in (cho phép mở rộng search trips trong tương lai).
- Cộng đồng và tài liệu phong phú.

### *1.4.2. TypeORM*

TypeORM là một thư viện ORM (Object Relational Mapping) mã nguồn mở dành cho Node.js, hỗ trợ cả JavaScript và TypeScript. Thư viện này cho phép lập trình viên làm việc với cơ sở dữ liệu thông qua các đối tượng trong mã nguồn thay vì phải viết nhiều câu lệnh SQL thủ công. TypeORM hỗ trợ nhiều hệ quản trị cơ sở dữ liệu phổ biến như PostgreSQL, MySQL, MariaDB, SQLite và MongoDB [6].




*Hình 1.7. Thư viện TypeORM*

Ưu điểm : cung cấp nhiều tính năng hữu ích như sử dụng Decorator để định nghĩa bảng và các mối quan hệ dữ liệu, hỗ trợ hai mô hình Active Record và Data Mapper, cùng công cụ Migration giúp quản lý và đồng bộ cấu trúc cơ sở dữ liệu. Nhờ khả năng tận dụng hệ thống kiểu dữ liệu của TypeScript, TypeORM giúp mã nguồn dễ đọc, dễ bảo trì và giảm thiểu lỗi trong quá trình phát triển.

Nhược điểm : hiệu năng có thể không tối ưu đối với các dự án lớn hoặc các truy vấn phức tạp. Ngoài ra, việc cấu hình ban đầu tương đối phức tạp, đặc biệt khi sử dụng cùng TypeScript và nhiều môi trường triển khai khác nhau.

## 1.5. Xác thực bằng JWT (Access + Refresh)

JWT (JSON Web Token) là chuẩn token tự ký gồm 3 phần: header, payload, signature. TripMate áp dụng mô hình dual-token [9]:

- Access Token: TTL 15 phút, payload chứa userId, role, dùng để xác thực mỗi request qua header Authorization: Bearer <token>.

- Refresh Token: TTL 7 ngày, lưu hash trong DB. Khi access token hết hạn, FE gọi /auth/refresh để lấy access mới mà không cần đăng nhập lại

Khi user đăng xuất, refresh token bị xoá khỏi DB. Khi đổi mật khẩu, tất cả refresh token bị thu hồi (logout all devices).

Mật khẩu được hash bằng bcrypt với salt rounds 10, đảm bảo không thể decode ngược dù DB bị lộ.

### 1.6. Realtime với Socket.IO

Socket.IO là một thư viện giúp xây dựng giao tiếp thời gian thực (real-time communication) giữa client và server dựa trên WebSocket, đồng thời có cơ chế fallback sang các phương thức khác như long polling trong trường hợp trình duyệt hoặc môi trường mạng không hỗ trợ WebSocket. Điều này giúp đảm bảo kết nối ổn định và tương thích tốt hơn trên nhiều thiết bị và điều kiện mạng khác nhau [10] .

TripMate dùng Socket.IO cho:

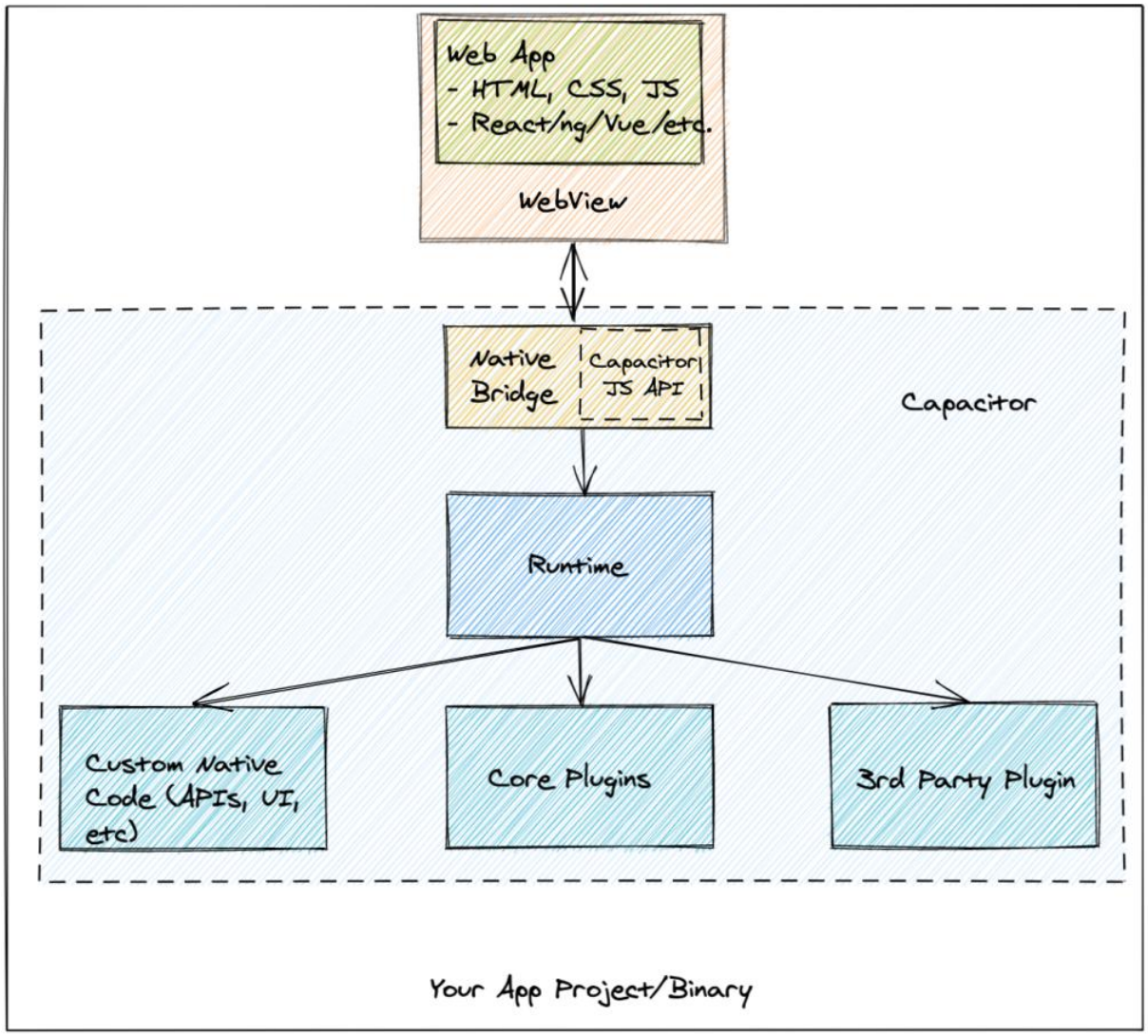
- Chat realtime trong group chuyến đi.
- Push notification real-time khi có like, comment, follow, booking mới.

Server bên Nest dùng @WebSocketGateway() decorator, FE dùng socket.io-client qua hook custom useSocket().

### 1.7. Hybrid mobile app với Capacitor

Capacitor là runtime cross-platform của Ionic Team, đóng gói web app thành ứng dụng native qua WebView. Khác với React Native (chạy native components qua JS bridge), Capacitor chạy 100% web code trong WebView native, kèm bridge cho phép gọi native API (camera, GPS, notifications...).

***Cơ chế làm việc***




*Hình 1.8. Cơ chế làm việc Capacitor*

Trong đó Web View là một phần không thể thiếu của hệ điều hành Android và iOS. Web View cho phép hiển thị trang web giống như một web browser nên các web app cũng sẽ hoạt động tốt. Tận dụng điều này, Capacitor gói gọn web app của chúng ta trong Web View, rồi hoạt động như một cầu nối giúp tương tác với các tính năng "native" (tức các tính năng đặc thù của hệ điều hành, ví dụ để mở một dialog, Android sẽ có cách implement khác, iOS có cách implement khác), khi cần sử dụng tính năng nào, chỉ cần gọi đến tính năng đó thông qua API.

TripMate chọn Capacitor vì codebase web React đã có sẵn, không cần viết lại bằng Dart hay JSX-native.

## 1.8. Xây dựng trợ lý ảo với RAG

### *1.8.1. Khái niệm RAG*

RAG (Retrieval-Augmented Generation) là một phương pháp kết hợp giữa khả năng truy xuất thông tin từ các nguồn dữ liệu bên ngoài (Retrieval) với sức mạnh tạo nội dung của mô hình ngôn ngữ lớn (LLM - Large Language Model) [7]

Hiểu 1 cách đơn giản RAG là một cách tiếp cận kết hợp giữa hai thành phần:

- Retrieval (Truy xuất): Tìm kiếm thông tin liên quan từ nguồn dữ liệu bên ngoài, như tài liệu, cơ sở dữ liệu hoặc website.
- Generation (Sinh nội dung): Sử dụng LLM để tạo câu trả lời dựa trên thông tin đã truy xuất.

### *1.8.2. Chunking*

Chunking là quá trình chia nhỏ các tài liệu hoặc đoạn văn bản dài thành các phần nhỏ hơn (được gọi là chunks) trước khi lưu trữ vào cơ sở dữ liệu vector. Đây là bước thiết yếu trong hệ thống RAG (Retrieval-Augmented Generation) để AI có thể tìm kiếm và trả lời câu hỏi chính xác nhất

***Vai trò của Chungking trong RAG***

- Tối ưu hóa khả năng tìm kiếm (Retrieval): Cơ sở dữ liệu vector chỉ hoạt động hiệu quả với các đoạn văn bản có độ dài vừa phải. Các đoạn quá dài sẽ làm loãng ý nghĩa, khiến AI không tìm được thông tin cốt lõi.
- Giữ đủ ngữ cảnh cho AI: Các chunk cần đủ nhỏ để tìm kiếm chính xác, nhưng vẫn phải đủ lớn để chứa một ý nghĩa trọn vẹn, giúp mô hình ngôn ngữ lớn (LLM) hiểu và trả lời đúng

***Phương pháp Chunking***

- Sentence-aware (Nhận thức câu): Hệ thống sẽ phân tích và phát hiện các ranh giới tự nhiên của câu (như dấu chấm, dấu câu). Nó tuyệt đối không cắt ngang một câu hay tách rời các ý nghĩa trọn vẹn
- Fixed-size (Kích thước cố định): Các câu nguyên vẹn này sẽ được ghép nối tiếp nhau cho đến khi đạt đến một giới hạn kích thước/token cố định

***Tham số mặc định***

*Bảng 1.1 Tham số mặc định của phương pháp chunking*

| Tham số | Giá trị | Ý nghĩa |
|---|---|---|
| marChars | 900 | Kích thước tối đa mỗi chunk ( ký tự ) |
| minChars | 350 | Kích thước tối thiểu, nếu ngắn hơn ngưỡng này sẽ được gộp vào chunk trước |
| overlapChars | 160 | ký tự tối đa được phép lặp lại giữa các đoạn liền kề |

### *1.8.3. Embedding*

***Khái niệm***

Vector embedding là các biểu diễn số chuyển đổi dữ liệu phức tạp – chẳng hạn như văn bản, hình ảnh và tài liệu – thành các mảng số thực đa chiều. Chúng thường được biểu diễn dưới dạng một chuỗi số trong không gian đa chiều, trong đó sự kết hợp của tất cả các giá trị đặc trưng cho dữ liệu đầu vào. Các phép nhúng này nắm bắt các mối quan hệ ngữ nghĩa, cho phép máy móc xử lý và so sánh dữ liệu một cách hiệu quả. Bằng cách ánh xạ các điểm dữ liệu tương tự lại gần nhau hơn trong không gian vectơ, các phép nhúng cho phép nhiều ứng dụng khác nhau, từ xử lý ngôn ngữ tự nhiên (NLP) và hệ thống đề xuất đến phát hiện bất thường, RAG và hệ thống trả lời câu hỏi.

***Mô hình embedding được biểu diễn như một hàm ánh xạ***

$$E : text \to \mathbb{R}^d, \quad v = E(chunk), \quad v \in \mathbb{R}^d$$

*Hình 1.9. Biểu diễn vector Embedding*

Trong đó :

- E là mô hình embedding, có nhiệm vụ chuyển đổi dữ liệu văn bản đầu vào thành một vector trong không gian d chiều
- chunk là đoạn văn bản đầu vào sau quá trình phân đoạn (chunking).
- v là vector biểu diễn ngữ nghĩa của chunk.

- $\mathbb{R}^d$ biểu diễn không gian vector thực có d chiều.

### *1.8.4. Vector Database ( pgvector )*

Vector database là loại cơ sở dữ liệu được thiết kế để:

- Lưu trữ vector (embedding)
- Tìm kiếm nearest neighbors
- Tối ưu cho tìm kiếm ngữ nghĩa, gợi ý, recommendation, search trên media,....

pgvector là một tiện ích mở rộng (extension) mã nguồn mở của PostgreSQL, cho phép cơ sở dữ liệu này hoạt động như một Vector Database. Nó được thiết kế để lưu trữ, quản lý và tìm kiếm các vector embedding.

***Ưu điểm của việc sử dụng pgvector***

1. Tất cả trong một chỗ : Bạn không cần thêm một service DB riêng cho vector. Data dạng bảng + vector + metadata đều ở trong PostgreSQL → kiến trúc đơn giản, đỡ đau đầu vận hành.

2. Độ bền & tính sẵn sàng đã được chứng minh : PostgreSQL có hơn 30 năm phát triển: backup, replication, transaction ACID, security,… đều đã rất trưởng thành.

3. SQL đầy đủ tính năng như : JOIN giữa bảng metadata và bảng embedding,Filter theo trường (user_id, type, tag,…),Kết hợp query quan hệ + similarity search trong một câu SQL

4. Hiệu năng và khả năng scale tốt :Với index phù hợp, PostgreSQL + pgvector xử lý tốt trên hàng trăm nghìn đến hàng triệu vector (tuỳ cấu hình)

5. Mã nguồn mở & cộng đồng lớn :Bạn không bị vendor lock-in. Có thể tự deploy, tuỳ biến, audit, nghiên cứu

### *1.8.5. Truy xuất tài liệu ( Retrivel )*

#### *1.8.5.1. Dense Retrival ( Semantic Search )*

***Khái niệm***

Dense Retrieval là phương pháp truy xuất thông tin dựa trên biểu diễn ngữ nghĩa, được sử dụng trong các hệ thống Semantic Search và RAG. Phương pháp này sử dụng các mô hình embedding dựa trên kiến trúc Transformer (như BERT, RoBERTa hoặc OpenAI Embeddings) để mã hóa câu hỏi của người dùng (Query) và các đoạn tài liệu

(Document/Chunk) thành các vector dày đặc (Dense Vectors) trong không gian vector đa chiều

Mỗi vector embedding biểu diễn đặc trưng ngữ nghĩa của văn bản thông qua các giá trị số được mô hình học từ dữ liệu. Các vector của tài liệu được lưu trữ và sử dụng trong quá trình truy xuất, trong đó hệ thống tìm kiếm các vector tài liệu có mức độ tương đồng cao với vector của truy vấn.

Độ tương đồng giữa các vector thường được xác định bằng Cosine Similarity, cho phép đánh giá mức độ gần nhau về mặt ngữ nghĩa giữa Query và Document. Kết quả truy xuất sau đó được sử dụng làm ngữ cảnh đầu vào cho mô hình ngôn ngữ trong hệ thống RAG.

***Phép đo Độ tương đồng Cosine Similarity***

Về mặt toán học, độ tương đồng Cosine giữa vector truy vấn A và vector tài liệu B được định nghĩa là giá trị cosine của góc θ tạo bởi hai vector trong không gian đa chiều $\mathbb{R}^d$.

***Công thức tính Cosine Similarity***

$$\text{Cosine Similarity}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{d} A_i B_i}{\sqrt{\sum_{i=1}^{d} A_i^2}\sqrt{\sum_{i=1}^{d} B_i^2}}$$

*Hình 1.10. Công thức tính Cosine Similarity*

Trong đó:

- A · B là tích vô hướng (Dot Product) của hai vector.
- ||A|| và ||B|| lần lượt là độ dài Euclid (L2 Norm) của vector A và B.
- d là số chiều của vector nhúng ngữ nghĩa.

### *1.8.5.2. Spase Retrieval ( Keyword Search)*

***Khái niệm***

Sparse Retrieval (Truy xuất thưa thớt) là phương pháp tìm kiếm thông tin bằng cách biểu diễn văn bản dựa trên các từ ngữ cụ thể xuất hiện trong tài liệu. Đây là kỹ thuật nền tảng, đối nghịch và bổ khuyết trực tiếp cho Dense Retrieval trong các hệ thống Semantic Search và RAG hiện đại.

***Phương pháp Best Matching 25 (BM25 )***

BM25 (Best Matching 25) là một thuật toán xếp hạng tài liệu thuộc nhóm phương pháp truy xuất thưa (Sparse Retrieval), được sử dụng phổ biến trong các hệ thống tìm kiếm thông tin và RAG. Phương pháp này đánh giá mức độ liên quan giữa truy vấn (Query) và tài liệu dựa trên sự xuất hiện của các từ khóa, đồng thời xem xét tần suất xuất hiện của từ trong tài liệu, độ quan trọng của từ trong toàn bộ tập dữ liệu và độ dài của tài liệu.

Khác với phương pháp chỉ đếm số lần xuất hiện của từ khóa, BM25 sử dụng hai thành phần chính là Term Frequency (TF) và Inverse Document Frequency (IDF) để tính điểm liên quan giữa truy vấn và tài liệu. Ngoài ra, thuật toán còn áp dụng cơ chế chuẩn hóa độ dài tài liệu nhằm tránh việc ưu tiên các tài liệu dài chỉ vì chứa nhiều từ khóa hơn.

***Công thức tính điểm BM25***

$$\text{Score}(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i,D)(k_1+1)}{f(q_i,D) + k_1\left(1 - b + b\frac{|D|}{avgdl}\right)}$$

*Hình 1.11. Công thức tính điểm BM25*

Trong đó:

- tài liệu cần đánh giá.
- (Q): truy vấn của người dùng.
- (q_i): từ khóa thứ (i) trong truy vấn.
- (f(q_i,D)): số lần xuất hiện của từ khóa (q_i) trong tài liệu (D).(|D|): độ dài của tài liệu.
- (IDF(q_i)): mức độ quan trọng của từ khóa trong toàn bộ tập tài liệu.
- (|D|): độ dài của tài liệu.
- (avgdl): độ dài trung bình của các tài liệu.
- (k_1) và (b): các tham số điều chỉnh mức ảnh hưởng của tần suất từ khóa và độ dài tài liệu.

### *1.8.5.3. Hybrid Retrieval*

Hybrid Search là phương pháp kết hợp nhiều kỹ thuật truy xuất thông tin nhằm tận dụng ưu điểm của các phương pháp tìm kiếm khác nhau.

Trong hệ thống TripMate, Hybrid Search kết hợp Dense và Sparse để bù trừ khuyết điểm cho nhau :

· ***Dense Retrieval (Cosine Similarity):***

- Cung cấp khả năng hiểu ngữ nghĩa và tìm kiếm theo ý nghĩa.
- Hỗ trợ truy xuất các tài liệu có nội dung tương đồng nhưng khác từ khóa.

· ***Sparse Retrieval (BM25):***

- Cung cấp khả năng khớp chính xác từ khóa.
- Hiệu quả với các thuật ngữ đặc biệt, tên riêng, mã định danh hoặc thông tin cần độ chính xác cao.

### *1.8.5.4. Ranking Fusion ( RRF)*

Reciprocal Rank Fusion (RRF) là một phương pháp hợp nhất kết quả xếp hạng (ranking fusion) được sử dụng trong các hệ thống truy xuất thông tin, đặc biệt là Hybrid Retrieval trong RAG. RRF kết hợp nhiều danh sách kết quả truy xuất từ các phương pháp khác nhau (ví dụ: Lexical Retrieval như BM25 và Dense Retrieval như Vector Search) thành một danh sách xếp hạng duy nhất.

Thay vì kết hợp trực tiếp điểm số của các phương pháp truy xuất, RRF dựa trên vị trí xếp hạng (rank) của tài liệu trong từng danh sách kết quả. Điều này giúp tránh vấn đề khác biệt về thang điểm giữa các phương pháp truy xuất khác nhau.

***Công thức tính điểm RRF***

$$RRF(d) = \sum_{i=1}^{k} \frac{1}{K + r_i(d)}$$

*Hình 1.12. Công thức tính điểm Ranking Fusion*

Trong đó:

- (d): tài liệu cần tính điểm hợp nhất.
- (k): số lượng danh sách kết quả truy xuất được kết hợp.
- (r_i(d)): vị trí xếp hạng (rank) của tài liệu (d) trong danh sách kết quả thứ (i).
- (K): hằng số điều chỉnh, thường được chọn là 60.

### *1.8.5.5. Cross-Encoder Reranker*

Cross-Encoder là một kiến trúc mô hình học sâu (thường dựa trên Transformer như BERT, RoBERTa) được sử dụng để đánh giá mối quan hệ và mức độ tương đồng giữa hai đoạn văn bản với độ chính xác tối ưu nhất.

Trong các hệ thống Tìm kiếm ngữ nghĩa (Semantic Search) và RAG hiện đại, Cross-Encoder đóng vai trò là một Reranker (Bộ tái sắp xếp thứ hạng) ở giai đoạn cuối nhằm tinh lọc dữ liệu trước khi gửi cho LLM.

---

### *1.8.6. Agent & ReAct Agent*

#### *1.8.6.1. Agent trong hệ thống AI*

Agent là một thành phần trong hệ thống AI có khả năng nhận thông tin đầu vào, phân tích mục tiêu của người dùng, lập kế hoạch và lựa chọn công cụ phù hợp để thực hiện nhiệm vụ. Khác với mô hình ngôn ngữ chỉ sinh câu trả lời dựa trên dữ liệu đã được cung cấp, Agent có khả năng tương tác với các công cụ bên ngoài như cơ sở dữ liệu, hệ thống tìm kiếm, API hoặc các mô hình truy xuất thông tin.

Một Agent thường gồm các thành phần chính:

- LLM (Large Language Model): đóng vai trò là bộ suy luận và ra quyết oning: phân tích yêu cầu và xây dựng kế hoạch xử lý.
- Tool Selection: lựa chọn công cụ phù hợp với nhiệm vụ.
- Memory/State: lưu trữ trạng thái hoặc thông tin trong quá trình thực hiện.

Trong hệ thống RAG, Agent đóng vai trò điều phối, quyết định khi nào cần sử dụng Retrieval, truy vấn Database hoặc các công cụ khác.định.

#### *1.8.6.2. ReAct Agent*

ReAct (Reasoning and Acting) là một phương pháp xây dựng Agent kết hợp giữa quá trình suy luận (Reasoning) và hành động (Acting). Phương pháp này cho phép mô hình ngôn ngữ không chỉ tạo câu trả lời mà còn thực hiện các hành động thông qua việc gọi công cụ bên ngoài.

## 1.9. Xây dựng hệ thống gợi ý chuyến du lịch

### *1.9.1. Content-based*

Phương pháp lọc dựa trên nội dung (Content-based Filtering) đưa ra các gợi ý bằng cách phân tích các thuộc tính (thẻ tag, đặc điểm) của các địa điểm du lịch và so khớp chúng với hồ sơ sở thích (User Profile) được xây dựng từ lịch sử lựa chọn của người dùng đó. Nguyên lý cốt lõi là: "Nếu người dùng thích địa điểm A trong quá khứ, hệ thống sẽ gợi ý địa điểm B có các đặc trưng tương tự địa điểm A".

### *1.9.2. Collaborative Filtering*

Lọc cộng tác (Collaborative Filtering) là phương pháp gợi ý không phụ thuộc vào thuộc tính của địa điểm mà hoàn toàn dựa vào hành vi và tương tác lịch sử của cộng đồng người dùng trên hệ thống

Hệ thống tripmate dùng tương tác của chính user đó chứ không phải Collaborative Filtering thuần ( vốn cần tính độ tương đồng của các user qua ma trận user-item)

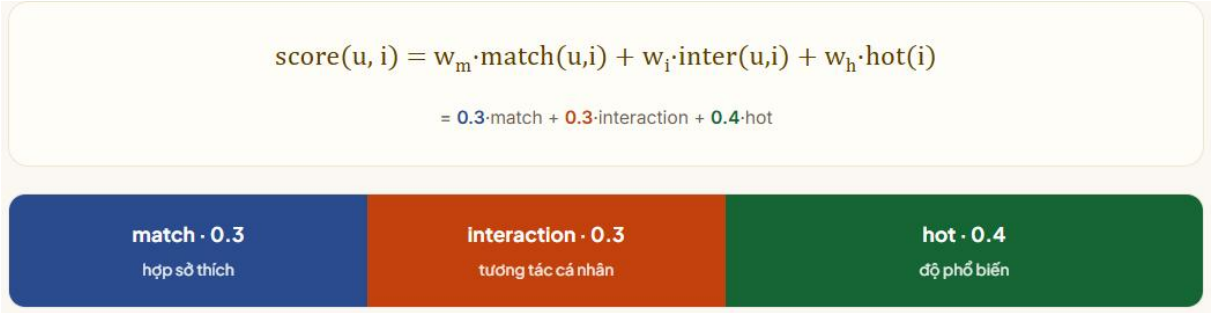
### *1.9.3. Popularity-based Filtering*

Lọc dựa trên độ phổ biến (Popularity-based Filtering) là phương pháp gợi ý phi cá nhân hóa (Non-personalized). Hệ thống tính toán và xếp hạng các địa điểm dựa trên tổng số lượt tương tác của toàn bộ cộng đồng trong một khoảng thời gian xác định. Đây là giải pháp tối ưu để giải quyết bài toán "Khởi đầu lạnh" (Cold-start Problem) khi hệ thống chưa có bất kỳ dữ liệu nào về người dùng mới.

### *1.9.4. Weighted Hybrid System*

Trong kiến trúc này, khi người dùng yêu cầu danh sách gợi ý, hệ thống sẽ kích hoạt đồng thời cả 3 bộ lọc độc lập: Bộ lọc sở thích , Bộ lọc tương tác , và Bộ lọc độ hot .

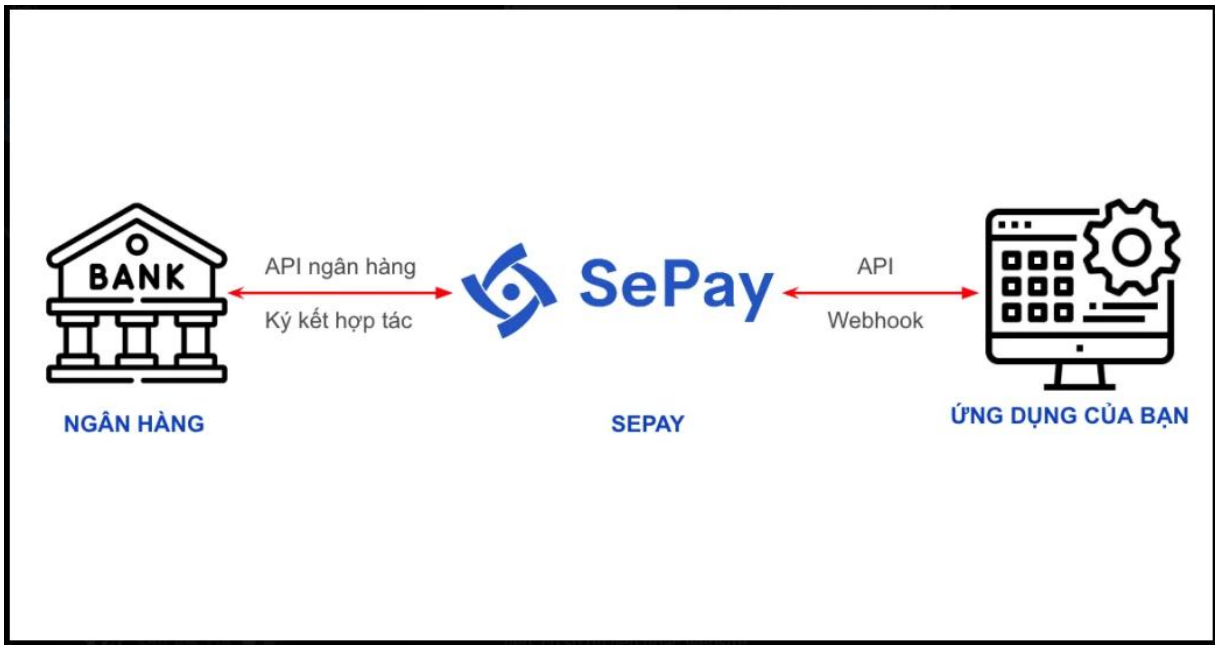
***Công thức tính tổng điểm***




*Hình 1.13. Công thức tính điểm Weighted*

## 1.10. Cổng thanh toán SePay

SePay là dịch vụ trung gian Việt Nam giúp tự động hoá ghi nhận giao dịch chuyển khoản ngân hàng.




*Hình 1.14. Cơ chế thanh toán Sepay*

Cơ chế hoạt động:

1. User chọn nạp X VNĐ → BE tạo mã NAP_XXXXXXXX duy nhất, lưu txn PENDING.
2. FE hiển thị QR + thông tin tài khoản MB Bank + nội dung chứa mã.
3. User chuyển khoản qua app ngân hàng.
4. SePay đọc giao dịch nhận về (qua kết nối ngân hàng), trích xuất mã từ description.
5. SePay POST webhook về BE TripMate kèm số tiền, mã giao dịch.
6. BE so khớp mã + số tiền với txn PENDING, nếu khớp → cộng tiền vào ví, đánh dấu SUCCESS, gửi notification.

Lợi thế: hoàn toàn tự động, không cần admin xác nhận thủ công

# CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 2.1. Phân tích yêu cầu

### *2.1.1 Xác định tác nhân của hệ thống*

Hệ thống TripMate được thiết kế với nhiều nhóm người dùng khác nhau, mỗi nhóm đóng vai trò và có quyền hạn riêng trong hệ thống, được gọi là các tác nhân (actors). Cụ thể, hệ thống bao gồm bốn tác nhân chính.

*Bảng 2.1. Xác định tác nhân hệ thống*

| STT | Tác nhân | Mô tả | Cách tham gia |
|---|---|---|---|
| 1 | Khách vãng lai | Là khách du lịch nhưng chưa có tài khoản trên hệ thống | Truy cập đường link website hoặc tải và vào ứng dụng |
| 2 | Khách du lịch | Là khách du lịch đã có tài khoản trên hệ thống | Đăng nhập vào hệ thống |
| 3 | Hướng dẫn viên | Hướng dẫn viên du lịch của hệ thống | |
| 4 | Quản trị viên | Quản lý và giám sát toàn bộ hệ thống | Đăng nhập vào hệ thống với quyền quản trị |

### *2.1.2. Xác định yêu cầu chức năng*

- Khách du lịch vãng lai(Guest) : người chưa đăng nhập vào hệ thống

*Bảng 2.2. Xác định yêu cầu khách vãng lai*

| Mã | Chức năng | Mô tả |
|---|---|---|
| G1 | Xem trang chủ | Xem trang chủ, chuyến đi , hướng dẫn viên |
| G2 | Xem danh sách/ chi tiết chuyến đi | Xem chi tiết chuyến đi, lịch trình, ảnh, thành viên (không thể tham gia) |
| G3 | Xem mạng xã hội | Bài viết , bình luận, lượt tương tác công khai |
| G4 | Xem hồ sơ hướng dẫn viên | Xem chi tiết hồ sơ hướng dẫn viên |

| | | |
|---|---|---|
| G5 | Tìm kiếm | Tìm kiếm chuyến đi, người dùng, địa điểm |
| G6 | Đăng nhập/đăng ký | Tạo tài khoản và đăng nhập vào hệ thống |

-Khách du lịch thành viên (Traveler) : người dùng đã có tài khoản và đăng nhập vào hệ thống

✧ Kế thừa toàn bộ quyền của Khách vãng lai (Guest) và bổ sung các chức năng sau

*Bảng 2.3. Xác định yêu cầu khách du lịch*

| Mã | Chức năng | Mô tả |
|---|---|---|
| T1 | Cập nhật hồ sơ cá nhân | Cập nhật ảnh đại diện, ảnh bìa, tiểu sử, sở thích và thông tin liên lạc |
| T2 | Đổi mật khẩu, đăng xuất | Đổi mật khẩu và đăng xuất khỏi hệ thống |
| T3 | Theo dõi | Theo dõi trang cá nhân người khác (follow) |
| T4 | Quản lý bài viết | Đăng bài viết, thả tim, bình luận, xóa, lưu |
| T5 | Tạo chuyến đi | Tạo chuyến đi, soạn lịch trình |
| T6 | Chỉnh sửa, hủy chuyến đi | Chỉnh sửa thông tin toàn bộ chuyến đi hoặc hủy chuyến đi ( vai trò owner) |
| T7 | Gửi yêu cầu tham gia chuyến đi | Gửi yêu cầu tham gia chuyến đi và đợi chấp nhận hoặc từ chối từ chủ chuyến |
| T8 | Chấp nhận yêu cầu | Chấp nhận yêu cầu tham gia của người khác (owner) |
| T9 | Đặt lịch | Đặt lịch hướng dẫn viên cho tour(owner) |
| T10 | Nạp tiền | Nạp tiền vào hệ thống qua ngân hàng |
| T11 | Đăng ký hồ sơ HDV | Điền thông tin theo mẫu và nộp hồ sơ chờ quản trị viên duyệt |
| T12 | Đánh giá | Đánh giá chuyến đi, đánh giá hướng dẫn |

| | | |
|---|---|---|
| | | viên |

-Hướng dẫn viên du lịch ( Guide ) : người đã được admin chấp nhận

✧Kế thừa toàn bộ quyền Travaler và bổ sung các chức năng sau :

*Bảng 2.4. Xác định yêu cầu hướng dẫn viên*

| Mã | Chức năng | Mô tả |
|---|---|---|
| H1 | Quản lý hồ sơ HDV | Quản lý thông tin được công khai( vùng hoạt động, chuyên môn, ngôn ngữ, chi phí thuê ,...) |
| H2 | Nhận yêu cầu đặt lịch | Xem danh sách đặt lịch của du khách |
| H3 | Xử lý yêu cầu đặt lịch | Chấp nhận hoặc từ chối yêu cầu của du khách |
| H4 | Quản lý lịch bận | Thêm , sửa, xóa lịch bận |
| H5 | Quản lý doanh thu | Xem dashboard toàn bộ doanh thu |
| H6 | Rút tiền | Gửi yêu cầu rút tiền và chờ duyệt |

- Quản trị viên (Admin ) : đăng nhập với vài trò quản trị viên

*Bảng 2.5. Xác định yêu cầu quản trị viên*

| Mã | Chức năng | Mô tả |
|---|---|---|
| A1 | Đăng nhập | Đăng nhập vài trang quản lý hệ thống |
| A2 | Thống kê tổng quan | Xem tổng quan hệ thống , lượng người dùng, chuyến đi, doanh thu, giao dịch |
| A3 | Duyệt hồ sơ hướng dẫn viên | Phê duyệt hoặc từ chối đơn đăng ký |
| A4 | Quản lý người dùng | Xem danh sách người dùng, khóa tài khoản |
| A5 | Nạp tiền | Nạp tiền thủ công cho người dùng , có thể nạp hàng loạt |

| | | |
|---|---|---|
| A6 | Quản lý địa điểm | Thêm,sửa, xóa địa điểm |
| A7 | Gửi thông báo hệ thống | Gửi thông báo đến một hoặc nhiều người dùng |
| A8 | Xem thông kê doanh thu chi tiết | Báo cáo doanh thu chi tiết của hướng dẫn viên |

### *2.1.3. Yêu cầu phi chức năng*

#### *2.1.3.1. Yêu cầu hiệu năng*

- API response time trung bình < 500ms cho 95% request (đo trên localhost).
- Trang chủ load < 2.5s với mạng 3G mô phỏng.
- Hỗ trợ tối thiểu 100 user đồng thời (đối với BE 1 instance, 4GB RAM).
- Lazy load ảnh và code splitting để giảm initial bundle.

#### *2.1.3.2. Yêu cầu bảo mật*

*Bảng 2.6. Yêu cầu bảo mật*

| Cơ chế | Mô tả | Mục đích |
|---|---|---|
| Bcrypt(salt rounds ≥ 10) | Mật khẩu được mã hóa bằng thuật toán bcrypt với số vòng salt từ 10 trở lên | Bảo vệ mật khẩu, chống giải mã khi dữ liệu bị lộ |
| JWT(HS256, secret ≥ 32 ký tự) | Token xác thực được ký bằng thuật toán HS256 và khóa bí mật đủ mạnh | Xác thực người dùng an toàn giữa client và server |
| Refresh Token Rotation | Refresh token được thay đổi sau mỗi lần sử dụng, không cho phép dùng lại | Giảm nguy cơ bị đánh cắp token và tái sử dụng |
| Input Validation | Kiểm tra dữ liệu đầu vào ở tất cả endpoint | Ngăn injection, lỗi dữ liệu và tăng tính an toàn hệ thống |
| CORS Whitelist | Chỉ cho phép các domain được cấu hình truy cập API | Hạn chế truy cập trái phép từ domain không tin cậy |
| Rate Limiting | Giới hạn số request trong một khoảng thời gian (đặc biệt ở login) | Chống tấn công và đoán mật khẩu |
| SePay Webhook | Xác thực webhook bằng | Đảm bảo chỉ SePay hợp lệ |

| | | |
|---|---|---|
| Authentication | API key trong header | mới được gửi dữ liệu về hệ thống |

#### *2.1.3.3. Yêu cầu khả dụng*

- Hoạt động trên Chrome, Firefox, Edge, Safari (web).
- Hoạt động trên Android 7.0+ qua APK.
- Hỗ trợ tiếng Việt với dấu Unicode đầy đủ.

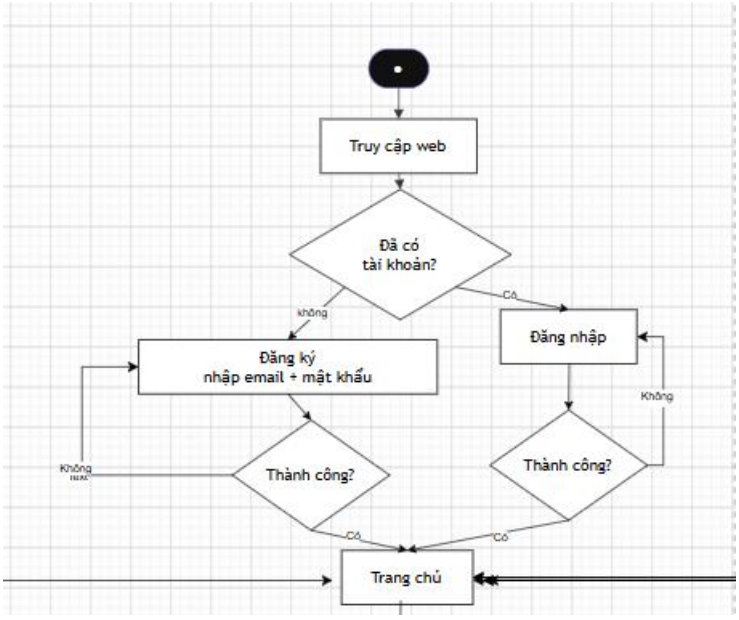
#### *2.1.3.4. Yêu cầu bảo trì và mở rộng*

- Code có TypeScript strict mode, comment ngắn gọn cho logic phức tạp.
- Module hoá rõ ràng để mở rộng tính năng mới không ảnh hưởng module cũ.
- Tài liệu API sinh tự động qua Swagger.
- Hỗ trợ migration DB khi chuyển sang production.

## 2.2. Phân tích hệ thống

### *2.2.1. Sơ đồ nguyên lý hoạt động*

#### *2.2.1.1. Luồng đăng nhập*




*Hình 2.1. Luồng đăng nhập*

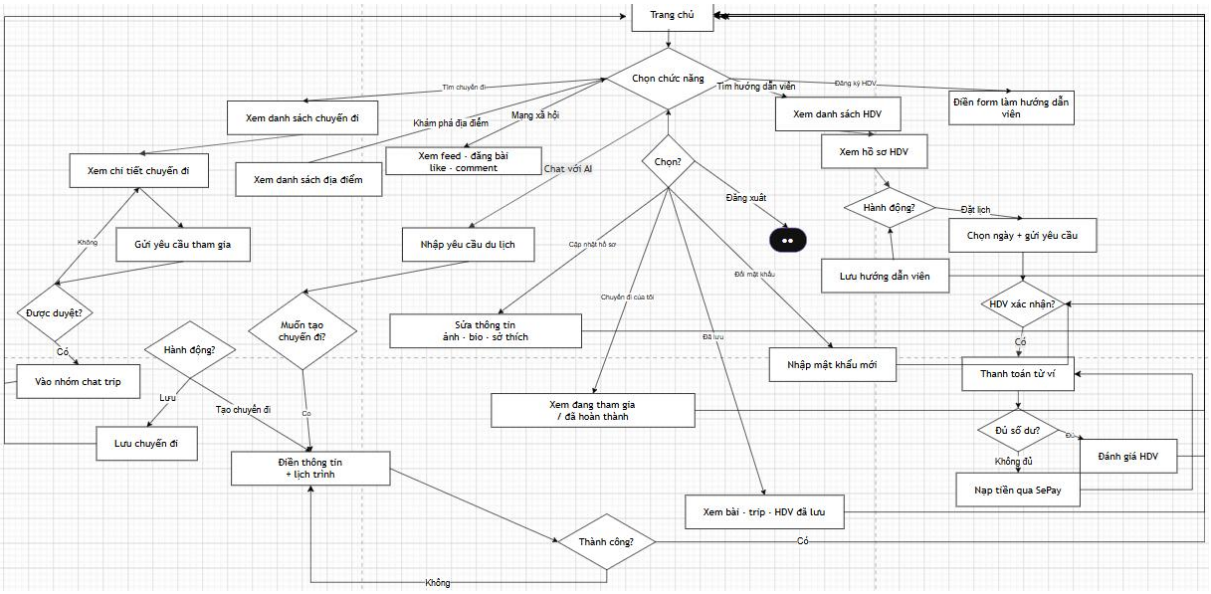
Khi người dùng truy cập vào website, hệ thống sẽ kiểm tra xem người dùng đã có tài khoản hay chưa. Nếu chưa có tài khoản, người dùng sẽ được chuyển đến chức năng Đăng ký, tại đây người dùng cần nhập các thông tin cần thiết như email và mật khẩu để tạo tài khoản mới. Sau khi gửi thông tin đăng ký, hệ thống tiến hành kiểm tra tính hợp lệ của dữ liệu. Nếu quá trình đăng ký thành công, người dùng sẽ được chuyển đến trang chủ hoặc thực hiện đăng nhập để sử dụng hệ thống. Trong trường hợp đăng ký

thất bại do dữ liệu không hợp lệ hoặc email đã tồn tại, hệ thống sẽ hiển thị thông báo lỗi và yêu cầu người dùng nhập lại thông tin.

Nếu người dùng đã có tài khoản, họ có thể thực hiện chức năng Đăng nhập bằng cách nhập email và mật khẩu đã đăng ký. Hệ thống sẽ xác thực thông tin đăng nhập với dữ liệu được lưu trữ trong cơ sở dữ liệu. Nếu thông tin chính xác, hệ thống tạo phiên đăng nhập và chuyển người dùng đến trang chủ. Ngược lại, nếu thông tin đăng nhập không hợp lệ, hệ thống sẽ thông báo lỗi và cho phép người dùng thực hiện đăng nhập lại.

Luồng này giúp đảm bảo chỉ những người dùng đã được xác thực mới có thể truy cập và sử dụng các chức năng yêu cầu quyền thành viên của hệ thống.

### *2.2.1.2. Luồng hoạt động của khách du lịch*




*Hình 2.2. Luồng hoạt động của khách du lịch*

Sau khi đăng nhập thành công và truy cập vào trang chủ, người dùng có thể lựa chọn nhiều chức năng khác nhau tùy theo nhu cầu sử dụng. Đối với việc khám phá và tham gia các chuyến đi, người dùng có thể xem danh sách các chuyến đi đang mở, xem chi tiết từng chuyến đi và gửi yêu cầu tham gia. Khi yêu cầu được chủ chuyến đi chấp thuận, người dùng sẽ được thêm vào nhóm chat của chuyến đi và có thể lưu chuyến đi vào danh sách yêu thích để tiện theo dõi.

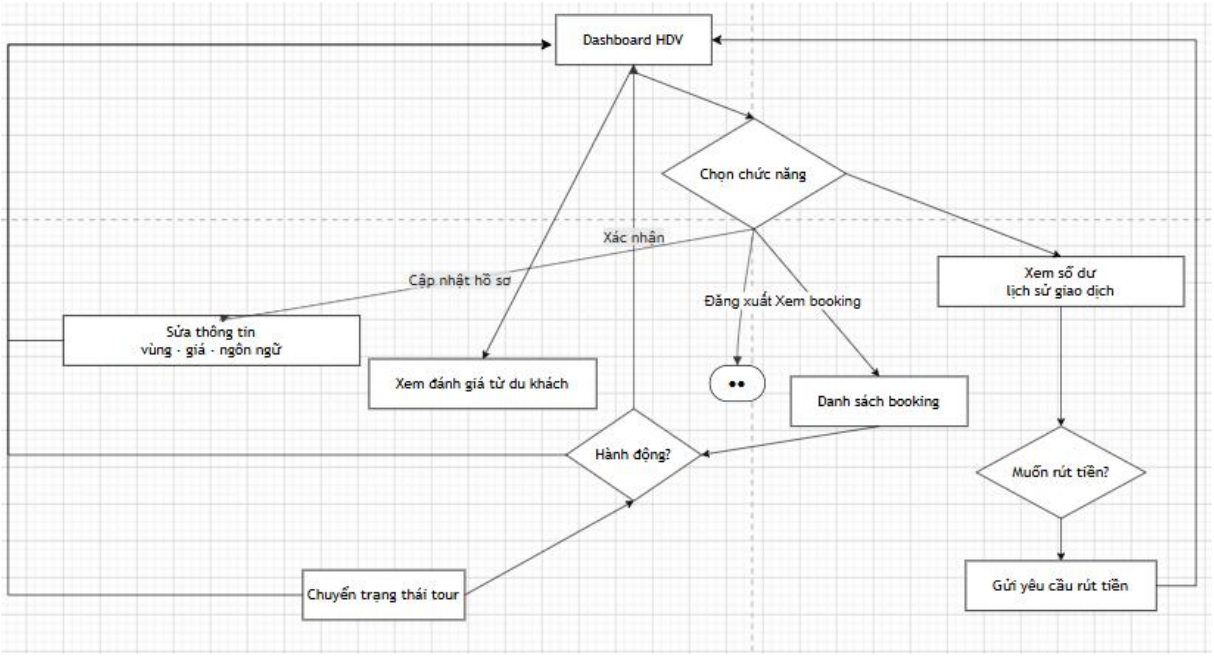
Bên cạnh đó, người dùng có thể tương tác với cộng đồng thông qua việc xem bảng tin (feed), đăng bài chia sẻ trải nghiệm du lịch, thích (like), bình luận hoặc chia

sẻ các bài viết. Hệ thống cũng cho phép người dùng nhập yêu cầu du lịch để nhận các gợi ý phù hợp từ chatbot AI, hỗ trợ tìm kiếm thông tin và lên kế hoạch cho chuyến đi.

Ngoài chức năng tham gia chuyến đi, người dùng còn có thể quản lý tài khoản cá nhân bằng cách cập nhật thông tin hồ sơ, thay đổi mật khẩu và theo dõi các chuyến đi đã tham gia hoặc đang theo dõi. Người dùng cũng có thể xem danh sách hướng dẫn viên, xem hồ sơ chi tiết và thực hiện đặt dịch vụ hướng dẫn viên. Khi gửi yêu cầu đặt dịch vụ, hướng dẫn viên sẽ xem xét và xác nhận hoặc từ chối yêu cầu. Nếu yêu cầu được chấp nhận, người dùng tiến hành thanh toán bằng số dư trong ví hoặc nạp thêm tiền thông qua hệ thống SePay. Sau khi giao dịch thành công, booking sẽ được xác nhận và hướng dẫn viên được thêm vào chuyến đi.

Nhìn chung, biểu đồ mô tả toàn bộ các hoạt động chính mà khách du lịch thành viên có thể thực hiện trên hệ thống TripMate, từ khám phá địa điểm, tham gia chuyến đi, tương tác cộng đồng, sử dụng AI hỗ trợ du lịch cho đến đặt dịch vụ hướng dẫn viên và quản lý tài khoản cá nhân.

### *2.2.1.3. Luồng hoạt động của hướng dẫn viên*




*Hình 2.3. Luồng hoạt động của hướng dẫn viên*

Sau khi đăng nhập thành công, hướng dẫn viên được chuyển đến màn hình Dashboard HDV để quản lý các hoạt động trên hệ thống. Tại đây, hướng dẫn viên có thể lựa chọn nhiều chức năng khác nhau tùy theo nhu cầu sử dụng.

Đối với việc quản lý hồ sơ cá nhân, hướng dẫn viên có thể cập nhật các thông tin như khu vực hoạt động, mức giá dịch vụ và ngôn ngữ hỗ trợ. Sau khi cập nhật, hệ thống sẽ lưu lại các thay đổi để phục vụ quá trình tìm kiếm và đặt dịch vụ của khách hàng.

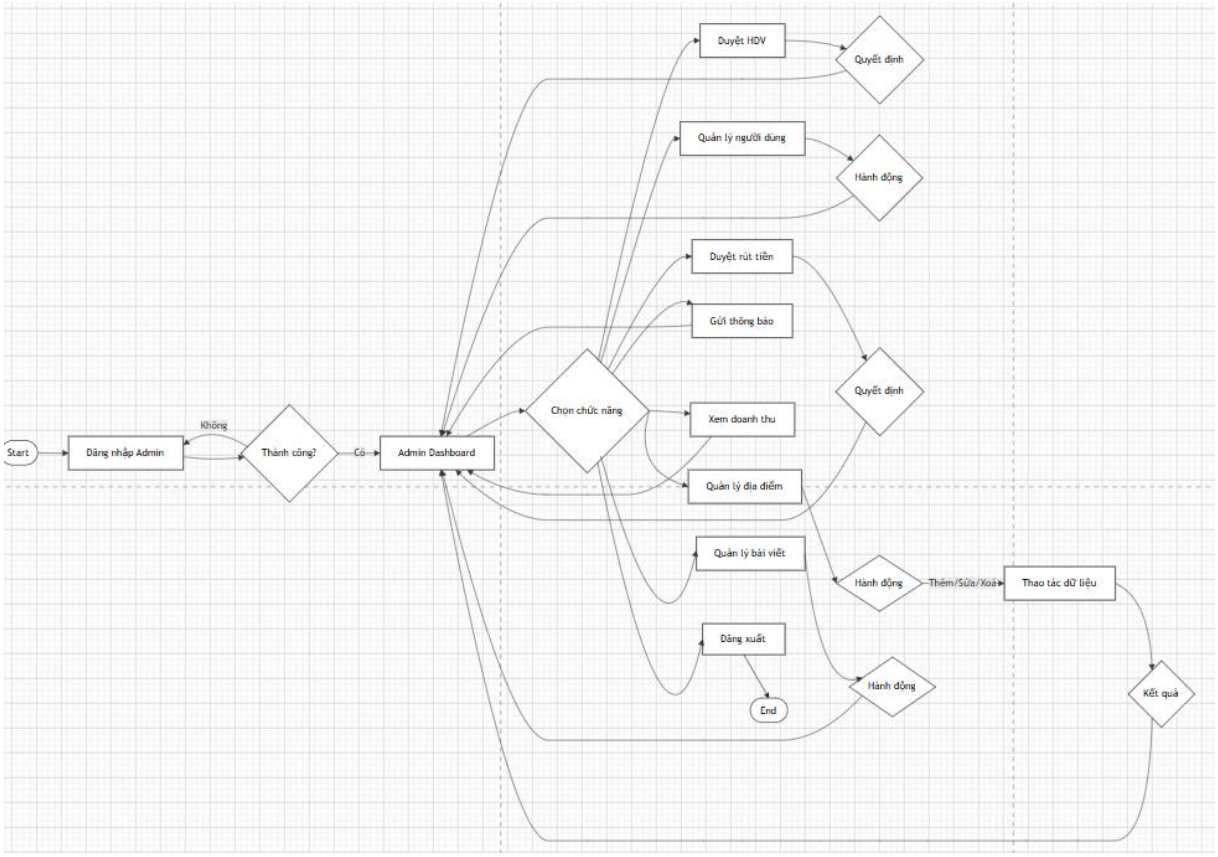
Hướng dẫn viên cũng có thể xem các đánh giá và nhận xét từ khách hàng đã sử dụng dịch vụ.

Trong quá trình hoạt động, hướng dẫn viên có thể truy cập danh sách các booking để theo dõi các yêu cầu đặt dịch vụ. Từ danh sách này, hướng dẫn viên thực hiện các hành động như xác nhận booking hoặc cập nhật trạng thái của tour khi cần thiết.

Ngoài ra, hệ thống cung cấp chức năng quản lý tài chính, cho phép hướng dẫn viên xem số dư hiện tại và lịch sử giao dịch. Khi có nhu cầu rút tiền, hướng dẫn viên gửi yêu cầu rút tiền đến hệ thống để được xử lý theo quy trình quy định.

Bên cạnh các chức năng trên, hướng dẫn viên có thể thực hiện các thao tác khác hoặc đăng xuất khỏi hệ thống khi kết thúc phiên làm việc.

### *2.2.1.4. Luồng hoạt động của quản trị viên*




*Hình 2.4. Luồng hoạt động của quản trị viên*

Khi đăng nhập thành công, quản trị viên được chuyển đến màn hình Dashboard để truy cập các chức năng quản trị.

Từ Dashboard, quản trị viên có thể lựa chọn các chức năng quản lý khác nhau. Đối với việc quản lý hướng dẫn viên, quản trị viên thực hiện xem xét hồ sơ đăng ký và đưa ra quyết định phê duyệt hoặc từ chối. Trong chức năng quản lý người dùng, quản trị viên có thể thực hiện các thao tác quản lý tài khoản theo quy định của hệ thống.

Ngoài ra, quản trị viên có quyền xem và xử lý các yêu cầu rút tiền từ hướng dẫn viên, đồng thời gửi thông báo đến người dùng hoặc hướng dẫn viên khi cần thiết. Hệ thống cũng cung cấp chức năng thống kê doanh thu nhằm hỗ trợ theo dõi tình hình hoạt động của nền tảng.

Đối với dữ liệu nội dung, quản trị viên có thể quản lý địa điểm và bài viết bằng các thao tác thêm mới, chỉnh sửa hoặc xóa dữ liệu. Sau khi thực hiện thao tác, hệ thống trả về kết quả xử lý để xác nhận thành công hoặc thất bại.

Khi hoàn tất các công việc quản trị, quản trị viên có thể đăng xuất để kết thúc phiên làm việc.

### *2.2.2. Sơ đồ ca sử dụng*

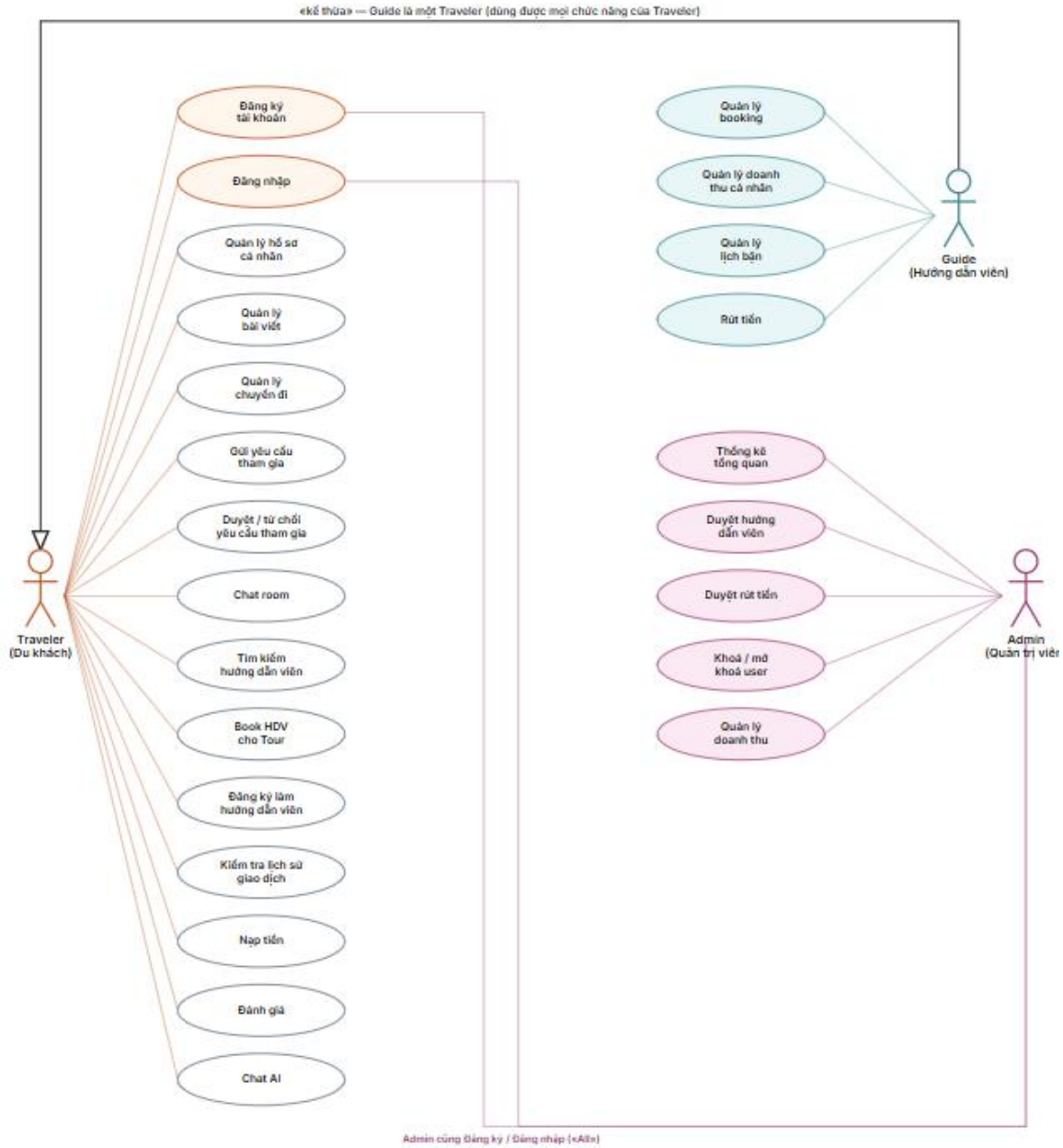
#### *2.2.2.1 Tổng quan chức năng hệ thống*

*Bảng 2.7. Tổng quan chức năng hệ thống*

| Use Case | Actor |
|---|---|
| Đăng ký tài khoản | All |
| Đăng nhập | All |
| Quản lý hồ sơ cá nhân | Traveler |
| Quản lý bài viết | Traveler |
| Quản lý chuyến đi | Traveler/Owner |
| Gửi yêu cầu tham gia | Traveler |
| Duyệt từ chối yêu cầu tham gia | Owner Trip |
| Chat room | Member Trip |
| Đăng ký làm hướng dẫn viên | Traveler |

| | |
|---|---|
| Tìm kiếm hướng dẫn viên | Traveler |
| Book HDV cho Tour | Owner Trip |
| Kiểm tra lịch sử giao dịch | Traveler |
| Nạp tiền | Traveler |
| Đánh giá | Traveler |
| Chat AI | Traveler, Guide |
| Quản lý booking | Guide |
| Quản lý doanh thu cá nhân | Guide |
| Quản lý lịch bận | Guide |
| Rút tiền | Guide |
| Thống kê tổng quan | Admin |
| Duyệt hướng dẫn viên | Admin |
| Duyệt rút tiền | Admin |
| Khoá / mở khoá user | Admin |
| Quản lý doanh thu | Admin |

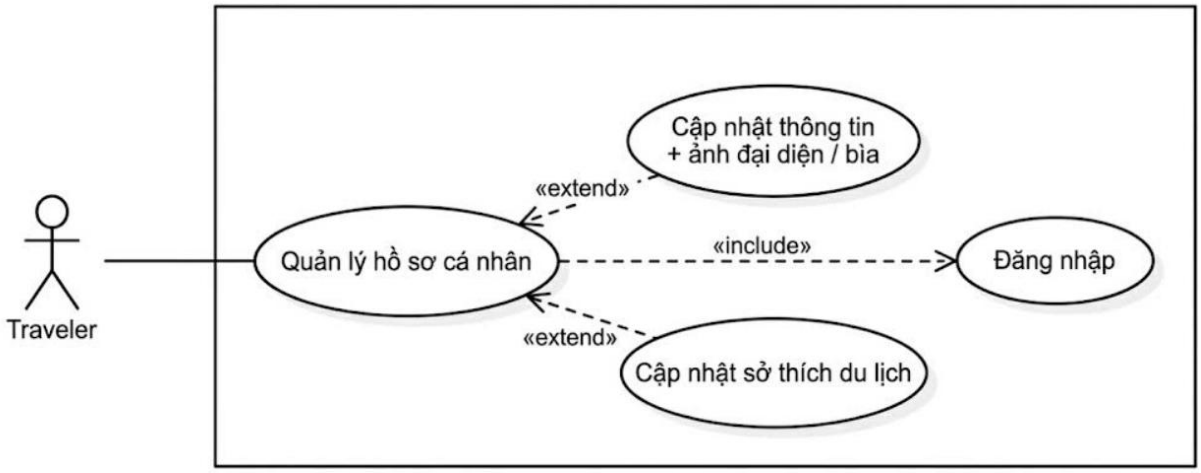
## *Sơ đồ Usecase tổng quan*




*Hình 2.5. Sơ đồ UseCase tổng quan*

### *2.4.2.2. Tổng quan chức năng khách du lịch*

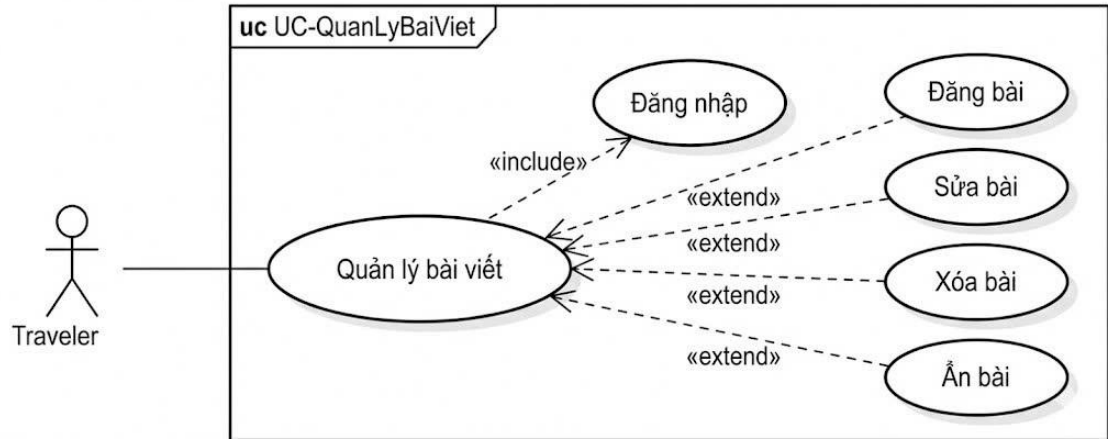
- Sơ đồ ca sử dụng chức năng chức năng quản lý hồ sơ cá nhân




*Hình 2.5. Sơ đồ ca sử dụng chức năng quản lý hồ sơ cá nhân*

| **Mã Usecase** | UC01 |
|---|---|
| **Tên Usecase** | Quản lý hồ sơ cá nhân |
| **Tác nhân** | Traveler (và các vai trò kế thừa: Guide, Admin |
| **Mô tả** | Cho phép người dùng xem và cập nhật thông tin cá nhân: tên, ảnh đại diện, ảnh bìa, tiểu sử, liên kết mạng xã hội và sở thích du lịch (phục vụ gợi ý cá nhân hoá). |
| **Sự kiện kích hoạt** | Người dùng truy cập trang hồ sơ và nhấn "Chỉnh sửa". |
| **Tiền điều kiện** | Người dùng đã đăng nhập (có access token hợp lệ). |
| **Hậu điều kiện** | Thông tin hồ sơ được cập nhật trong CSDL; hồ sơ hiển thị mới được áp dụng. |
| **Luồng sự kiện chính** | 1. Người dùng mở trang hồ sơ cá nhân.<br>2.Hệ thống hiển thị thông tin hiện tại.<br>3.Người dùng nhấn "Chỉnh sửa", sửa các trường (tên, bio, ảnh, sở thích…).<br>4.Người dùng nhấn "Lưu".<br>5. Hệ thống validate dữ liệu (PUT /users/me).<br>6. Hệ thống lưu vào CSDL và trả về hồ sơ mới.<br>7. Giao diện cập nhật, hiển thị thông báo thành công. |
| **Luồng sự kiện thay thế** | 1. Người dùng cập nhật ảnh: hệ thống upload ảnh lên Firebase Storage trước, lấy URL rồi mới lưu hồ sơ.<br>2. Người dùng huỷ chỉnh sửa: hệ thống bỏ thay đổi, giữ thông tin cũ. |
| **Luồng sự kiện ngoại lệ** | Token hết hạn → hệ thống trả 401, yêu cầu đăng nhập lại. |

- Sơ đồ ca sử dụng chức năng quản lý bài viết

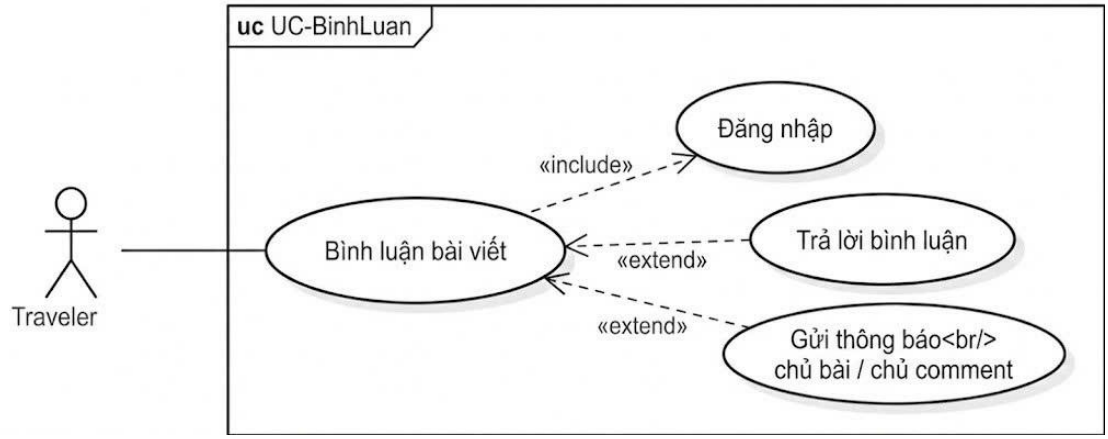



*Hình 2.6. Sơ đồ ca sử dụng chức năng quản lý bài viết*

| **Mã Usecase** | UC02 |
|---|---|
| **Tên Usecase** | Quản lý bài viết (feed) |
| **Tác nhân** | Traveler/Guide |
| **Mô tả** | Cho phép người dùng đăng, chỉnh sửa, xoá bài viết chia sẻ trải nghiệm kèm ảnh; xem feed theo các chế độ (dành cho bạn / đang theo dõi / xu hướng). |
| **Sự kiện kích hoạt** | Người dùng nhấn "Đăng bài" hoặc mở trang Feed. |
| **Tiền điều kiện** | Người dùng đã đăng nhập (có access token hợp lệ). |
| **Hậu điều kiện** | Bài viết được lưu và hiển thị trên feed; bộ đếm bài viết của người dùng tăng. |
| **Luồng sự kiện chính** | 1. Người dùng nhấn "Đăng bài".<br>2. Nhập tiêu đề, nội dung, địa điểm, chọn ảnh, chọn quyền xem (public/friends).<br>3. Nhấn "Đăng".<br>4. Hệ thống upload ảnh, validate dữ liệu (POST |

| | |
|---|---|
| | /posts).<br>5. Hệ thống lưu bài viết, trả về và chèn lên đầu feed.. |
| **Luồng sự kiện thay thế** | 1.Sửa bài: người dùng chọn "Chỉnh sửa" → cập nhật (PUT /posts/:id).<br>2.Xoá bài: người dùng chọn "Xoá" → xác nhận → DELETE /posts/:id |
| **Luồng sự kiện ngoại lệ** | Token hết hạn → hệ thống trả 401, yêu cầu đăng nhập lại.<br>Nội dung rỗng/ảnh sai định dạng → báo lỗi 400, không đăng.<br>Người dùng không phải tác giả khi sửa/xoá → hệ thống từ chối 403. |

- Sơ đồ ca sử dụng chức năng tương tác bài viết

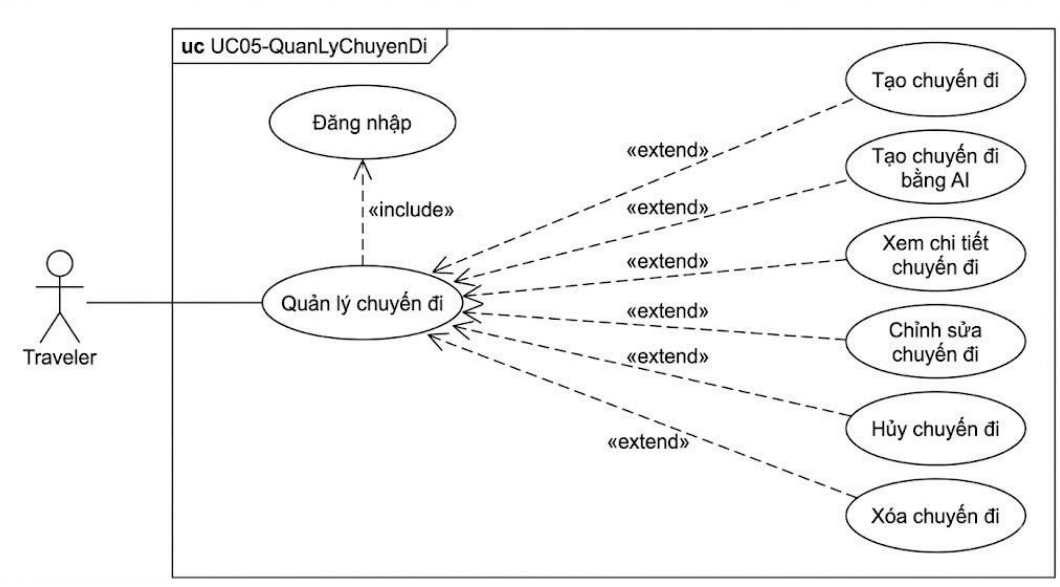



*Hình 2.7. Sơ đồ ca sử dụng chức năng tương tác bài viết*

| | |
|---|---|
| **Mã Usecase** | UC03 |
| **Tên Usecase** | Tương tác bài viết |
| **Tác nhân** | Traveler/Guide |

| | |
|---|---|
| **Mô tả** | Cho phép người dùng thích (like), bình luận (có trả lời lồng nhau), chia sẻ bài viết và theo dõi tác giả. |
| **Sự kiện kích hoạt** | Người dùng nhấn nút like / comment / share / follow trên một bài viết. |
| **Tiền điều kiện** | Người dùng đã đăng nhập; bài viết tồn tại và người dùng có quyền xem. |
| **Hậu điều kiện** | Tương tác được ghi nhận; bộ đếm tương ứng cập nhật; tác giả nhận thông báo realtime |
| **Luồng sự kiện chính** | 1. Người dùng nhấn "Thích".<br>2.Hệ thống ghi lượt like (POST /posts/:id/like), tăng like_count.<br>3.Hệ thống đẩy thông báo realtime tới tác giả.<br>4. Giao diện cập nhật trạng thái đã thích. |
| **Luồng sự kiện thay thế** | 1.Bình luận: người dùng nhập nội dung → POST /posts/:id/comments; có thể trả lời bình luận khác qua parent_id.<br>2. Bỏ thích: nhấn lại → huỷ like, giảm bộ đếm.<br>3. Theo dõi tác giả: POST /users/:id/follow. |
| **Luồng sự kiện ngoại lệ** | Token hết hạn → hệ thống trả 401, yêu cầu đăng nhập lại.<br>Bài viết đã bị xoá → hệ thống báo 404. |

- Sơ đồ ca sử dụng chức năng quản lý chuyến đi

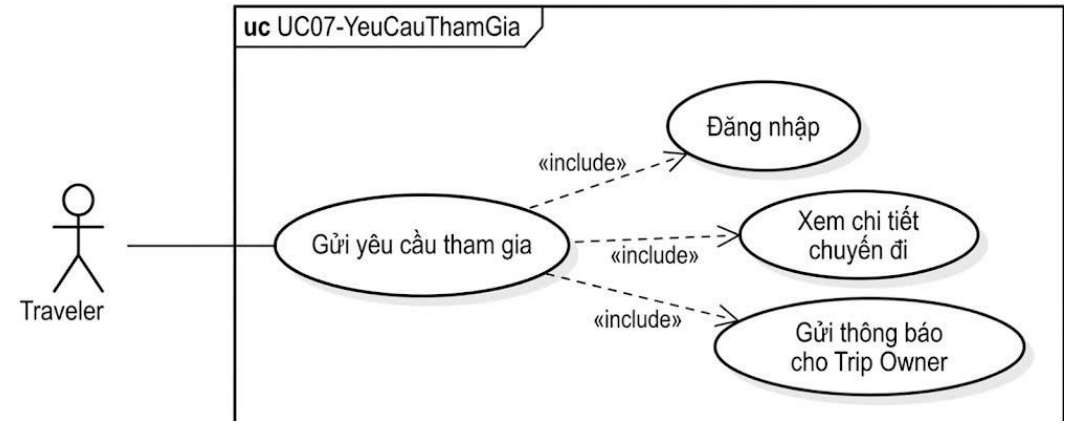



*Hình 2.8. Sơ đồ ca sử dụng chức năng quản lý chuyến đi*

| **Mã Usecase** | UC04 |
|---|---|
| **Tên Usecase** | Quản lý chuyến đi |
| **Tác nhân** | Traveler (chủ chuyến — Owner) |
| **Mô tả** | Cho phép người dùng tạo, chỉnh sửa, xoá chuyến đi; soạn lịch trình từng ngày, thêm gallery ảnh. Khi tạo chuyến, hệ thống tự lập nhóm chat. |
| **Sự kiện kích hoạt** | Người dùng nhấn "Tạo chuyến đi". |
| **Tiền điều kiện** | Người dùng đã đăng nhập |
| **Hậu điều kiện** | Chuyến đi được lưu với người tạo là trưởng nhóm; nhóm chat được tạo tự động. |
| **Luồng sự kiện chính** | 1. Người dùng nhập tiêu đề, điểm xuất phát, điểm đến, ngày, số thành viên tối đa, danh mục, ảnh bìa.<br>2. Soạn lịch trình từng ngày (hoạt động theo giờ).<br>3. Nhấn "Tạo".<br>4. Hệ thống validate, kiểm tra trùng ngày với chuyến |

| | |
|---|---|
| | đã tham gia.<br>5. Hệ thống lưu chuyến (POST /trips), thêm người tạo làm leader.<br>6. Hệ thống tạo nhóm chat cho chuyến.<br>7. Chuyển tới trang chi tiết chuyến. |
| **Luồng sự kiện thay thế** | 1.Chỉnh sửa chuyến: chọn các trường cần sửa, còn lại giữ nguyên (PUT /trips/:id).<br>2. Xoá/huỷ chuyến: chủ chuyến chọn "Huỷ" → POST /trips/:id/cancel. |
| **Luồng sự kiện ngoại lệ** | 1. Ngày chuyến trùng với chuyến đang tham gia → hệ thống chặn (400, nêu tên chuyến trùng).<br>2. Thiếu trường bắt buộc → báo lỗi validate 400.<br>3. Người sửa/xoá không phải chủ chuyến → từ chối 403. |

- Sơ đồ ca sử dụng chức năng gửi yêu cầu tham gia chuyến đi



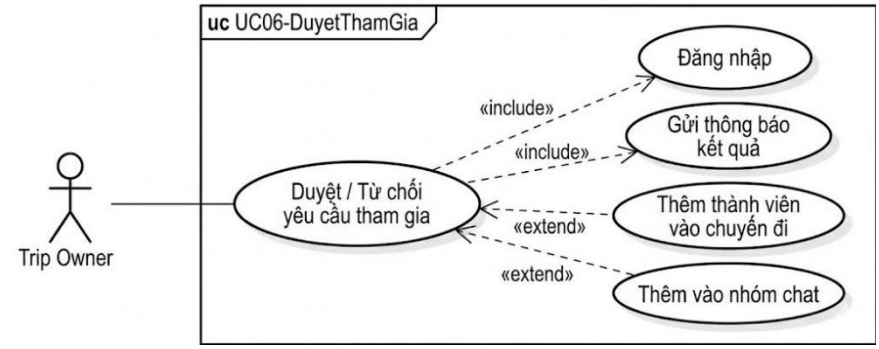

*Hình 2.9. Sơ đồ ca sử dụng chức năng gửi yêu cầu tham gia*

| **Mã Usecase** | UC05 |
|---|---|
| **Tên Usecase** | Gửi yêu cầu tham gia chuyến đi |

| | |
|---|---|
| **Tác nhân** | Traveler |
| **Mô tả** | Cho phép người dùng gửi yêu cầu xin tham gia một chuyến đi công khai kèm lời nhắn; chủ chuyến sẽ duyệt sau |
| **Sự kiện kích hoạt** | Người dùng nhấn "Tham gia" ở trang chi tiết chuyến đi |
| **Tiền điều kiện** | Người dùng đã đăng nhập,chưa là thành viên; chuyến chưa đầy. |
| **Hậu điều kiện** | Yêu cầu được tạo ở trạng thái "chờ duyệt"; chủ chuyến nhận thông báo; bộ đếm tương tác/độ hot của chuyến tăng. |
| **Luồng sự kiện chính** | 1. Người dùng mở chi tiết chuyến đi.<br>2.Nhấn "Tham gia", nhập lời nhắn (tuỳ chọn).<br>3.Hệ thống kiểm tra: chưa là thành viên, chuyến chưa đầy, không trùng ngày.<br>4.Hệ thống tạo yêu cầu trạng thái pending (POST /trips/:id/join).<br>5.Tăng request_count, đánh dấu tương tác của người dùng.<br>6. Đẩy thông báo tới chủ chuyến. |
| **Luồng sự kiện thay thế** | Đã gửi yêu cầu trước đó (đã từ chối/huỷ) → hệ thống đặt lại trạng thái về pending thay vì tạo mới. |
| **Luồng sự kiện ngoại lệ** | 1. Chuyến đã đầy thành viên → báo lỗi "Chuyến đã đủ người".<br>2. Người dùng là chủ chuyến/đã là thành viên → báo lỗi, không cho gửi. |

| | |
|---|---|
| | 3. Ngày chuyến trùng chuyến đang tham gia → cảnh báo/chặn. |

- Sơ đồ ca sử dụng chức năng xử lý yêu cầu tham gia của member

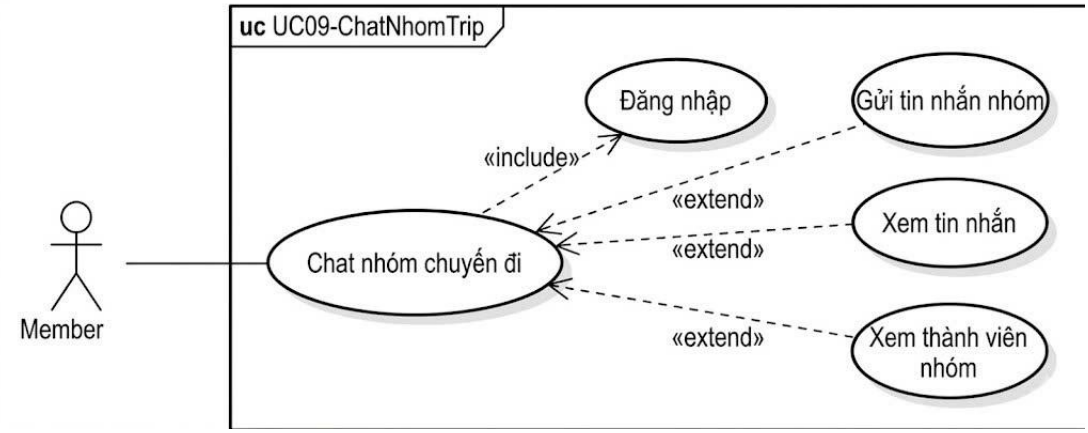



*Hình 2.10. Sơ đồ ca sử dụng chức năng xử lý yêu cầu tham gia*

| **Mã Usecase** | UC06 |
|---|---|
| **Tên Usecase** | Duyệt / từ chối yêu cầu tham gia |
| **Tác nhân** | Traveler (chủ chuyến — Owner) |
| **Mô tả** | Cho phép chủ chuyến xem danh sách yêu cầu tham gia và chấp nhận hoặc từ chối từng yêu cầu. |
| **Sự kiện kích hoạt** | Chủ chuyến xem chi tiết chuyến đi phía dưới sẽ thấy yêu cầu tham gia của chuyến mình tạo. |
| **Tiền điều kiện** | Người dùng đã đăng nhập. Người dùng là chủ chuyến; tồn tại yêu cầu ở trạng thái pending. |
| **Hậu điều kiện** | Nếu chấp nhận: người gửi trở thành thành viên, được thêm vào nhóm chat, chuyến tăng số thành viên. Người gửi nhận thông báo kết quả. |
| **Luồng sự kiện chính** | 1. Chủ chuyến mở danh sách yêu cầu.<br>2. Xem thông tin người gửi và lời nhắn.<br>3. Nhấn "Chấp nhận" |

| | |
|---|---|
| | 4.Hệ thống kiểm tra chuyến chưa đầy (POST /trips/:id/requests/:reqId/accept).<br>5. Thêm người gửi vào trip_members, tăng member_count, thêm vào nhóm chat.<br>6. Cập nhật yêu cầu sang accepted, đẩy thông báo cho người gửi. |
| **Luồng sự kiện thay thế** | 1. Từ chối: chủ chuyến nhấn "Từ chối" → yêu cầu chuyển rejected, thông báo người gửi (/reject).<br>2. Chuyến đã đầy khi duyệt → hệ thống báo lỗi, không thêm thành viên. |
| **Luồng sự kiện ngoại lệ** | Người thao tác không phải chủ chuyến → từ chối 403. 403. |

- Sơ đồ ca sử dụng chức năng chat

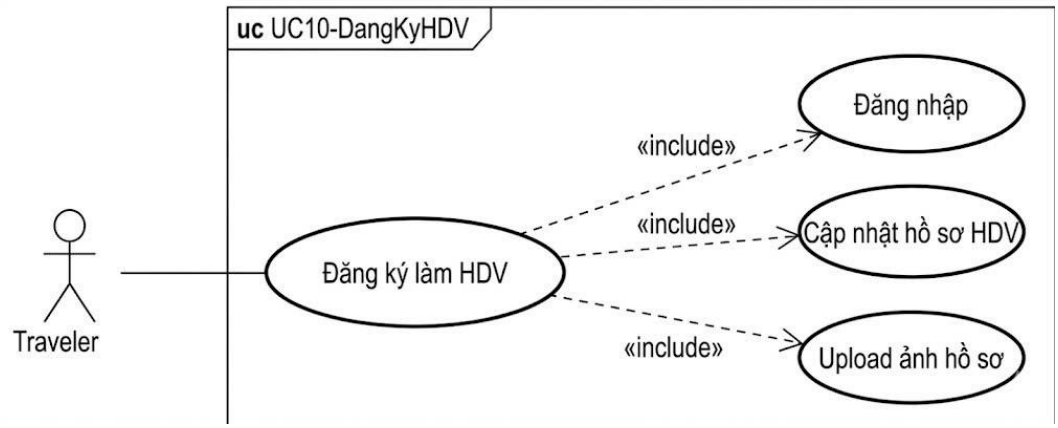



*Hình 2.11. Sơ đồ ca sử dụng chức năng chat nhóm*

| **Mã Usecase** | UC07 |
|---|---|
| **Tên Usecase** | Chat nhóm chuyến đi |
| **Tác nhân** | Member Trip (thành viên chuyến đi) |
| **Mô tả** | Cho phép các thành viên trong một chuyến đi trao đổi qua nhóm chat realtime (gửi văn bản, ảnh đính kèm), |

| | |
|---|---|
| | với trạng thái đã gửi/đã nhận/đã đọc. |
| **Sự kiện kích hoạt** | Thành viên mở nhóm chat của chuyến đi. |
| **Tiền điều kiện** | Người dùng là thành viên của chuyến; nhóm chat đã tồn tại (tạo tự động khi tạo chuyến). |
| **Hậu điều kiện** | Tin nhắn được lưu và phát realtime tới các thành viên đang kết nối. |
| **Luồng sự kiện chính** | 1. Thành viên mở nhóm chat.<br>2. Hệ thống tải lịch sử tin nhắn.<br>3. Thành viên nhập nội dung, nhấn gửi.<br>4. Client phát sự kiện send_message qua Socket.IO.<br>5. Server lưu tin nhắn, cập nhật "tin nhắn cuối".<br>6. Server phát tin tới các thành viên khác trong room realtime. |
| **Luồng sự kiện thay thế** | 1 . Gửi ảnh: upload ảnh trước, gửi kèm attachment.<br>2 . Thành viên gõ: client phát typing để hiển thị "đang nhập".<br>3 . Thành viên đọc: phát mark_as_read, cập nhật trạng thái đã đọc. |
| **Luồng sự kiện ngoại lệ** | 1. Mất kết nối WebSocket → Socket.IO tự fallback long-polling và thử kết nối lại.<br>2. Người dùng không phải thành viên → không được vào room (từ chối). |

- Sơ đồ ca sử dụng chức năng đăng ký làm hướng dẫn viên

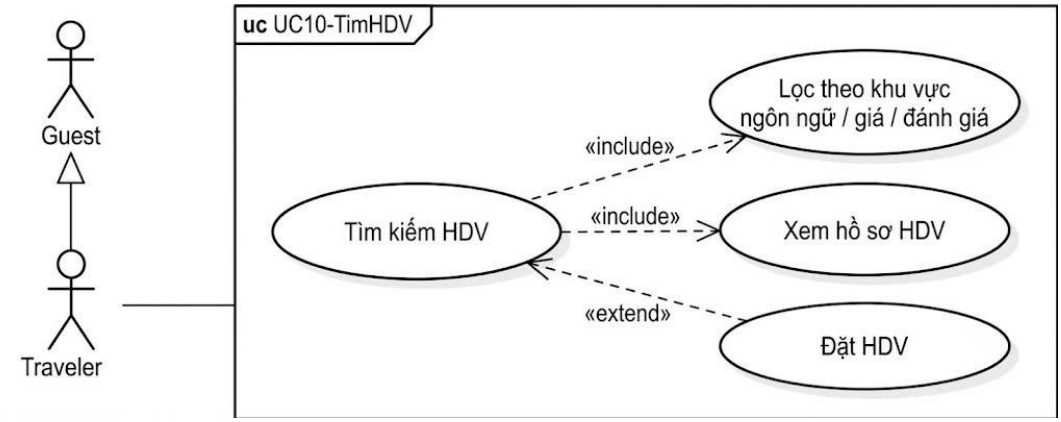



*Hình 2.12. Sơ đồ ca sử dụng chức năng đăng ký làm HDV*

| **Mã Usecase** | UC08 |
|---|---|
| **Tên Usecase** | Đăng ký làm hướng dẫn viên |
| **Tác nhân** | Traveler |
| **Mô tả** | Cho phép du khách nộp hồ sơ xin trở thành hướng dẫn viên (vùng hoạt động, chuyên môn, ngôn ngữ, giá, kinh nghiệm). Hồ sơ chờ Admin duyệt. |
| **Sự kiện kích hoạt** | Người dùng nhấn "Đăng ký làm hướng dẫn viên". |
| **Tiền điều kiện** | Người dùng đã đăng nhập; chưa có hồ sơ HDV đang chờ/đã duyệt |
| **Hậu điều kiện** | Hồ sơ HDV được tạo ở trạng thái pending; chờ Admin xét duyệt |
| **Luồng sự kiện chính** | 1. Người dùng mở form đăng ký HDV.<br>2. Nhập vùng hoạt động, chuyên môn, ngôn ngữ, giá/ngày, kinh nghiệm, giới thiệu.<br>3. Nhấn "Gửi đăng ký".<br>4. Hệ thống validate dữ liệu (POST /guides/apply).<br>5. Hệ thống tạo hồ sơ trạng thái pending. |

| | |
|---|---|
| | 6. Hiển thị thông báo "Đang chờ duyệt". |
| **Luồng sự kiện thay thế** | Hồ sơ đã từng bị từ chối: người dùng chỉnh sửa và nộp lại → đặt lại trạng thái pending. |
| **Luồng sự kiện ngoại lệ** | 1. Người dùng đã là HDV hoặc đang có hồ sơ chờ duyệt → hệ thống báo lỗi, không tạo trùng.<br>2. Thiếu trường bắt buộc → báo lỗi validate 400. |

- Sơ đồ ca sử dụng chức năng tìm kiếm hướng dẫn viên

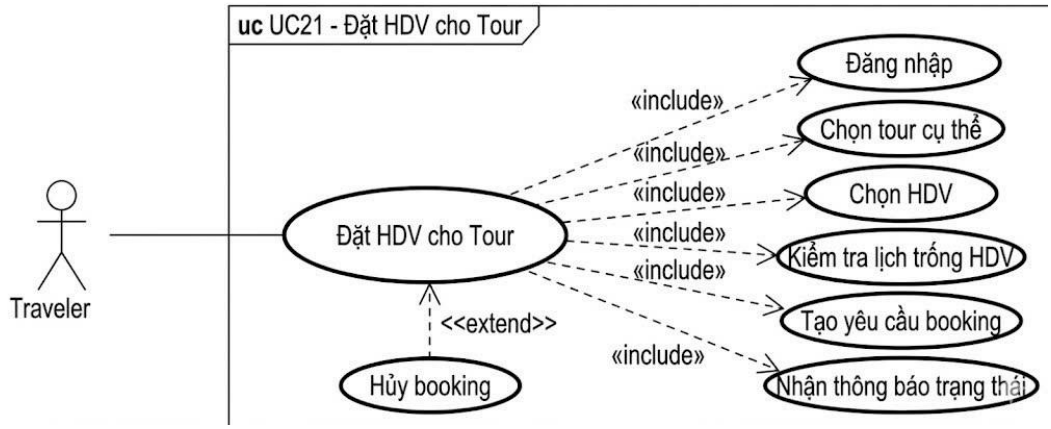



*Hình 2.13. Sơ đồ ca sử dụng chức năng tìm kiếm HDV*

| **Mã Usecase** | UC09 |
|---|---|
| **Tên Usecase** | Tìm kiếm hướng dẫn viên |
| **Tác nhân** | Traveler |
| **Mô tả** | Cho phép người dùng tìm và lọc hướng dẫn viên theo vùng, danh mục, ngôn ngữ, khoảng giá, khoảng ngày rảnh; xem hồ sơ chi tiết và đánh giá |
| **Sự kiện kích hoạt** | Người dùng mở trang "Hướng dẫn viên" hoặc nhập bộ lọc. |
| **Tiền điều kiện** | Chỉ cần vào web không bắt buộc đăng nhập (route công khai) |

| | |
|---|---|
| **Hậu điều kiện** | Hệ thống trả về danh sách HDV đã duyệt, khớp tiêu chí lọc. |
| **Luồng sự kiện chính** | 1. Người dùng mở danh sách HDV.<br>2. Chọn bộ lọc: vùng, danh mục, ngôn ngữ, giá, khoảng ngày.<br>3. Hệ thống truy vấn HDV trạng thái approved khớp tiêu chí (GET /guides).<br>4. Hiển thị danh sách kèm giá, đánh giá, trạng thái rảnh.<br>5. Người dùng nhấn một HDV để xem chi tiết (GET /guides/:id). |
| **Luồng sự kiện thay thế** | 1. Lọc theo khoảng ngày: hệ thống loại HDV đã có booking trùng khoảng đó.<br>2. Người dùng xem lịch bận của HDV (GET /guides/:id/busy-dates). |
| **Luồng sự kiện ngoại lệ** | Không có HDV khớp tiêu chí → hiển thị danh sách rỗng kèm gợi ý nới lỏng bộ lọc. |

- Sơ đồ ca sử dụng chức năng đặt lịch hướng dẫn viên cho tour

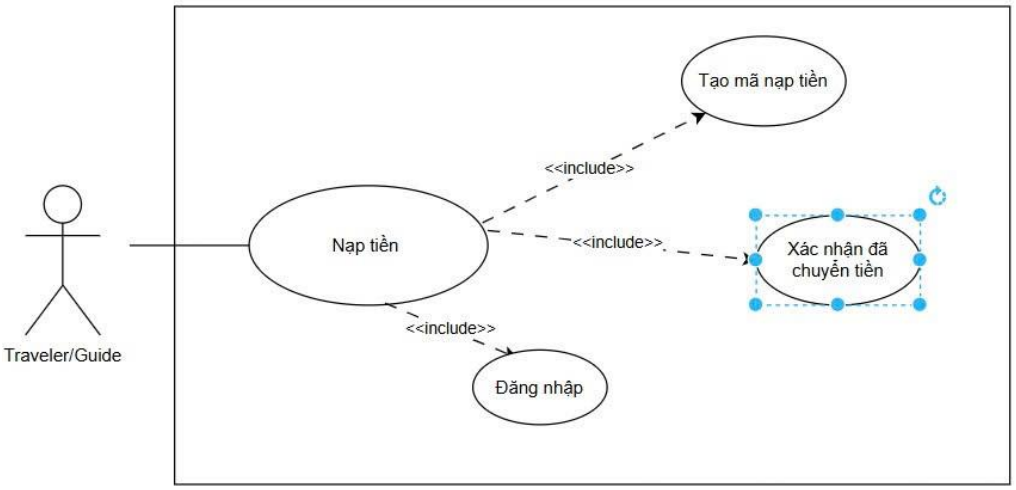



*Hình 2.14. Sơ đồ ca sử dụng đặt lịch hướng dẫn viên cho tour*

| **Mã Usecase** | UC10 |
|---|---|
| **Tên Usecase** | Đặt lịch hướng dẫn viên cho tour |
| **Tác nhân** | Traveler(Owner) |
| **Mô tả** | Cho phép du khách đặt một hướng dẫn viên cho khoảng ngày cụ thể (có thể gắn với một chuyến đi), tạo đơn booking chờ HDV xác nhận. |
| **Sự kiện kích hoạt** | Người dùng nhấn "Đăng ký làm hướng dẫn viên". |
| **Tiền điều kiện** | Người dùng đã đăng nhập, đang tham gia một chuyến đi chưa khởi hành, nhấn thuê HDV đã được duyệt và rảnh trong khoảng ngày chọn |
| **Hậu điều kiện** | Người dùng đã đăng nhập, đang tham gia một chuyến đi chưa khởi hành, HDV đã được duyệt và rảnh trong khoảng ngày chọn và<br>Đơn booking được tạo ở trạng thái "chờ HDV xác nhận"; HDV nhận thông báo. |
| **Luồng sự kiện chính** | 1. Người dùng mở hồ sơ HDV, nhấn "Đặt".<br>2. Chọn ngày bắt đầu, số ngày, số khách, (tuỳ chọn) gắn chuyến đi.<br>3. Hệ thống tính tổng tiền theo giá/ngày.<br>4. Người dùng xác nhận (POST /guides/bookings).<br>5. Hệ thống kiểm tra HDV rảnh, tạo booking trạng thái pending_acceptance.<br>6. Đẩy thông báo tới HDV. |
| **Luồng sự kiện thay thế** | Đặt từ trong một chuyến đi: hệ thống điền sẵn điểm |

| | |
|---|---|
| | đến và ngày từ chuyến (POST /trips/:id/hire-guide). |
| **Luồng sự kiện ngoại lệ** | 1. HDV đã bận khoảng ngày đó → báo lỗi, không tạo booking.<br>2. Người dùng tự đặt chính mình → từ chối.<br>3. HDV không tồn tại/chưa duyệt → báo lỗi. |

- Sơ đồ ca sử dụng chức năng nạp tiền

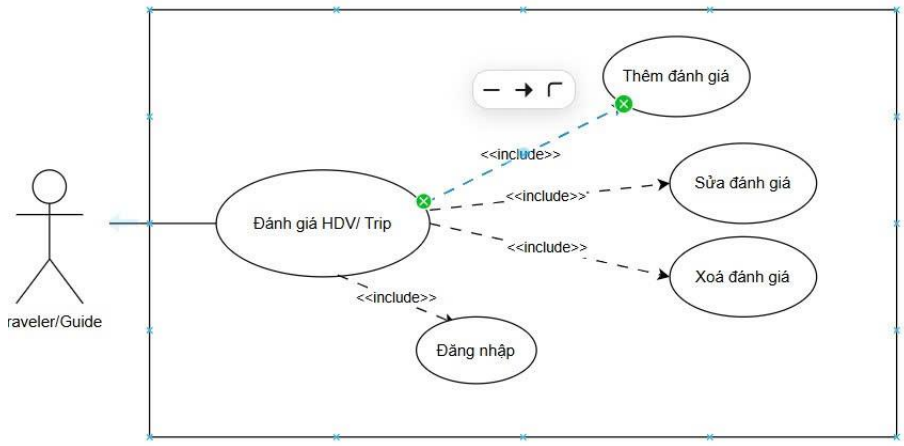



*Hình 2.15. Sơ đồ ca sử dụng chức năng nạp tiền*

| **Mã Usecase** | UC11 |
|---|---|
| **Tên Usecase** | Nạp tiền vào ví |
| **Tác nhân** | Traveler và Guide |
| **Mô tả** | Cho phép người dùng nạp tiền vào ví điện tử qua chuyển khoản ngân hàng; hệ thống tự động ghi nhận qua webhook SePay mà không cần admin xác nhận |
| **Sự kiện kích hoạt** | Người dùng nhấn "Nạp tiền" và nhập số tiền. |
| **Tiền điều kiện** | Người dùng đã đăng nhập; có ví (tạo tự động khi đăng ký) |

| | |
|---|---|
| **Hậu điều kiện** | Khi khớp giao dịch: số dư khả dụng tăng; giao dịch ghi SUCCESS; người dùng nhận thông báo |
| **Luồng sự kiện chính** | 1. Người dùng nhập số tiền nạp.<br>2. Hệ thống sinh mã NAP_XXXXXXXX, tạo giao dịch PENDING (POST /payments/sepay/intent).<br>3. Hiển thị mã QR + thông tin tài khoản MB Bank + nội dung chứa mã + đồng hồ đếm ngược.<br>4. Người dùng chuyển khoản qua app ngân hàng.<br>5. SePay đọc giao dịch, POST webhook về hệ thống.<br>6. Hệ thống so khớp mã + số tiền với giao dịch PENDING.<br>7. Khớp → cộng tiền vào ví, đánh dấu SUCCESS, đẩy thông báo. |
| **Luồng sự kiện thay thế** | Không có |
| **Luồng sự kiện ngoại lệ** | Số tiền lệch so với mã → đánh dấu FAILED, thông báo lỗi.<br>Webhook không có/đsai API key → từ chối 401.<br>Hết thời gian đếm ngược chưa chuyển → giao dịch giữ PENDING/huỷ |

- Sơ đồ ca sử dụng chức năng đánh giá

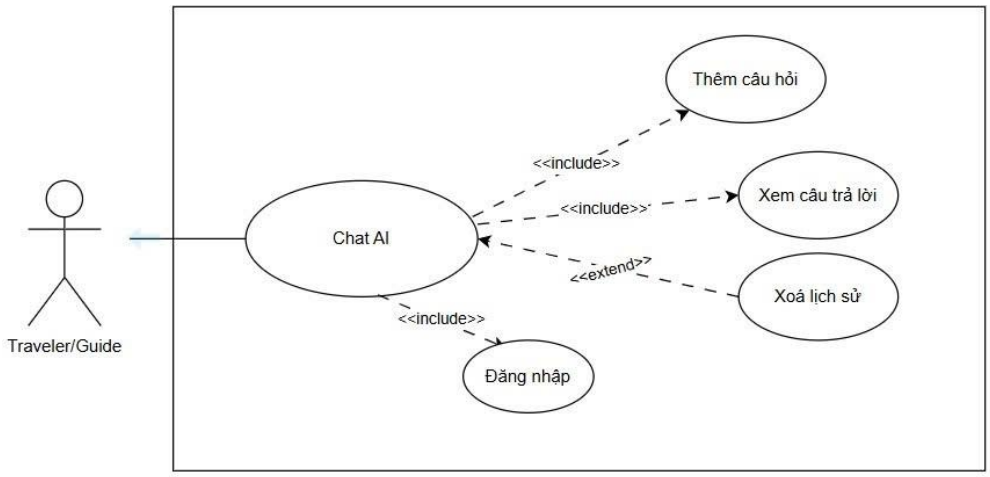



*Hình 2.16. Sơ đồ ca sử dụng chức năng đánh giá*

| **Mã Usecase** | UC12 |
|---|---|
| **Tên Usecase** | Dánh giá chuyến đi / HDV / địa điểm |
| **Tác nhân** | Traveler |
| **Mô tả** | Cho phép người dùng chấm điểm sao và viết nhận xét cho nhiều loại đối tượng (chuyến đi, hướng dẫn viên, địa điểm, thành viên cùng chuyến); có thể trả lời đánh giá. |
| **Sự kiện kích hoạt** | Người dùng nhấn "Đánh giá" sau khi hoàn thành chuyến/tour, hoặc trên trang đối tượng. |
| **Tiền điều kiện** | Người dùng đã đăng nhập |
| **Hậu điều kiện** | Đánh giá được lưu; điểm trung bình và số lượt đánh giá của đối tượng cập nhật. |
| **Luồng sự kiện chính** | 1. Người dùng mở form đánh giá đối tượng.<br>2. Chọn số sao (1–5), nhập nhận xét.<br>3. Nhấn "Gửi" (POST /reviews).<br>4. Hệ thống lưu đánh giá theo target_type + target_id. |

| | |
|---|---|
| | 5. Cập nhật rating trung bình của đối tượng. |
| **Luồng sự kiện thay thế** | 1. Trả lời một đánh giá khác: tạo đánh giá con qua parent_id (rating = 0).<br>2. Thích một đánh giá: POST /reviews/:id/like. |
| **Luồng sự kiện ngoại lệ** | Điểm sao ngoài 1–5 → báo lỗi validate. |

- Sơ đồ ca sử dụng chức năng bot chat



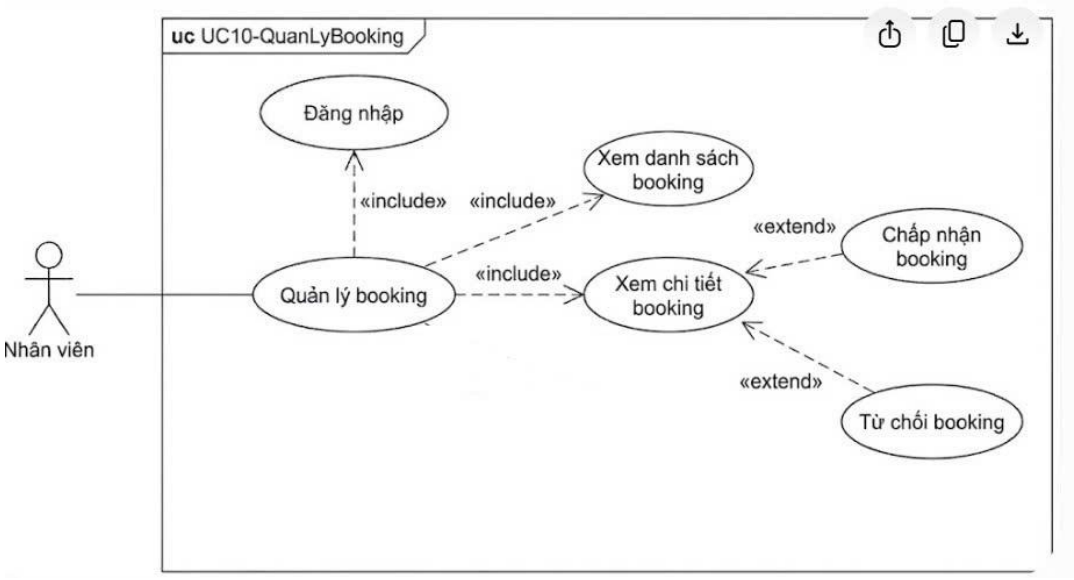

*Hình 2.17. Sơ đồ ca sử dụng chức năng bot chat*

| **Mã Usecase** | UC13 |
|---|---|
| **Tên Usecase** | Chat với trợ lý AI và sinh chuyến đi |
| **Tác nhân** | Traveler |
| **Mô tả** | Cho phép người dùng hỏi trợ lý AI về du lịch; AI hiểu yêu cầu, truy xuất dữ liệu thật (chuyến đi, kho 63 tỉnh) và gợi ý lộ trình; người dùng có thể chốt để tạo chuyến đi thật. |
| **Sự kiện kích hoạt** | Người dùng mở bong bóng chat AI và gửi câu hỏi. |
| **Tiền điều kiện** | Người dùng đã đăng nhập |

| | |
|---|---|
| **Hậu điều kiện** | AI trả về câu trả lời/lộ trình; nếu chốt, một chuyến đi thật được tạo và lưu. |
| **Luồng sự kiện chính** | 1. Người dùng gửi câu hỏi (vd "gợi ý đi Đà Nẵng 3 ngày").<br>2. Hệ thống chạy pipeline: HIỂU, LÀM VIỆC → TRẢ VỀ (LLM viết lộ trình tiếng Việt) (POST /ai/ask).<br>3. Hiển thị câu trả lời + lộ trình gợi ý.<br>4. Người dùng chỉnh sửa địa điểm / chốt ngày, số người, chi phí.<br>5. Người dùng nhấn "Tạo chuyến" (POST /ai/create-trip).<br>6. Hệ thống tạo chuyến đi thật từ lộ trình. |
| **Luồng sự kiện thay thế** | Người dùng yêu cầu thay một địa điểm → AI tra dữ liệu thật; nếu không có thì tham khảo Gemini và ghi rõ nguồn. |
| **Luồng sự kiện ngoại lệ** | 1. LLM lỗi/chưa sẵn sàng → hệ thống fallback trả lời dựa trên truy vấn DB thuần, chatbot không "chết".<br>2. Người dùng chưa đăng nhập khi tạo chuyến → yêu cầu đăng nhập. |

#### 2.4.2.3. Tổng quan chức năng hướng dẫn viên

- Sơ đồ ca sử dụng chức năng quản lý booking

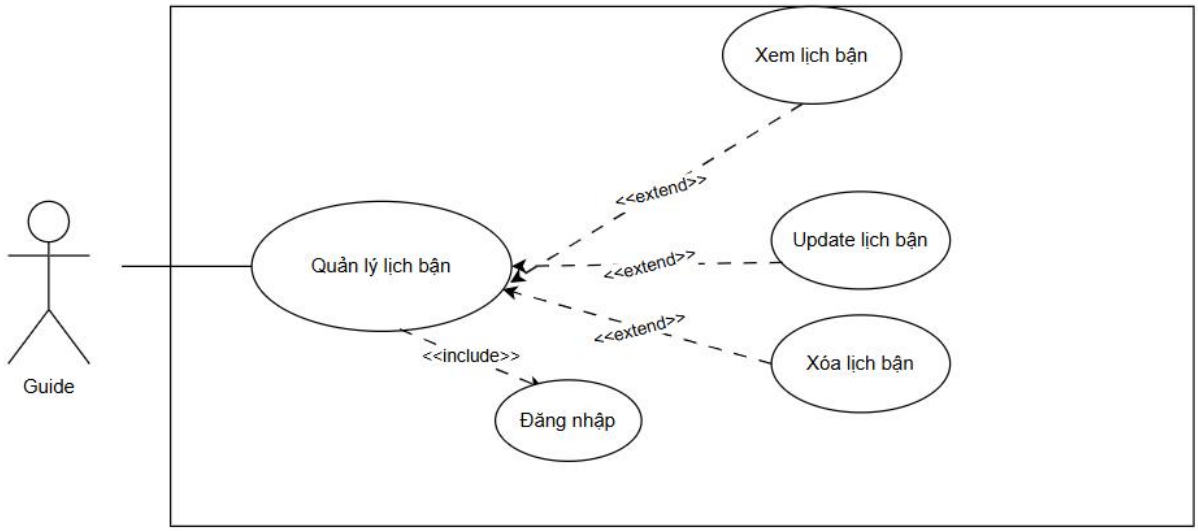



*Hình 2.18. Sơ đồ ca sử dụng chức năng quản lý booking*

| **Mã Usecase** | UC14 |
|---|---|
| **Tên Usecase** | Quản lý đặt lịch |
| **Tác nhân** | Guide |
| **Mô tả** | Cho phép HDV xem danh sách booking gửi đến, xem chi tiết, chấp nhận hoặc từ chối; sau khi khách thanh toán thì thực hiện tour và đánh dấu hoàn thành. |
| **Sự kiện kích hoạt** | HDV mở trang quản lý đặt lịch. |
| **Tiền điều kiện** | Người dùng có vai trò Guide (đã được duyệt). |
| **Hậu điều kiện** | Trạng thái booking được cập nhật theo vòng đời; các bên nhận thông báo tương ứng. |
| **Luồng sự kiện chính** | 1. HDV mở danh sách booking của mình<br>2. Chọn một booking xem chi tiết.<br>3. HDV nhấn "Chấp nhận"<br>4.Hệ thống chuyển trạng thái sang "chờ thanh toán", thông báo khách. |

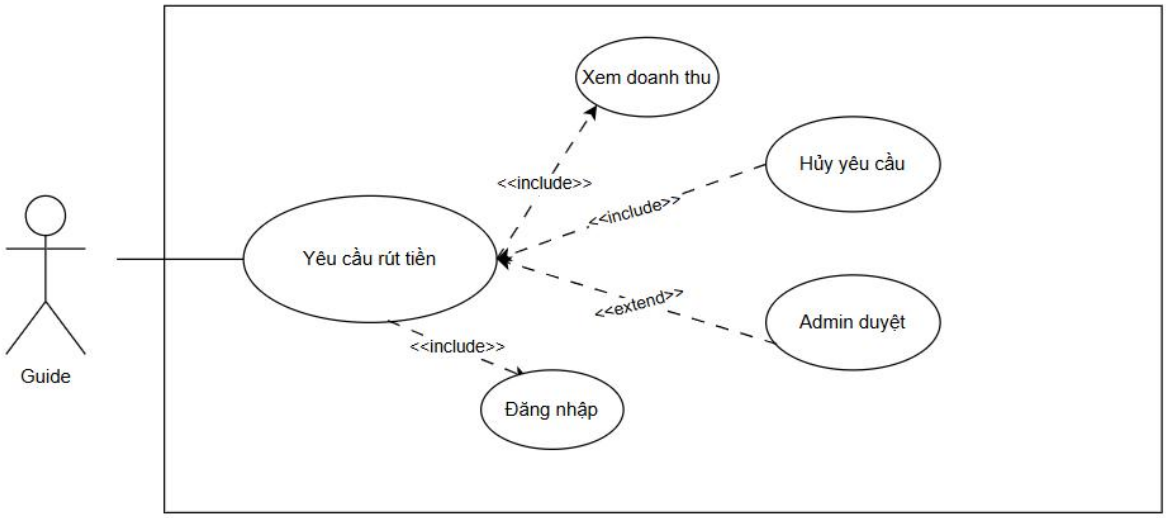
| | |
|---|---|
| | 5.Khách thanh toán → tiền đóng băng.<br>6. Sau tour, HDV/khách đánh dấu "Hoàn thành" → hệ thống giải phóng tiền cho HDV (trừ 10% hoa hồng). |
| **Luồng sự kiện thay thế** | HDV "Từ chối" booking → trạng thái rejected, thông báo khách (kèm lý do nếu có). |
| **Luồng sự kiện ngoại lệ** | 1. Khách không thanh toán trong thời hạn → booking hết hạn (expired).<br>2. Người thao tác không phải HDV của booking → từ chối 403. |

- Sơ đồ ca sử dụng chức năng quản lý lịch bận




*Hình 2.19. Sơ đồ ca sử dụng chức năng quản lý lịch bận*

- Sơ đồ ca sử dụng chức năng quản lý doanh thu và yêu cầu rút tiền



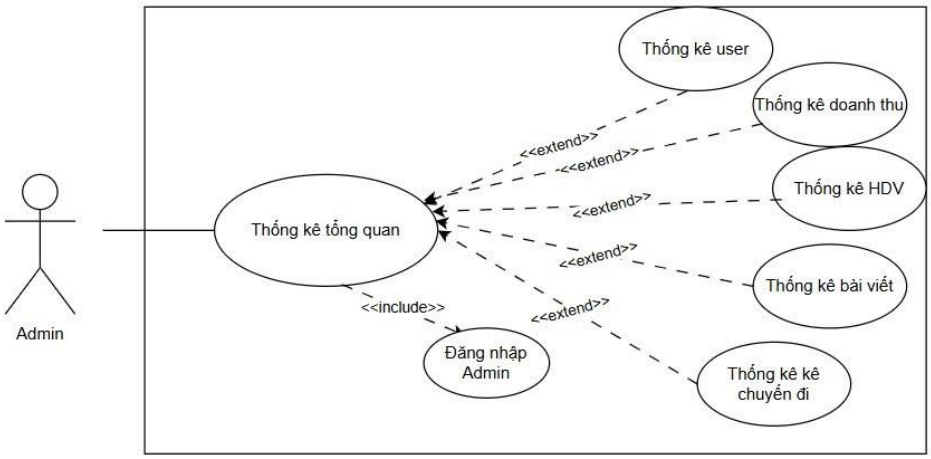

*Hình 2.20. Sơ đồ ca sử dụng quản lý doanh thu và rút tiền*

| **Mã Usecase** | UC15 |
|---|---|
| **Tên Usecase** | Quản lý doanh thu và rút tiền |
| **Tác nhân** | Guide |
| **Mô tả** | Cho phép HDV xem doanh thu (tiền nhận, hoa hồng đã trừ), thống kê, lịch sử giao dịch và gửi yêu cầu rút tiền về tài khoản ngân hàng. |
| **Sự kiện kích hoạt** | HDV mở trang ví/doanh thu hoặc nhấn "Rút tiền" |
| **Tiền điều kiện** | Người dùng là Guide; số dư khả dụng đủ để rút. |
| **Hậu điều kiện** | Yêu cầu rút được tạo trạng thái chờ duyệt; số dư khả dụng bị trừ tạm. |
| **Luồng sự kiện chính** | 1. HDV mở trang ví (GET /guides/wallet/me).<br>2. Xem số dư khả dụng/đóng băng, lịch sử giao dịch, thống kê. |

| | |
|---|---|
| | 3. Nhấn "Rút tiền", nhập số tiền + thông tin ngân hàng.<br>4. Hệ thống kiểm tra số dư, tạo yêu cầu rút (POST /guides/wallet/withdrawals).<br>5. Trừ tạm số dư khả dụng, đặt trạng thái chờ admin duyệt. |
| **Luồng sự kiện thay thế** | Không có |
| **Luồng sự kiện ngoại lệ** | 1. Số dư không đủ → báo lỗi, không tạo yêu cầu.<br>2. Thông tin ngân hàng thiếu/sai định dạng → báo lỗi validate. |

### 2.4.2.4. Tổng quan chức năng Admin

- Sơ đồ ca sử dụng chức năng thống kê tổng quan

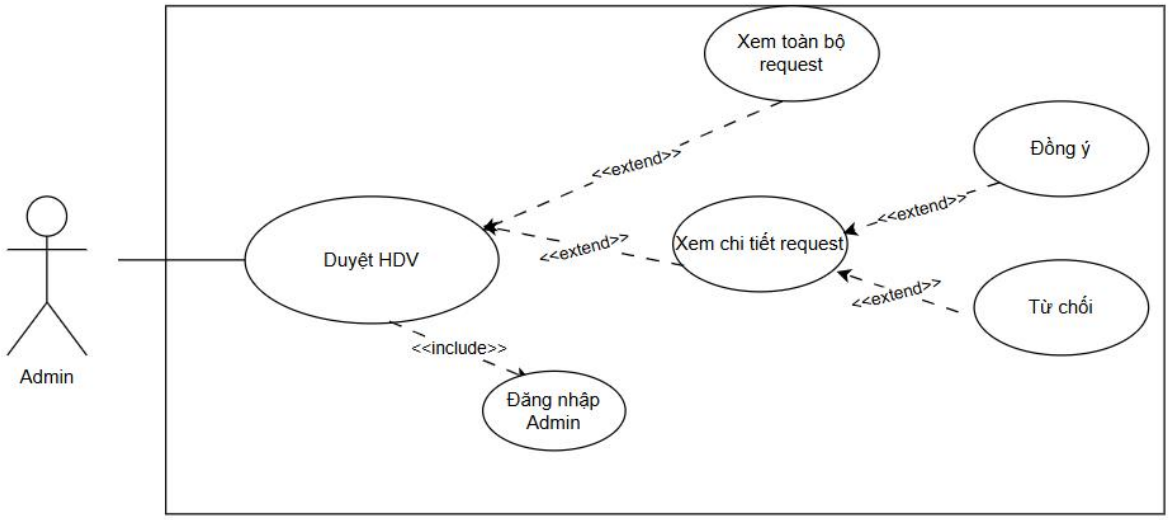



*Hình 2.21. Sơ đồ ca sử dụng chức năng thống kê tổng quan*

| **Mã Usecase** | UC16 |
|---|---|
| **Tên Usecase** | Thống kê tổng quan (Dashboard) |
| **Tác nhân** | Admin |
| **Mô tả** | Cung cấp bảng điều khiển tổng quan: số người dùng, số chuyến đi, doanh thu toàn hệ thống, số giao dịch, |

| | |
|---|---|
| | thống kê đăng ký và doanh thu theo hướng dẫn viên. |
| **Sự kiện kích hoạt** | Admin đăng nhập console và mở Dashboard. |
| **Tiền điều kiện** | Người dùng có vai trò Admin. |
| **Hậu điều kiện** | Hệ thống hiển thị các chỉ số thống kê tổng hợp. |
| **Luồng sự kiện chính** | 1. Admin mở Dashboard<br>2. Hệ thống tổng hợp số liệu: user, trip, doanh thu, giao dịch.<br>3. Hiển thị thẻ chỉ số và biểu đồ.<br>4. Admin xem thống kê doanh thu , đăng ký , doanh thu theo HDV |
| **Luồng sự kiện thay thế** | Admin lọc theo khoảng thời gian để xem xu hướng |
| **Luồng sự kiện ngoại lệ** | Người thao tác không phải Admin → chặn 403 |

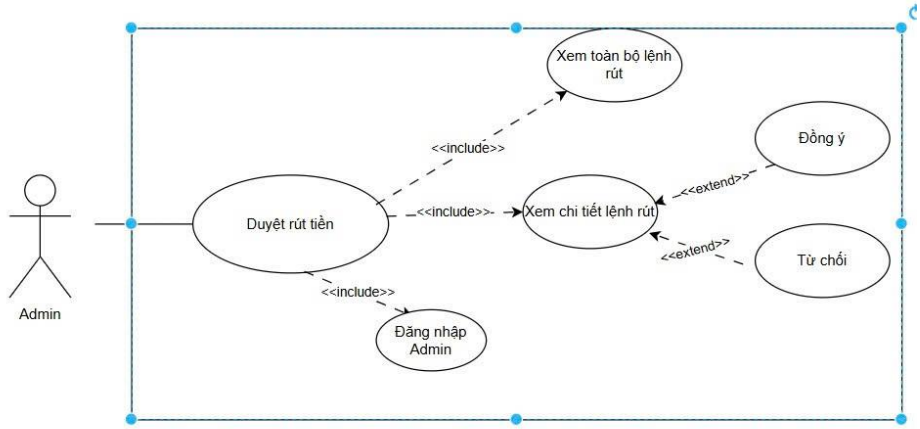
- Sơ đồ ca sử dụng chức năng duyệt yêu cầu làm hướng dẫn viên




*Hình 2.22. Sơ đồ ca sử dụng duyệt yêu cầu làm hướng dẫn viên*

| **Mã Usecase** | UC17 |
|---|---|
| **Tên Usecase** | Duyệt yêu cầu làm hướng dẫn viên |
| **Tác nhân** | Admin |
| **Mô tả** | Cho phép quản trị viên xem danh sách hồ sơ HDV chờ duyệt, xem chi tiết và phê duyệt hoặc từ chối (kèm lý do). |
| **Sự kiện kích hoạt** | Admin mở mục "Duyệt hướng dẫn viên" trong console. |
| **Tiền điều kiện** | Người dùng có vai trò Admin; tồn tại hồ sơ trạng thái pending. |
| **Hậu điều kiện** | Nếu duyệt: user được nâng quyền thành Guide, hồ sơ approved. Nếu từ chối: hồ sơ rejected kèm lý do. Người nộp nhận thông báo. |
| **Luồng sự kiện chính** | 1. Admin mở danh sách hồ sơ chờ duyệt<br>2. Xem chi tiết một hồ sơ.<br>3. Admin nhấn "Duyệt"<br>4. Hệ thống nâng role user thành guide, đặt hồ sơ approved.<br>5. Đẩy thông báo cho người nộp. |
| **Luồng sự kiện thay thế** | Từ chối: Admin nhấn "Từ chối", nhập lý do → hồ sơ rejected, thông báo kèm lý do. |
| **Luồng sự kiện ngoại lệ** | 1. Người thao tác không phải Admin → chặn 403<br>2. Hồ sơ đã được xử lý trước đó → báo trạng thái không hợp lệ. |

- Sơ đồ ca sử dụng chức năng duyệt rút tiền

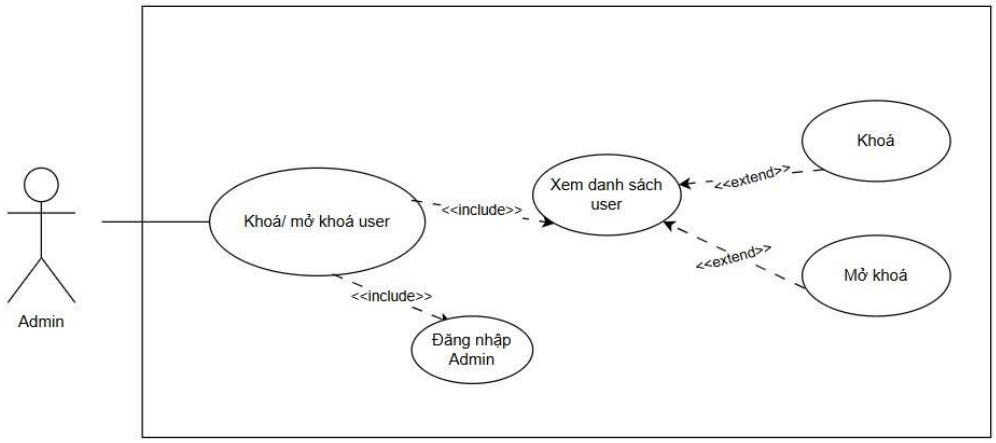



*Hình 2.23. Sơ đồ ca sử dụng duyệt rút tiền*

| **Mã Usecase** | UC18 |
|---|---|
| **Tên Usecase** | Duyệt yêu cầu rút tiền |
| **Tác nhân** | Admin |
| **Mô tả** | Cho phép quản trị viên xem các yêu cầu rút tiền đang chờ và phê duyệt (ghi nhận đã chi) hoặc từ chối (hoàn lại số dư cho HDV). |
| **Sự kiện kích hoạt** | Admin mở mục "Duyệt rút tiền". |
| **Tiền điều kiện** | Người dùng là Admin; tồn tại yêu cầu rút trạng thái chờ duyệt. |
| **Hậu điều kiện** | Nếu duyệt: ghi giao dịch rút thành công. Nếu từ chối: hoàn số dư về ví HDV. Người yêu cầu nhận thông báo. |
| **Luồng sự kiện chính** | 1. Admin mở danh sách yêu cầu rút<br>2. Xem chi tiết yêu cầu (số tiền, thông tin ngân hàng).<br>3. Admin nhấn "Duyệt" |

| | |
|---|---|
| | 4. Hệ thống ghi giao dịch WITHDRAW_SUCCESS, đẩy thông báo. |
| **Luồng sự kiện thay thế** | Từ chối: hệ thống hoàn số tiền về số dư khả dụng của HDV, ghi giao dịch hoàn, thông báo. |
| **Luồng sự kiện ngoại lệ** | 1. Người thao tác không phải Admin → chặn 403.<br>2. Yêu cầu đã xử lý → báo trạng thái không hợp lệ. |

- Sơ đồ ca sử dụng chức năng khoá tài khoản

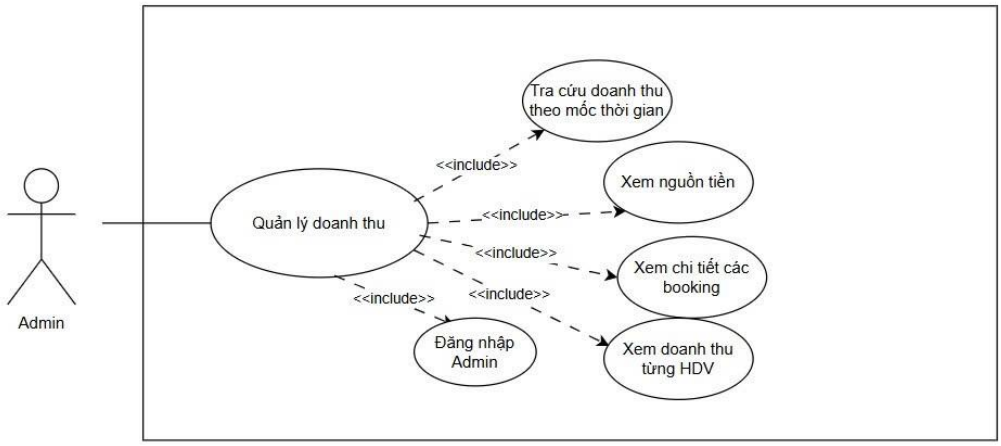



*Hình 2.24. Sơ đồ ca sử dụng chức năng khóa tài khoản*

| **Mã Usecase** | UC19 |
|---|---|
| **Tên Usecase** | Khóa tài khoản người dùng |
| **Tác nhân** | Admin |
| **Mô tả** | Cho phép quản trị viên xem danh sách người dùng, khoá hoặc mở khoá tài khoản vi phạm, và nạp tiền thủ công vào ví khi cần |
| **Sự kiện kích hoạt** | Admin mở mục "Quản lý user". |
| **Tiền điều kiện** | Người dùng là Admin |

| | |
|---|---|
| **Hậu điều kiện** | Trạng thái tài khoản (khoá/mở) |
| **Luồng sự kiện chính** | 1. Admin mở danh sách người dùng<br>2.Tìm/lọc user cần xử lý.<br>3. Admin nhấn "Khoá" (POST /admin/users/:id/lock).<br>4. Hệ thống đặt is_locked = true; user bị khoá không đăng nhập được. |
| **Luồng sự kiện thay thế** | 1. Mở khoá: POST /admin/users/:id/unlock → is_locked = false.<br>2. Nạp tiền thủ công: Admin nạp cho một hoặc nhiều user |
| **Luồng sự kiện ngoại lệ** | 1. Người thao tác không phải Admin → chặn 403.<br>2. User đang ở phiên đăng nhập → token bị vô hiệu ở lần |

- Sơ đồ ca sử dụng chức năng quản lý doanh thu




*Hình 2.25. Sơ đồ ca sử dụng chức năng quản lý doanh thu*

| **Mã Usecase** | UC20 |
|---|---|
| **Tên Usecase** | Quản lý chi tiết doanh thu |
| **Tác nhân** | Admin |
| **Mô tả** | Cho phép quản trị viên xem chi tiết doanh thu phát sinh từ từng hướng dẫn viên: tổng tiền các booking đã hoàn thành, phần hoa hồng hệ thống thu (10%), số tiền thực nhận của HDV, số tour hoàn thành và lịch sử giao dịch liên quan. Phục vụ đối soát và báo cáo tài chính |
| **Sự kiện kích hoạt** | Admin chọn một hướng dẫn viên trong mục thống kê/doanh thu và nhấn "Xem chi tiết doanh thu". |
| **Tiền điều kiện** | Người dùng có vai trò Admin; hướng dẫn viên tồn tại và đã được duyệt. |
| **Hậu điều kiện** | Hệ thống hiển thị bảng chi tiết doanh thu của HDV theo từng booking và tổng hợp; không làm thay đổi dữ liệu. |
| **Luồng sự kiện chính** | 1. Admin mở mục thống kê doanh thu, chọn một |

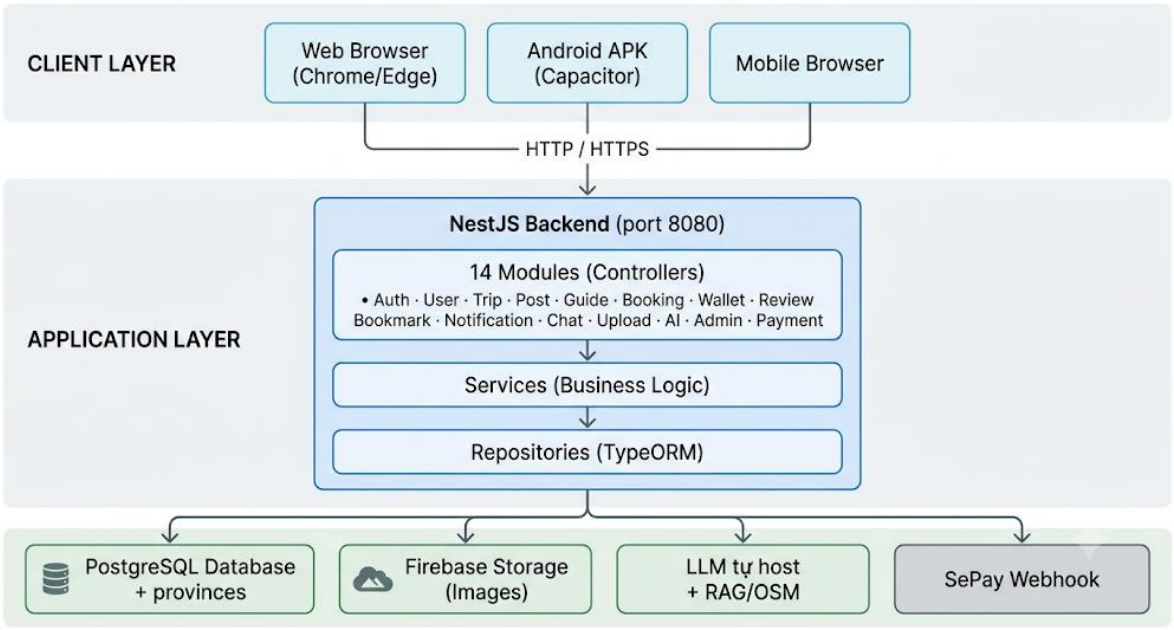
| | |
|---|---|
| | hướng dẫn viên.<br>2. Hệ thống truy vấn doanh thu của HDV đó<br>3. Hệ thống tổng hợp: tổng doanh thu, hoa hồng đã thu, thực nhận của HDV, số tour hoàn thành.<br>4. Hệ thống liệt kê chi tiết từng booking (khách, ngày, số tiền, trạng thái) và các giao dịch ví liên quan.<br>5. Hiển thị bảng chi tiết + tổng hợp cho Admin xem/đối soát. |
| **Luồng sự kiện thay thế** | 1. Admin lọc theo khoảng thời gian (tháng/quý) để xem doanh thu HDV theo kỳ.<br>2. Admin xuất/đối chiếu số liệu với mục thống kê doanh thu toàn hệ thống |
| **Luồng sự kiện ngoại lệ** | 1. Người thao tác không phải Admin → hệ thống chặn 403 (RolesGuard).<br>2b. ID hướng dẫn viên không tồn tại → báo lỗi 404. |

## 2.3. Thiết kế hệ thống

### *2.3.1. Kiến trúc tổng thể*

#### *2.3.1.1 Sơ đồ kiến trúc*

- Hệ thống TripMate được thiết kế theo kiến trúc 3-tier mở rộng




*Hình 2.26. Sơ đồ kiến trúc hệ thống*

#### *2.3.1.2. Trách nhiệm*

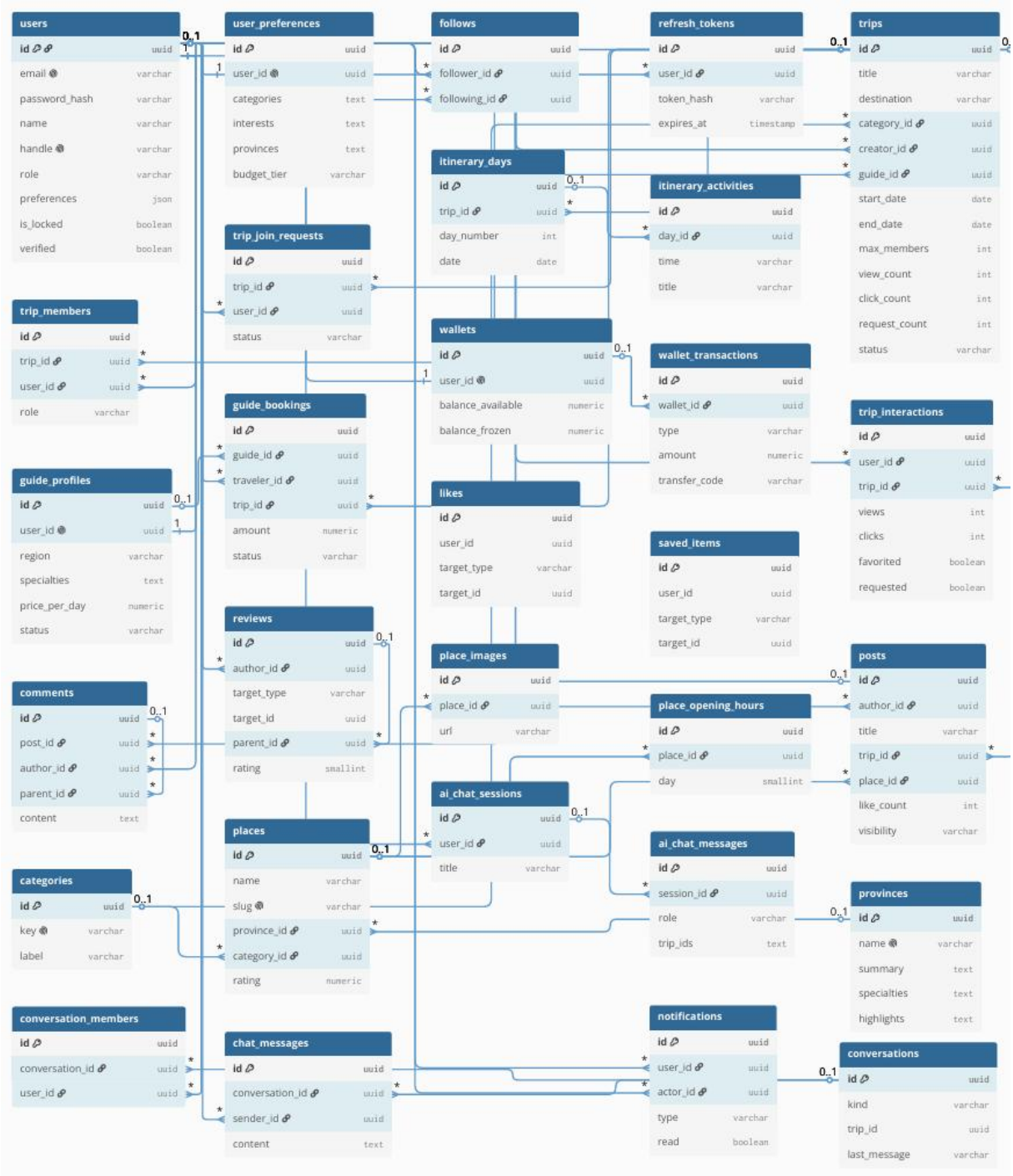
- Client layer : chịu trách nhiệm hiển thị, validation cơ bản, quản lý local state. Không chứa business logic nhạy cảm. Bản đồ vẽ trực tiếp ở client qua Leaflet + OpenStreetMap.
- Application layer (BE): xử lý toàn bộ business logic, validate dữ liệu, quản lý transaction, gọi 3rd party. Trợ lý AI chạy mô hình ngôn ngữ tự host ngay trong tiến trình backend.
- Data layer: PostgreSQL cho dữ liệu cấu trúc (kèm kho kiến thức 63 tỉnh cho RAG), Firebase cho ảnh; dịch vụ ngoài duy nhất bắt buộc là SePay (webhook). Gemini chỉ là tuỳ chọn dự phòng.

### *2.3.2. Thiết kế cơ sở dữ liệu*

#### *2.3.2.1. Sơ đồ quan hệ*

Biểu đồ ERD của hệ thống bao gồm 29 thực thể (bảng dữ liệu) với hơn 150 thuộc tính, được phân thành các nhóm chức năng chính gồm quản lý người dùng, quản lý chuyến đi, hướng dẫn viên và thanh toán, mạng xã hội du lịch, quản lý địa điểm, nhắn tin và AI Chat. Các thực thể được liên kết thông qua nhiều mối quan hệ một-một, một-nhiều và nhiều-nhiều nhằm đáp ứng đầy đủ các nghiệp vụ của hệ thống. Trong đó, bảng USERS đóng vai trò trung tâm và có quan hệ với phần lớn các bảng còn lại, phản

ánh đặc điểm của một hệ thống lấy người dùng làm trọng tâm. Thiết kế này cho phép quản lý dữ liệu chặt chẽ, đảm bảo tính toàn vẹn tham chiếu và hỗ trợ mở rộng khi bổ sung các chức năng mới trong tương lai.




*Hình 2.27. Sơ đồ quan hệ*

#### *2.3.2.2. Mô tả chi tiết một số bảng quan trọng*

*Bảng 2.8. Cấu trúc bảng users*

| Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
|---|---|---|---|
| id | uuid | PK | Mã định danh người dùng |
| email | varchar(320) | UNIQUE, NOT NULL | Email đăng nhập |
| password_hash | varchar(255) | NOT NULL | Mật khẩu băm bcrypt, không trả qua API |
| name | varchar(100) | NOT NULL | Tên hiển thị |
| handle | varchar(50) | UNIQUE, NULL | @username cho URL hồ sơ |
| role | enum | DEFAULT 'traveler' | Vai trò trong hệ thống(traveler / guide / admin) |
| avatar | varchar(500) | NULL | URL ảnh đại diện |
| cover | varchar(500) | NULL | URL ảnh bìa |
| phone | varchar(11) | NULL | Số điện thoại |
| social_links | jsonb | NULL | Liên kết MXH (instagram, facebook…) |
| bio | text | NULL | Tiểu sử |
| preferences | jsonb | DEFAULT '{}' | Sở thích du lịch |
| created_at | timestamptz | DEFAULT NOW() | Ngày tạo |
| updated_at | timestamptz | DEFAULT NOW() | Ngày cập nhật |

*Bảng 2.9. Cấu trúc bảng trips*

| Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
|---|---|---|---|
| id | uuid | Primary Key | Mã chuyến đi |
| title | varchar(200) | NOT NULL | Tên chuyến đi |

| | | | |
|---|---|---|---|
| description | text | NOT NULL | Mô tả chuyến đi |
| origin_name | varchar(200) | NULL | Điểm xuất phát |
| destination | varchar(200) | NOT NULL | Điểm đến |
| origin_lat / origin_lng | numeric(9,6) | NULL | Tọa độ xuất phát |
| destination_lat / destination_lng | numeric(9,6) | NULL | Tọa độ điểm đến |
| category_id | uuid | Foreign Key -> categories | Loại hình chuyến |
| cover_image | varchar(500) | NOT NULL | Ảnh bìa chuyến đi |
| gallery_urls | text[] | default {} | Thư viện ảnh |
| start_date | date | NOT NULL | Ngày bắt đầu |
| end_date | date | NOT NULL | Ngày kết thúc |
| duration_days | int | NOT NULL | Số ngày chuyến đi |
| price_from | numeric(12,2) | default 0 | Chi phí chuyến đi |
| max_members | int | default 8 | Số thành viên tối đa |
| member_count | int | default 1 | Số thành viên hiện tại |
| creator_id | uuid | Foreign Key -> users | Người tạo |
| guide_id | uuid | Foreign Key -> users | HDV được thuê |
| tags | text[] | default {} | Từ khóa |
| status | enum | default 'published' | Trạng thái |
| created_at | timestamptz | DEFAULT NOW() | Ngày tạo |
| updated_at | timestamptz | DEFAULT NOW() | Ngày cập nhật |

*Bảng 2.10. Cấu trúc bảng guide_profiles*

| Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
|---|---|---|---|
| id | uuid | Primary Key | Mã hồ sơ |
| user_id | uuid | Foreign Key -> users | Tài khoản Traveler |
| region | varchar(200) | NOT NULL | Khu vực hoạt động |
| region_keys | text[] | default {} | Khóa vùng để lọc |
| category_keys | text[] | default {} | Loại hình tour |
| languages | text[] | default {} | Ngôn ngữ giao tiếp |
| specialties | text[] | default {} | Chuyên môn |
| bio | text | NULL | Giới thiệu |
| years_experience | int | default 0 | Số năm kinh nghiệm |
| price_per_day | numeric(12,2) | default 0 | Giá theo ngày |
| currency | varchar(8) | default 'VND' | Đơn vị tiền |
| rating | numeric(3,2) | default 0 | Điểm đánh giá |
| review_count | int | default 0 | Số lượt đánh giá |
| id_card_number | varchar(30) | NOT NULL | CCCD (ẩn) |
| status | enum | default 'pending' | Trạng thái hồ sơ |
| reject_reason | varchar(500) | NULL | Lý do từ chối |
| created_at | timestamptz | DEFAULT NOW() | Ngày tạo |
| updated_at | timestamptz | DEFAULT NOW() | Ngày cập nhật |

*Bảng 2.11. cấu trúc bảng guide_bookings*

| Tên trường | Kiểu dữ liệu | Ràng buộc | Mô tả |
|---|---|---|---|
| id | uuid | Primary Key | Mã đặt lịch |
| guide_id | uuid | Foreign Key -> users | Hướng dẫn viên được đặt |
| traveler_id | uuid | Foreign Key -> | Người đặt lịch |

| | | | |
|---|---|---|---|
| | | users | |
| trip_id | uuid | Foreign Key -> trips | Chuyến đi gắn kèm |
| tour_title | varchar(200) | NOT NULL | Tên tour |
| tour_cover | varchar(200) | NOT NULL | Ảnh tour |
| destination | varchar(200) | NOT NULL | Điểm đến |
| start_date | date | NOT NULL | Ngày bắt đầu |
| end_date | date | NOT NULL | Ngày kết thúc |
| duration_days | int | default 1 | Số ngày |
| group_size | int | default 1 | Số khách |
| amount | numeric(12,2) | default 0 | Số tiền |
| status | enum | default 'pending_acceptance' | Trạng thái booking |
| cancel_reason | varchar(500) | NOT NULL | Lý do huỷ |
| accepted_at | timestamp | NULL | Thời gian xác nhận |
| paid_at | timestamp | NULL | Thời gian thanh toán |
| completed_at | timestamp | NULL | Thời gian hoàn thành |
| created_at | timestamp | DEFAULT NOW() | Thời gian tạo |
| updated_at | timestamp | DEFAULT NOW() | Thời gian cập nhật |

## 2.4. Thiết kế API và phân quyền

### *2.4.1. Convention API*

- Mọi endpoint có prefix /api
- Response format chuẩn: { data, message?, meta? }
- Lỗi: { statusCode, message, error, timestamp, path }
- Phân trang: query ?page=1&pageSize=20, response có meta.total, meta.totalPages

### *2.4.2. Phân quyền*

Áp dụng decorator-based:

- @Public() — bỏ qua JwtAuthGuard
- @Roles('admin') — chỉ admin
- Mặc định mọi route đều yêu cầu auth qua JwtAuthGuard global

### *2.4.3. Một số endpoint tiêu biểu*

*Bảng 2.12. Một số endpoint thể tiêu biểu*

| Endpoint | Method | Auth | Mô tả |
|---|---|---|---|
| /api/auth/register | POST | Public | Đăng ký |
| /api/auth/login | POST | Public | Đăng nhập |
| /api/auth/refresh | POST | Public | Lấy access mới |
| /api/users/me | GET | Auth | Lấy profile của tôi |
| /api/trips | GET | Public | Lấy danh sách trip |
| /api/trips | POST | Auth | Tạo trip |
| /api/trips/:id/join-requests | POST | Auth | Gửi yêu cầu tham gia |
| /api/bookings | POST | Auth | Đặt lịch |
| /api/payments/sepay/intent | POST | Auth | Tạo QR nạp tiền |
| /api/payments/sepay/webhook | POST | API key | Nhận webhook Sepay |
| /api/admin/stats | GET | Admin | Thống kê |

## 2.5. Thiết kế giao diện (UI/UX)

### *2.5.1. Hệ thống thiết kế*

TripMate sử dụng design system editorial (cảm hứng từ tạp chí du lịch):

- Màu chủ đạo: tone đất ấm (terracotta #B85940), nền be (#FAF6F0).
- Typography: cặp font Bricolage Grotesque (headline) + Inter (body).

- Border radius: bo tròn lớn (rounded-3xl) cho cảm giác mềm mại.
- Shadow: editorial shadow với độ mờ thấp, hiệu ứng layer.
- Glass blur cho navigation bar (top + bottom).

### *2.5.2. Cấu trúc trang chính*

Layout được chia 2 dạng:

- AuthLayout: dùng cho trang Login/Register, full-screen với hero ảnh.
- MainLayout: áp dụng cho 22 trang còn lại, có TopNav cố định + main content + Footer + BottomNav (mobile only) + AIChatBubble.

# Chương 3: TRIỂN KHAI VÀ ĐÁNH GIÁ

## 3.1.Môi trường và công cụ triển khai

### *3.1.1.Môi trường phát triển*

*Bảng 3.1. Môi trường phát triển*

| Thành phần | Cấu hình, phiên bản |
|---|---|
| Hệ điều hành | Windows 11 |
| Runtime | Node.js v20 LTS |
| Ngôn ngữ | TypeScript 5.x |
| Cơ sở dữ liệu | PostgreSQL 16 |
| Trình duyệt test | Chrome (DevTools mobile emulator), Firefox, Edge |
| Đóng gói mobile | JDK 21 (Temurin) + Android SDK command-line tools |
| Test API | Swagger UI (/docs), Postman, PowerShell Invoke-RestMethod |
| Quản lý phiên bản | Git |
| Thiết bị di động | Android thật |
| IDE | Visual Studio Code |

### *3.1.2.Công nghệ và thư viện sử dụng*

#### *3.1.2.1. Backend(NestJS)*

*Bảng 3.2. Thư viên sự dụng trong backend*

| Thư viện | Phiên bản | Vai trò |
|---|---|---|
| @nestjs/core, common | 10.4 | Framework nền, kiến trúc module hoá + DI |
| @nestjs/platform-express | 10.4 | HTTP server trên nền Express |
| typeorm + | 0.3 / 10.0 | ORM ánh xạ entity ↔ bảng PostgreSQL pg 8.13 |

| | | |
|---|---|---|
| @nestjs/typeorm | | Driver kết nối PostgreSQL |
| @nestjs/jwt, passport, passport-jwt, passport-local | 10.2 / 0.7 / 4.0 | Xác thực JWT (access + refresh) qua chiến lược Passport |
| bcrypt | 5.1 | Băm mật khẩu một chiều (salt rounds 10) |
| class-validator, class-transformer | 0.14 / 0.5 | Validate & chuyển đổi DTO đầu vào |
| @nestjs/websockets, @nestjs/platform-socket.io, socket.io | 10.4 / 4.8 | Chat realtime + push notification |
| @nestjs/schedule | 6.1 | Cron job (tự hoàn thành chuyến quá hạn…) |
| @nestjs/config + dotenv | 3.2 / 16.4 | Quản lý biến môi trường |
| node-llama-cpp | 3.18 | Chạy mô hình LLM tự host (Qwen2.5-3B) cho trợ lý AI |
| firebase-admin | 14.0 | Lưu trữ ảnh trên Firebase Storage |
| rxjs, reflect-metadata | 7.8 / 0.2 | Lập trình reactive + metadata cho decorator |

#### *3.1.2.2. Frontend(ReactJs)*

*Bảng 3.3. Thư viện sử dụng trong Frontend*

| Thư viện | Phiên bản | Vai trò |
|---|---|---|
| react, react-dom | 18.3 | Thư viện UI component-based |
| vite | 5.3 | Build tool + dev server (HMR nhanh) |
| typescript | 5.5 | Ngôn ngữ type-safe |
| react-router-dom | 6.24 | Định tuyến SPA, lazy load trang |
| @tanstack/react-query | 5.51 | Quản lý server state (cache, refetch, mutation) |
| zustand | 4.5 | Quản lý client state nhẹ (auth, user, AI…) |
| axios | 1.7 | HTTP client + interceptor |

| | | |
|---|---|---|
| | | tự gắn token / refresh |
| socket.io-client | 4.8 | Kết nối realtime tới gateway |
| leaflet, react-leaflet | 1.9 / 4.2 | Bản đồ + định tuyến OpenStreetMap/OSRM |
| tailwindcss + tailwind-merge + clsx | 3.4 / 2.4 / 2.1 | Styling tiện ích, gộp class điều kiện |

#### *3.1.2.3.Dịch vụ bên ngoài*

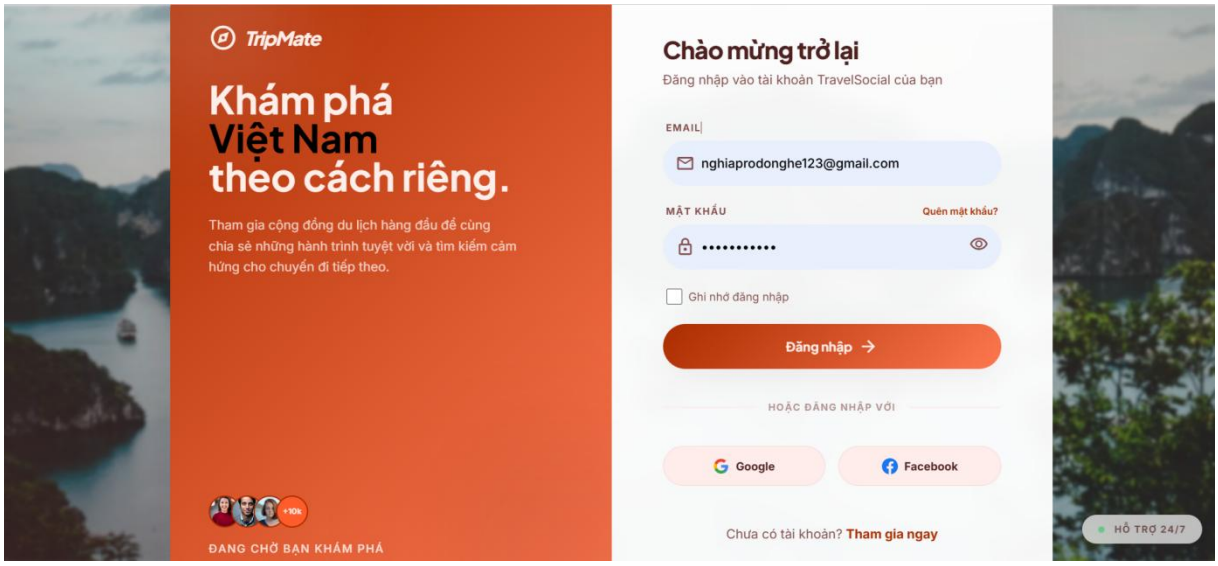
*Bảng 3.4. Thư viện và dịch vụ bên ngoài*

| Dịch vụ | Vai trò |
|---|---|
| PostgreSQL 16 | Cơ sở dữ liệu quan hệ chính |
| SePay | Cổng thanh toán — ghi nhận chuyển khoản tự động qua webhook |
| Firebase Storage | Lưu trữ ảnh (có fallback lưu local) |
| OpenStreetMap + Nominatim + OSRM | Tile bản đồ, geocoding, định tuyến (miễn phí, không cần API key) |
| Google Gemini | LLM dự phòng |
| Qwen2.5-3B-Instruct (GGUF) | Mô hình ngôn ngữ tự host chính cho trợ lý AI |
| Railway | Deploy backend |
| Vercel | Deploy frontend |
| @capacitor/core, @capacitor/cli, @capacitor/android | Đóng gói web app thành APK Android qua WebView |

## 3.2 Mô tả kết quả thực tế đạt được

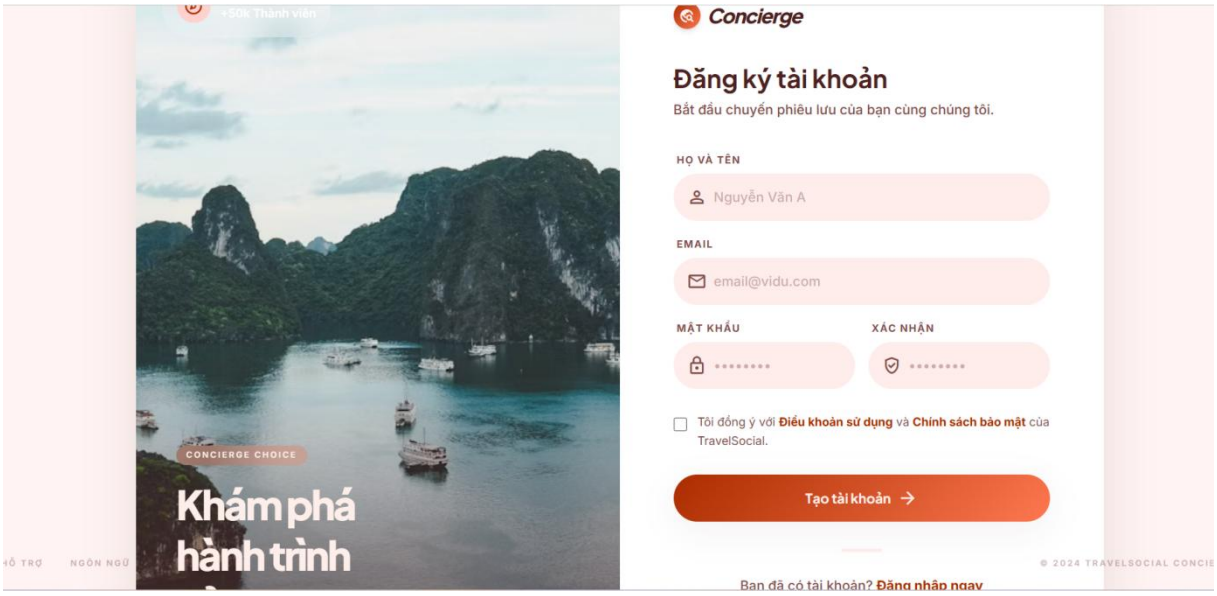
### *3.2.1. Giao diện chức năng chung*

- Chức năng đăng nhập : người dùng đăng nhập bằng email và mật khẩu



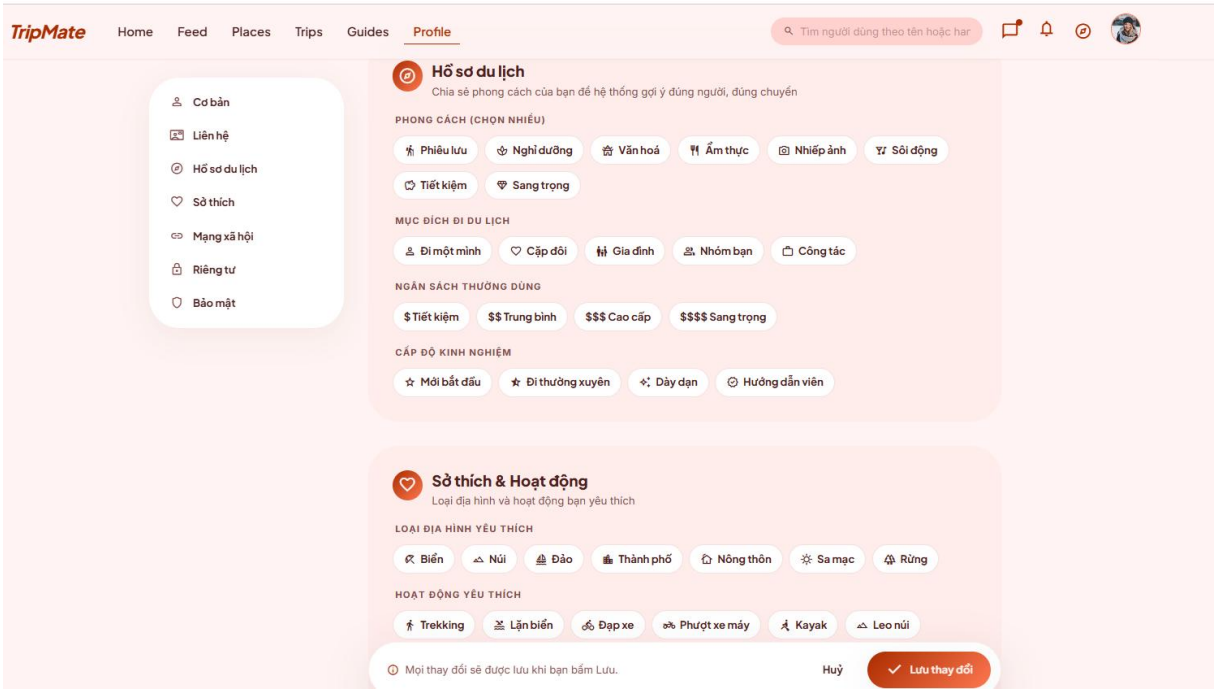

*Hình 3.1. Màn hình chức năng đăng nhập*

- Chức năng đăng ký : Điền họ và tên, email, mật khẩu. Sau đó xác thực OTP qua email



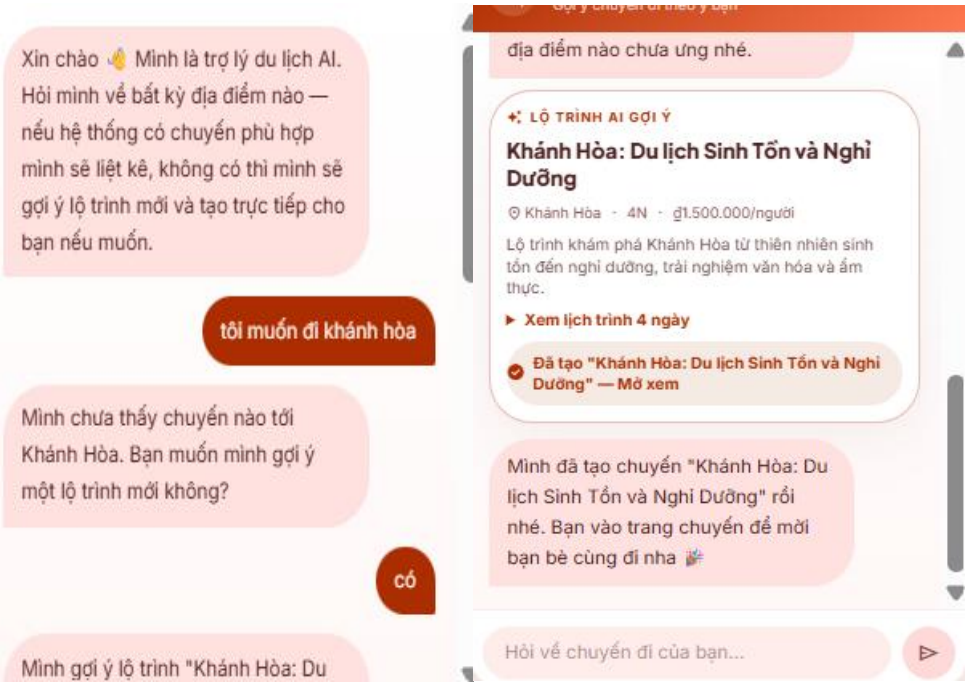

*Hình 3.2. Màn hình chức năng đăng ký*

- Chức năng chỉnh sửa hồ sơ người dùng: người dùng có thể thay đổi thông tin, kết nối trang mạng xã hội khác và thêm sở thích để được hệ thống gợi ý chuyến đi dễ dàng




*Hình 3.3. Màn hình chức năng chỉnh sửa hồ sơ người dùng*

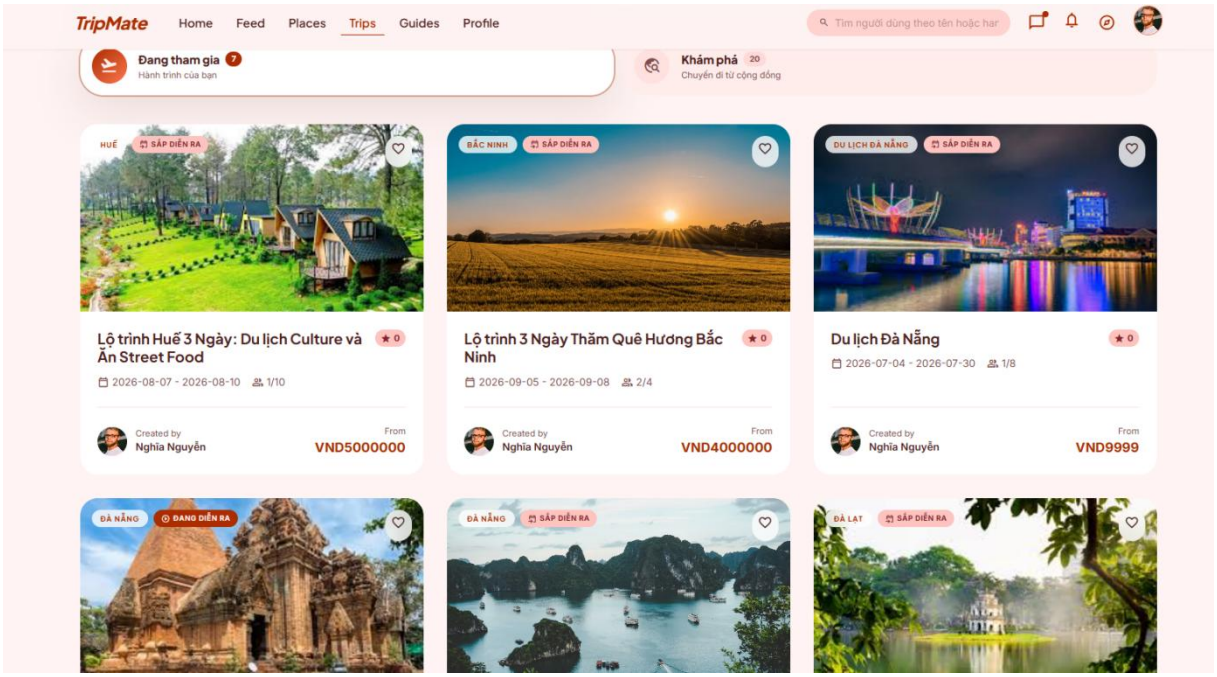
- Tính năng chatbot : hỏi thông tin bạn cần cho một chuyến đi hệ thống sẽ gợi ý cho bạn và hỗ trợ tạo




*Hình 3.4. Màn hình chức năng chat với AI*

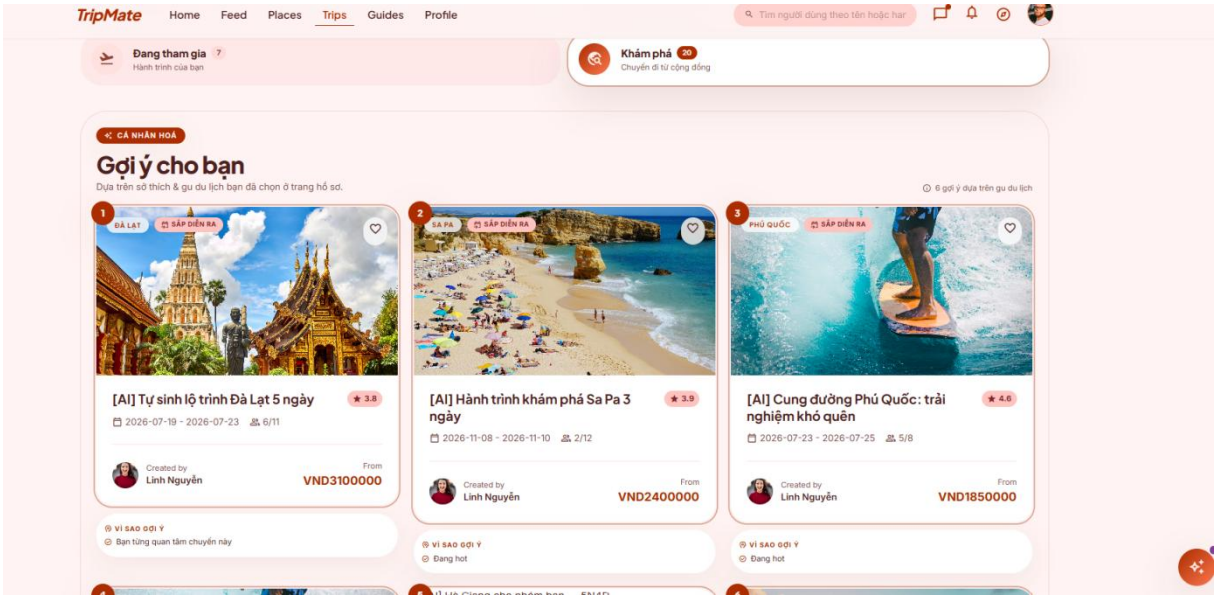
### *3.2.2 Giao diện chức quản lý chuyến đi*

- Chức năng xem danh sách chuyến đi : Trip phân loại đang tham gia và chưa tham gia



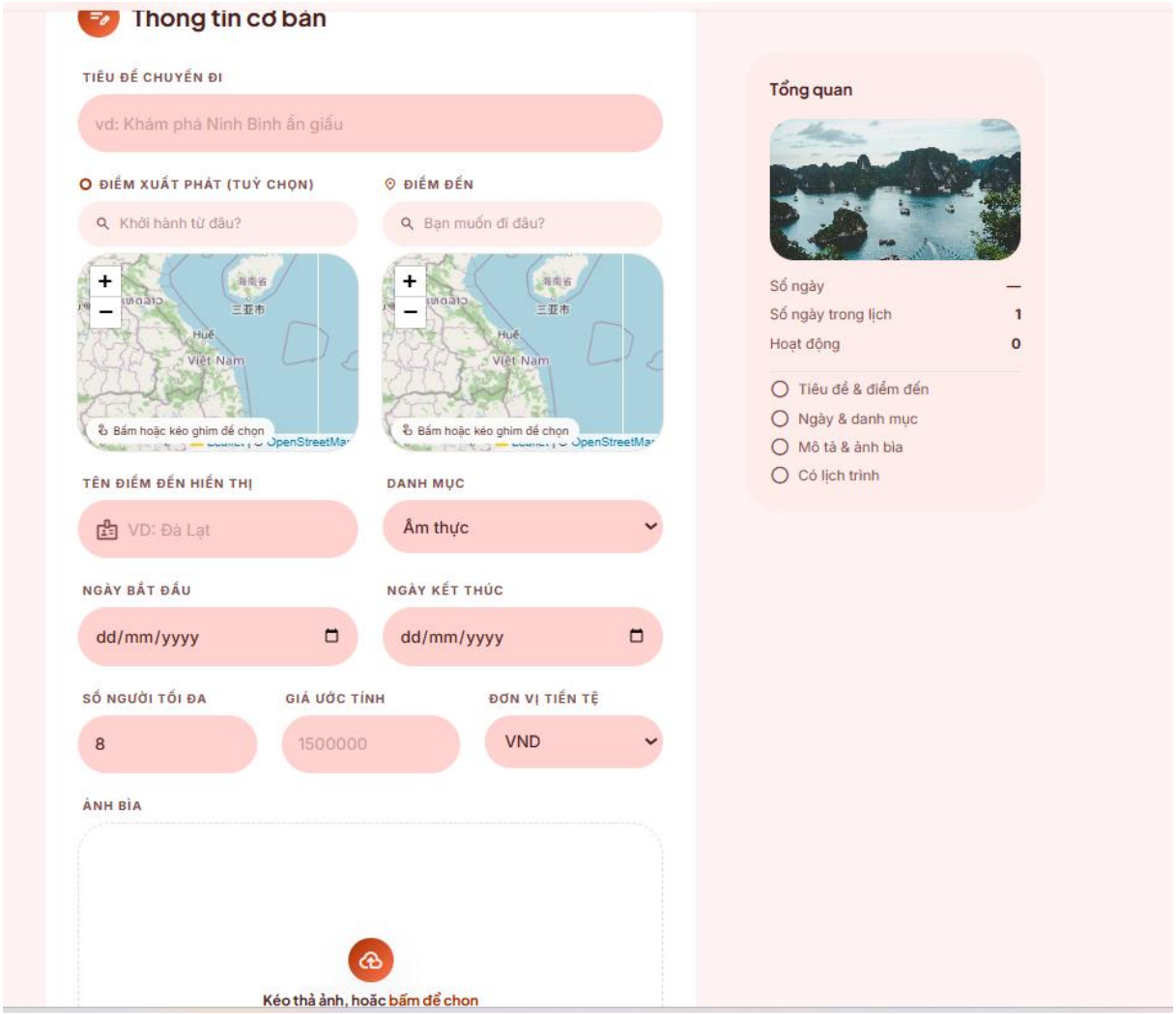

*Hình 3.5. Màn hình chức năng xem danh sách chuyến đi*

- Chức năng gợi ý chuyến đi có thể bạn thích : gợi ý dựa trên 3 yếu tố : hợp sở thích, tương tác nhiều với chuyến đi ( xem, lưu, thích ), chuyến đi được nhiều người tương tác ( hot)



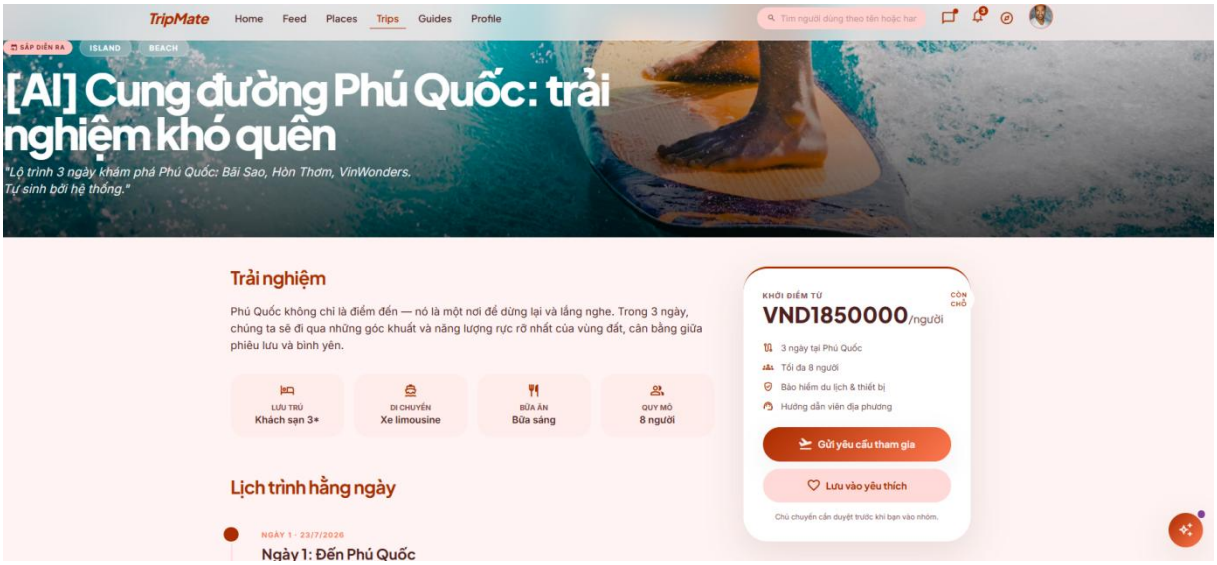

*Hình 3.6. Màn hình chức năng xem gợi ý về chuyến đi*

- Chức năng tạo chuyến đi : Nhập tiêu đề, tìm điểm xuất phát, điểm đến và nhập các thông tin cần thiết khác xong nhấn tạo



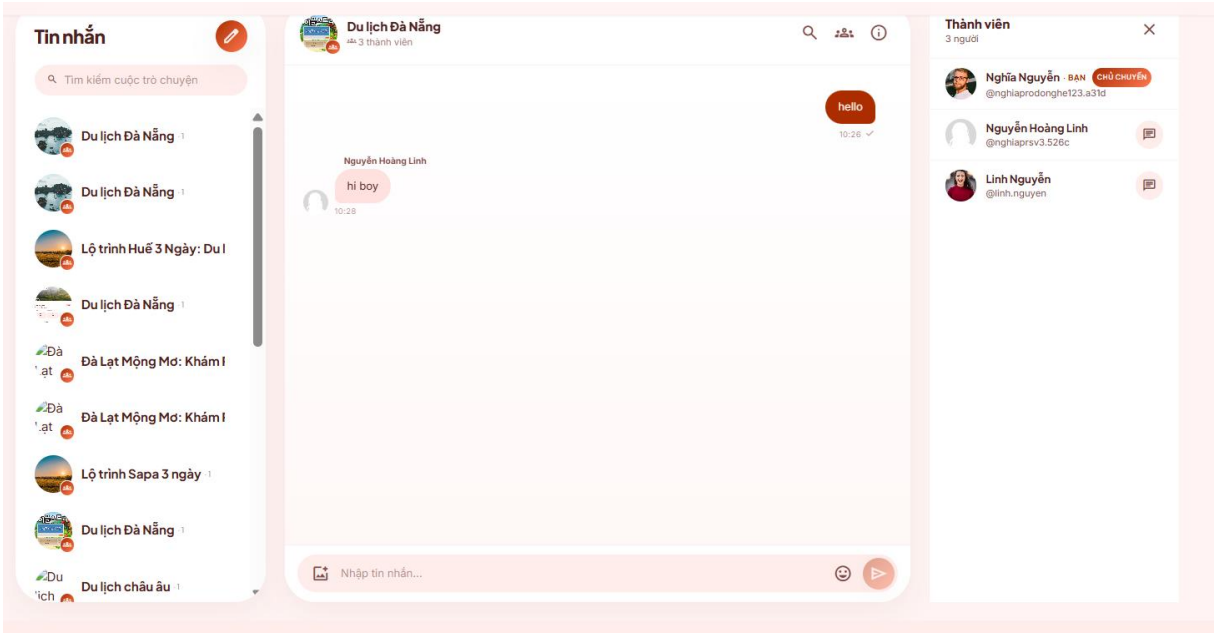

*Hình 3.7. Màn hình chức năng tạo chuyến đi*

- Chức năng xem chi tiết một chuyến đi : xem thông tin,bản đồ, lịch trình,yêu cầu tham gia và sửa chuyến đi (quyền chủ phòng)



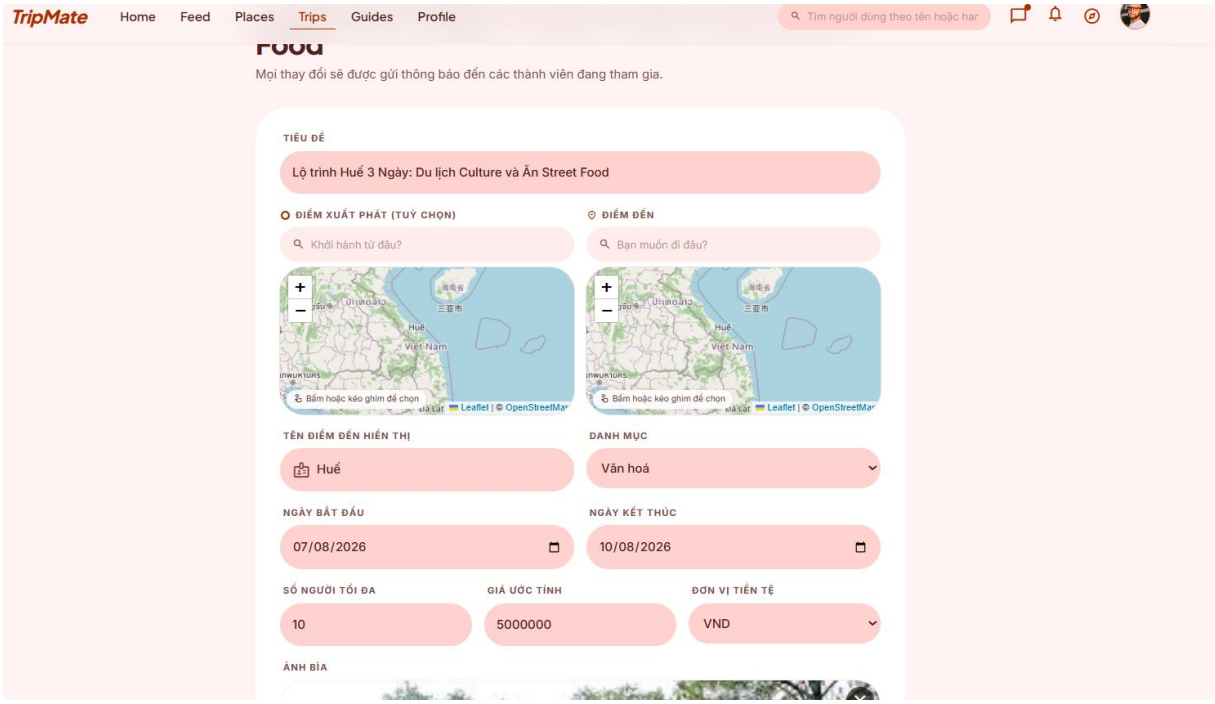

*Hình 3.8. Màn hình chức năng xem chi tiết chuyến đi*

- Chức năng chat room : nhóm chat sẽ tự tạo khi chuyến đi được tạo, member được thêm vào chuyến đi cũng sẽ được thêm vào nhóm chat để trao đổi



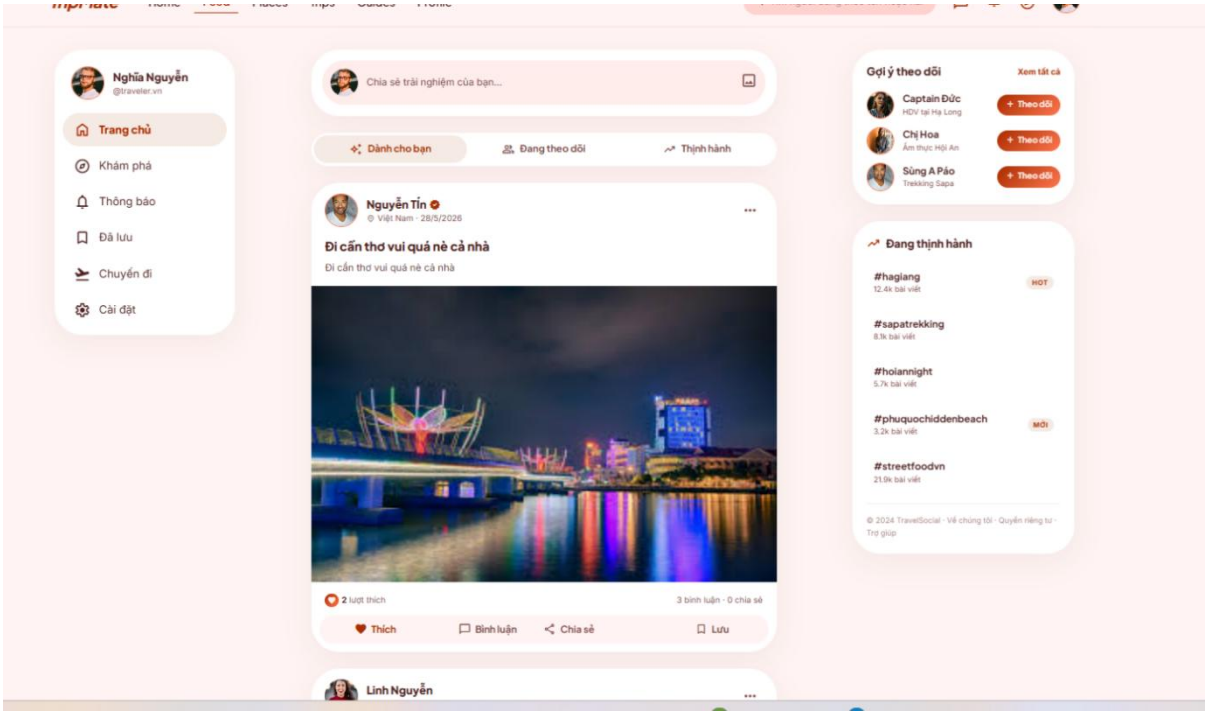

*Hình 3.9. Màn hình chức năng nhắn tin trong nhóm*

- Chức năng chỉnh sửa chuyến đi: chọn những chỗ cần sửa còn lại thì mặc định sẽ không đổi




*Hình 310. Màn hình chức năng chỉnh sửa chuyến đi*

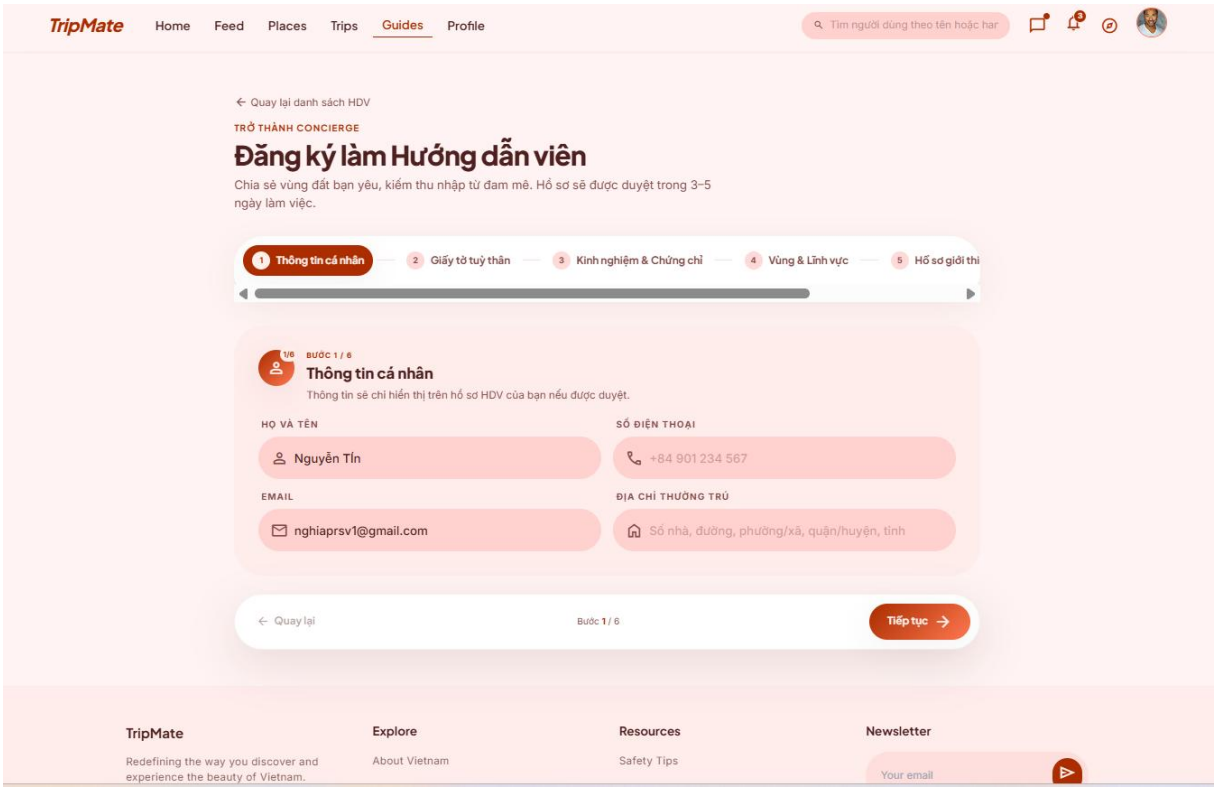
***3.2.3 .Chức năng quản lý bài viết*** : bao gồm các chức năng đăng bài( có thể upload ảnh, like,share, comment,reply comment, follow




*Hình 3.11. Màn hình chức năng quản lý bài viết*

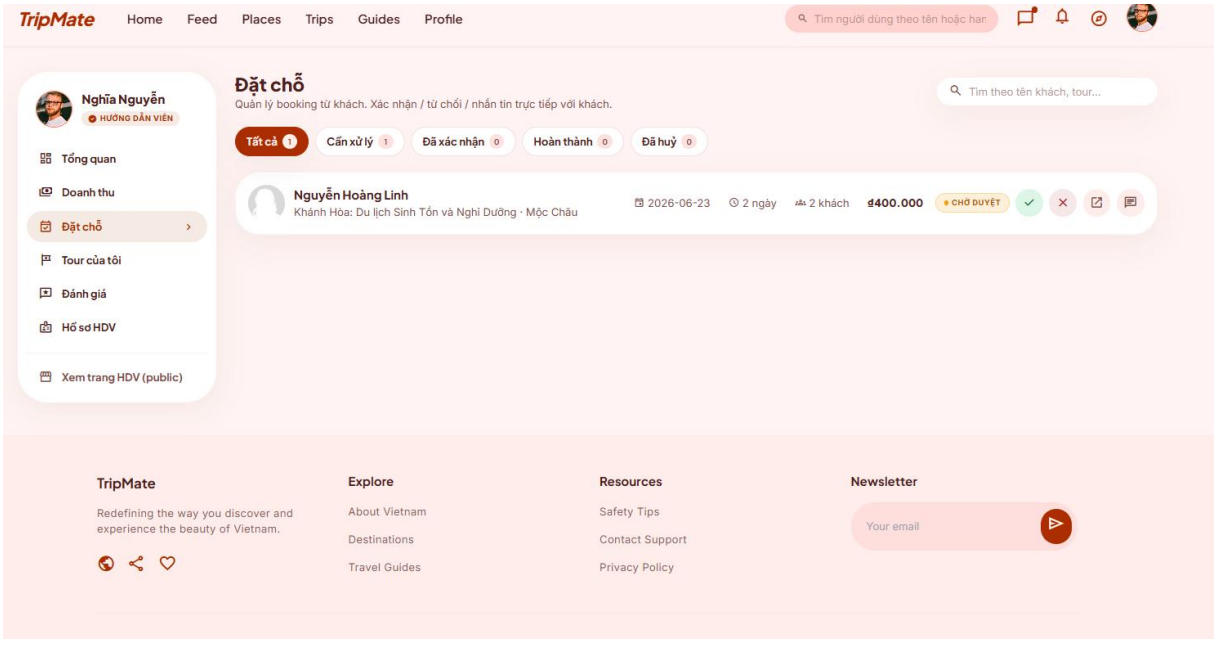
### *3.2.4. Tính năng hướng dẫn viên*

- Đăng ký làm hướng dẫn viên



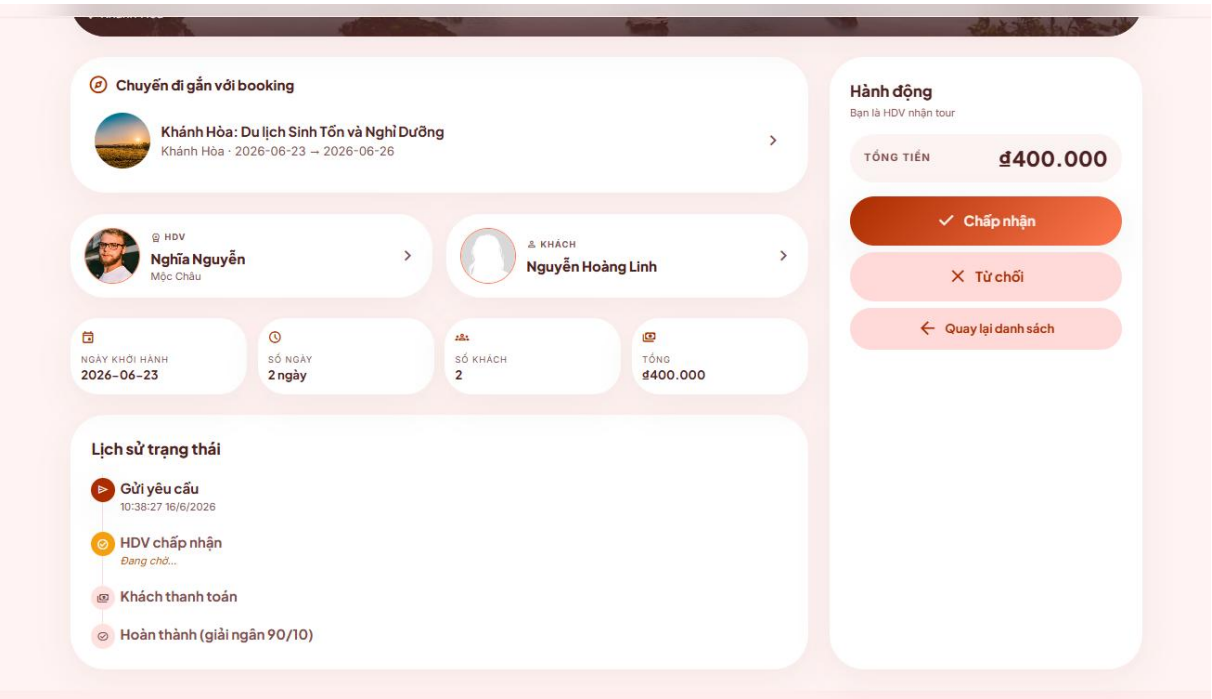

*Hình 3.12. Màn hình chức năng đăng ký làm hướng dẫn viên*

- Quản lý đặt lịch : Hướng dẫn viên có thể xem chi tiết đặt lịch, chấp nhận , từ chối yêu cầu



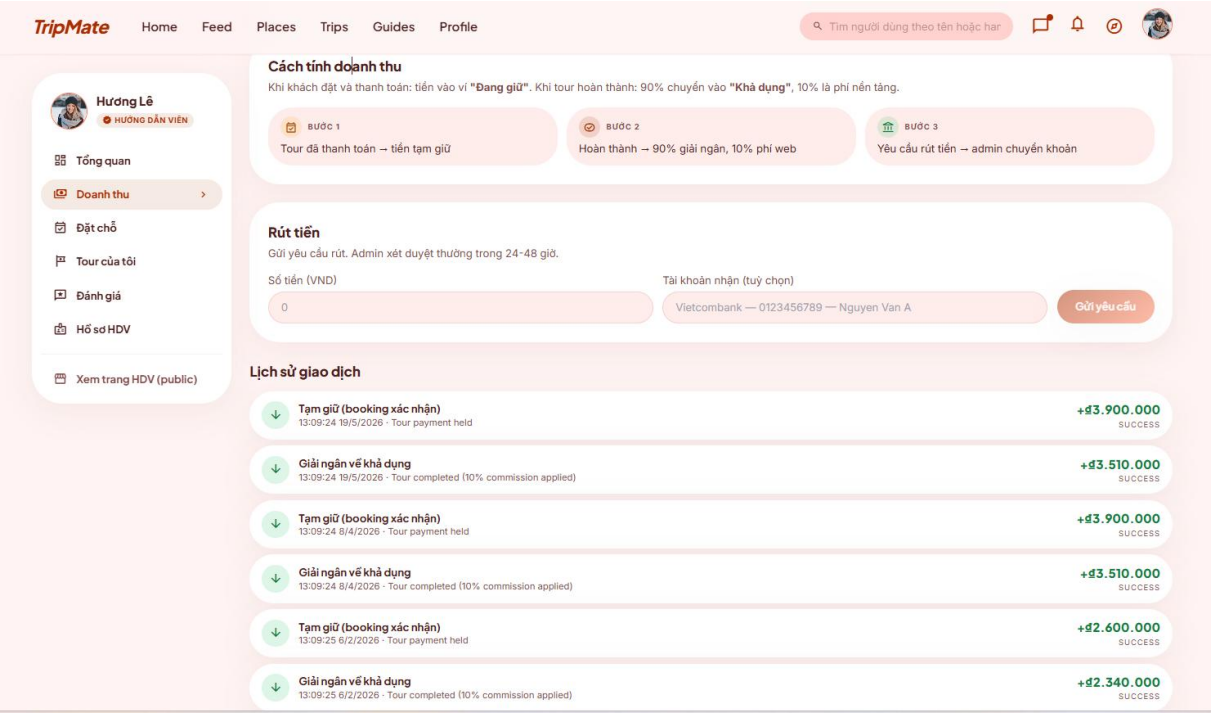

*Hình 3.13. Màn hình chức năng quản lý đặt lịch*

- Chi tiết đặt lịch : hiển thị thông tin toàn bộ đặt lịch và trạng thái




*Hình 3.14. Màn hình chức năng chi tiết đặc lịch*

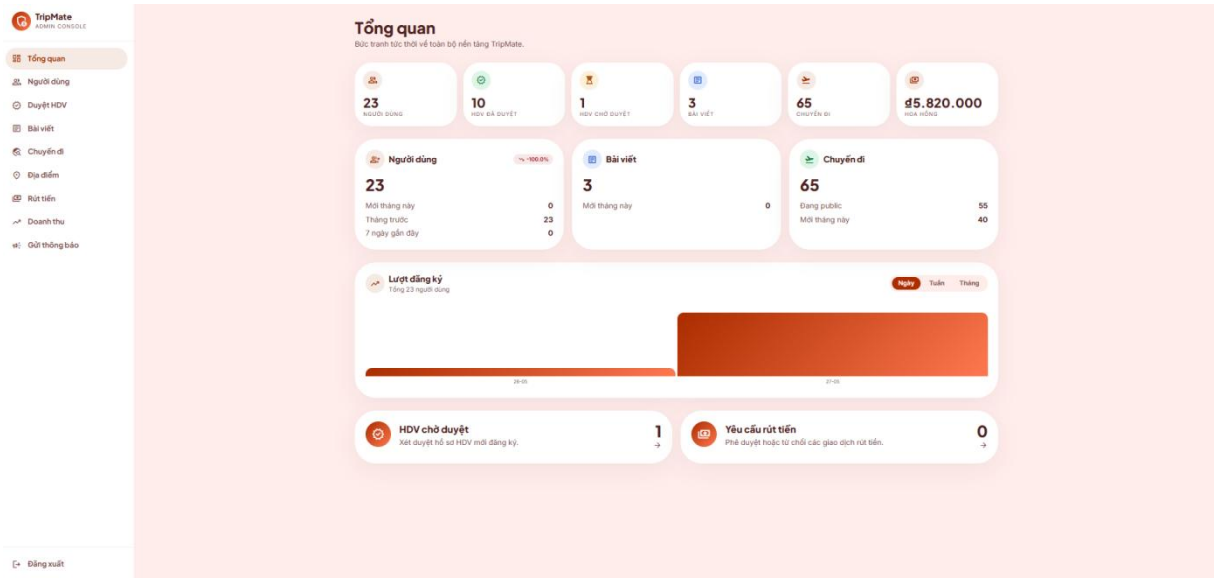
- Quản lý doanh thu: Hướng dẫn viên sẽ biết được toàn bộ số tiền nhận và chi trả hoa hồng , thống kê và rút tiền




*Hình 3.15. Màn hình chức năng quản lý doanh thu*

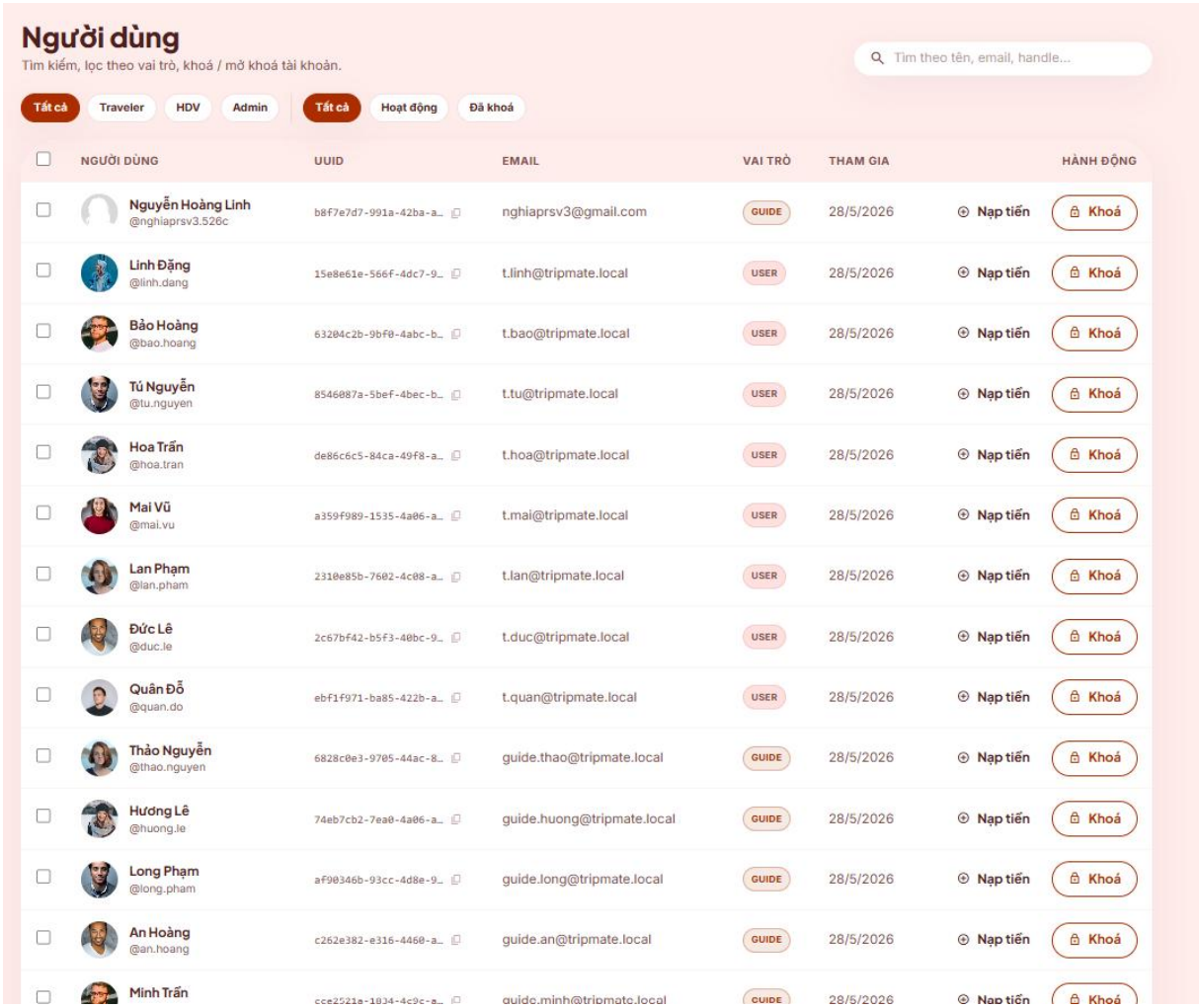
### *3.2.5. Giao diện Admin*

- Tổng quan hệ thống: thống kê toàn bộ hệ thống



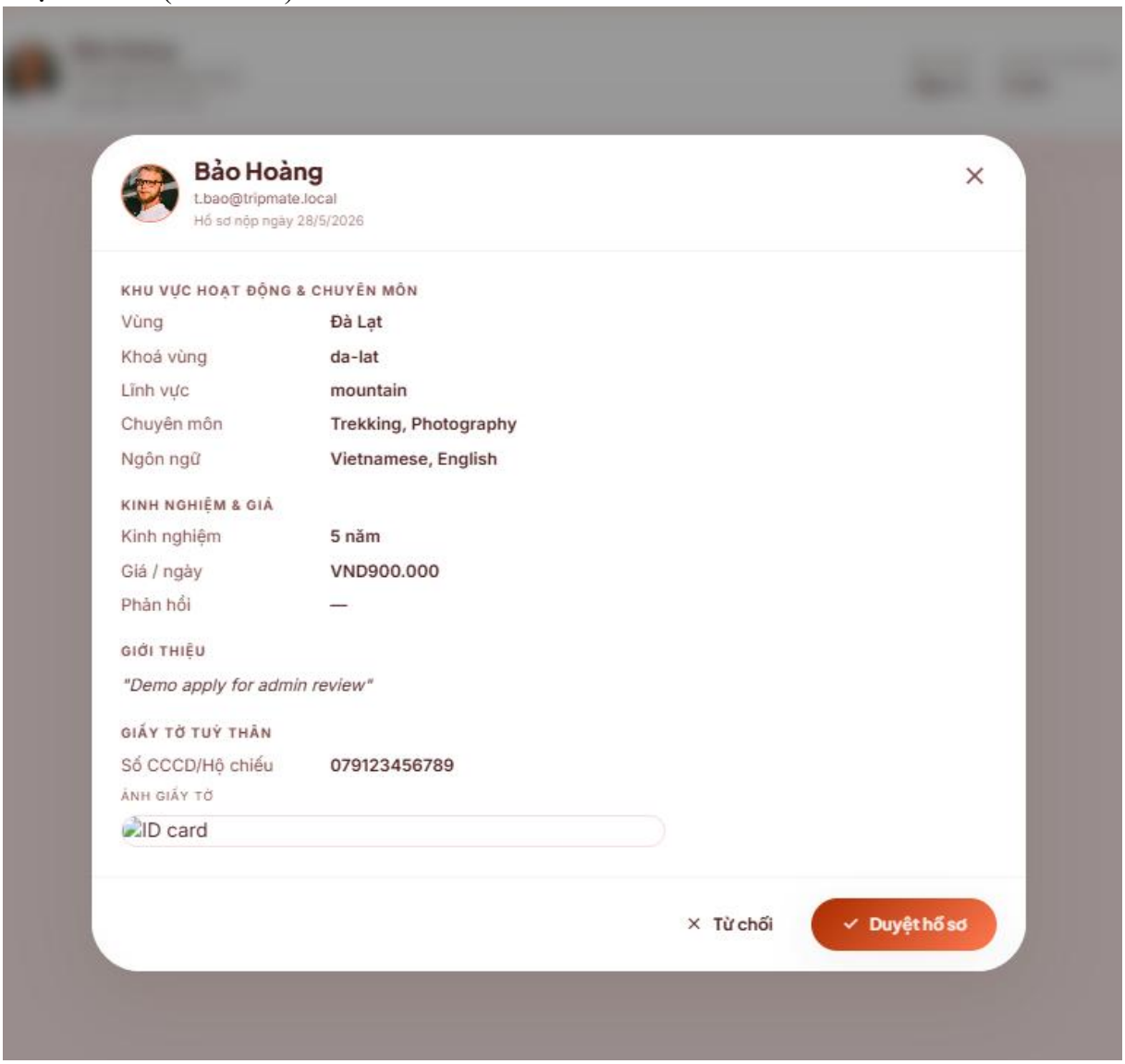

*Hình 3.16. Màn hình chức năng thống kê của Admin*

- Quản lý người dùng : xem danh sách và chi tiết người dùng, nạp tiền thủ công nếu sepay lỗi hoặc người dùng chuyển nhầm nội dung, khóa/mở tài khoản



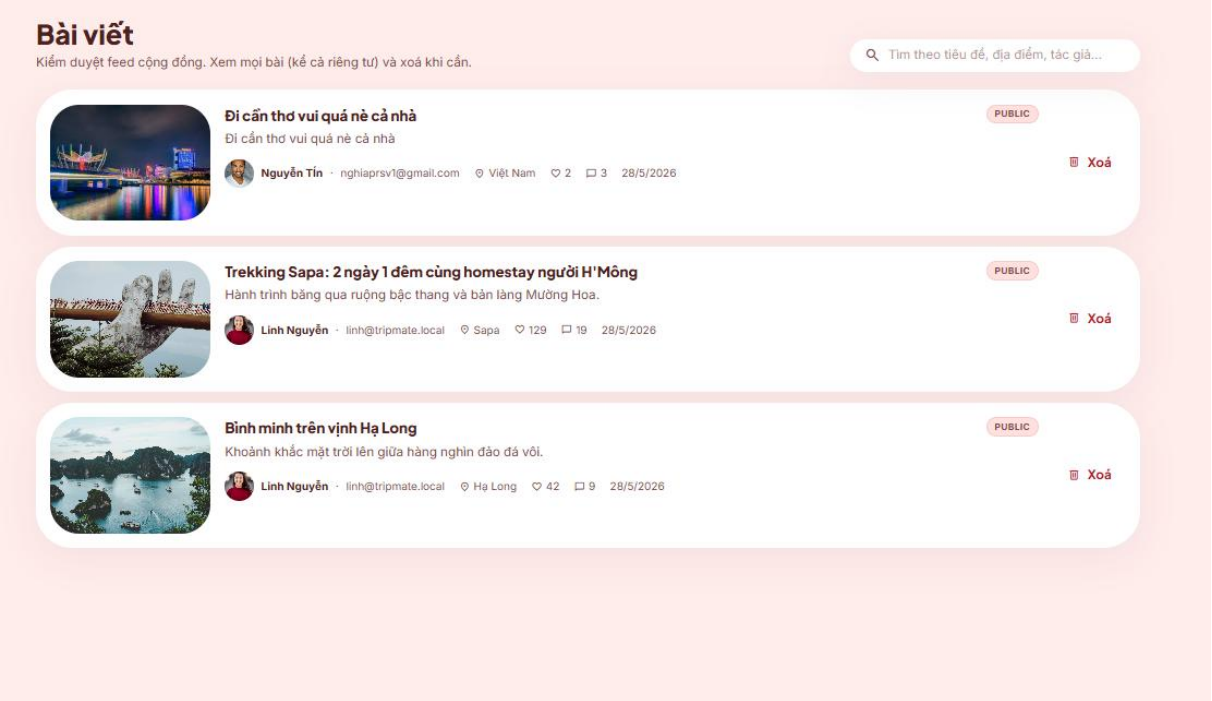

*Hình 3.17. Màn hình chức năng quản lý user*

- Duyệt hướng dẫn viên : xem danh sách request, xem chi tiết một request, duyệt hoặc từ chối(nêu lí do)



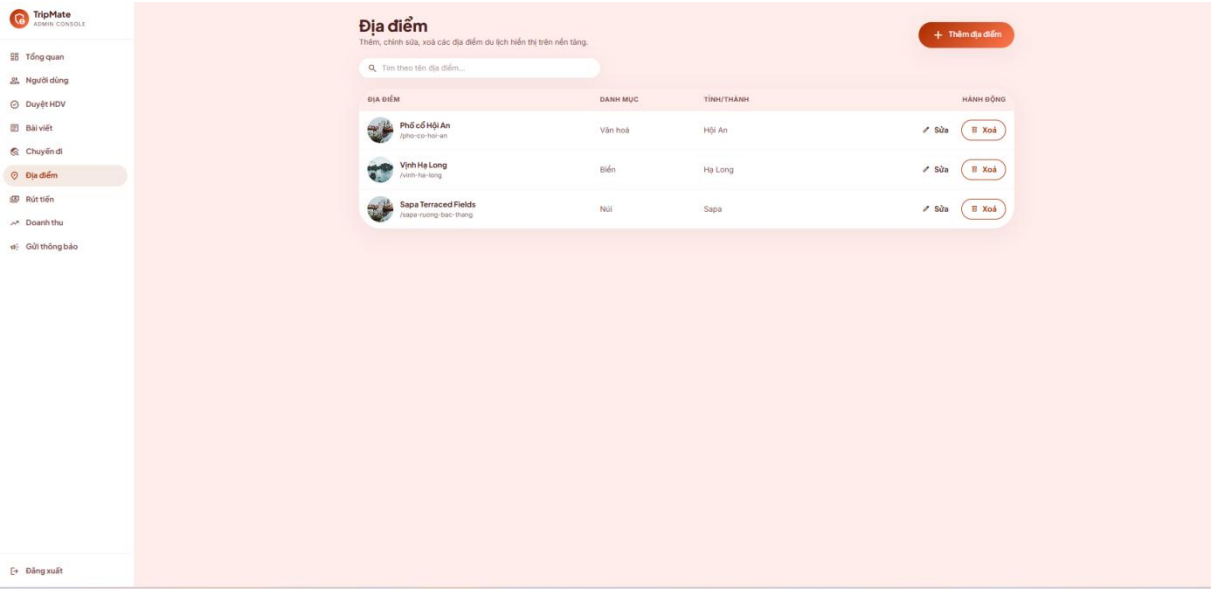

*Hình 3.18. Màn hình chức năng duyệt hồ sơ hướng dẫn viên*

- Quản lý bài viết : xem chi tiết, xóa ( nêu lí do)



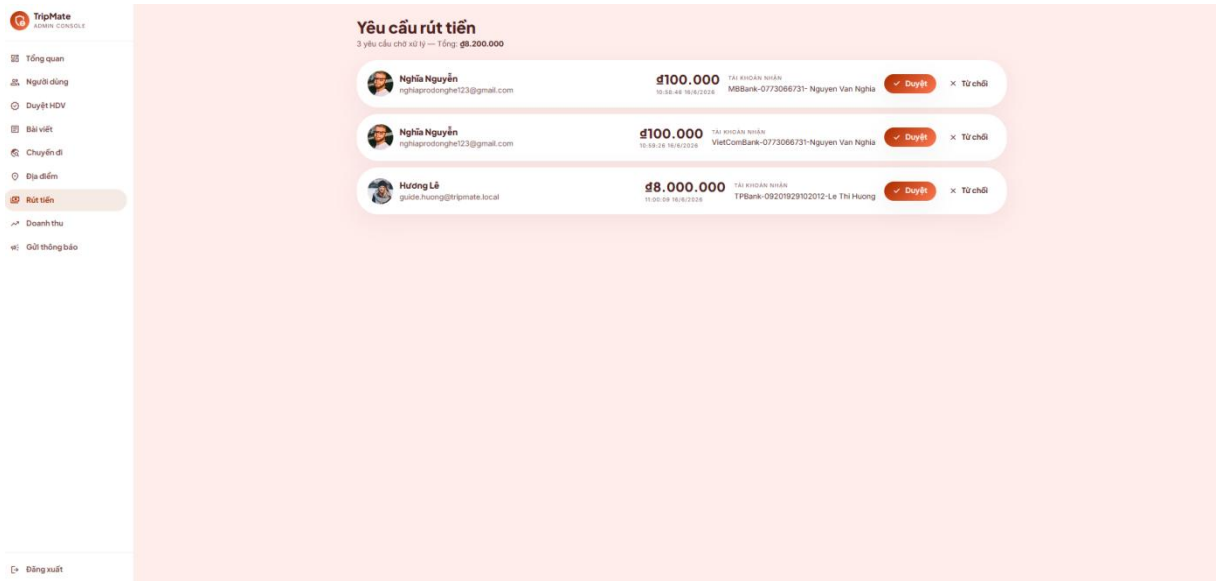

*Hình 3.19. Màn hình chức năng quản lý bài viết (Admin)*

- Quản lí địa điểm : thêm, xóa, xem chi tiết một địa điểm



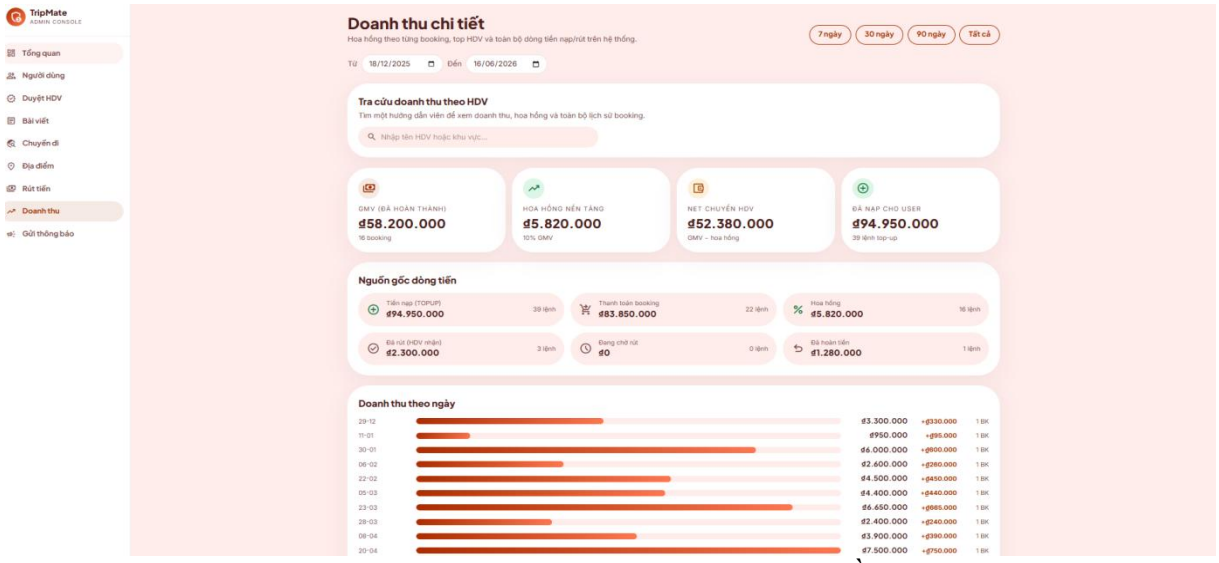

*Hình 3.20. Màn hình chức năng quản lý địa điểm*

- Quản lý rút tiền: duyệt hoặc từ chối rút tiền nêu lí do



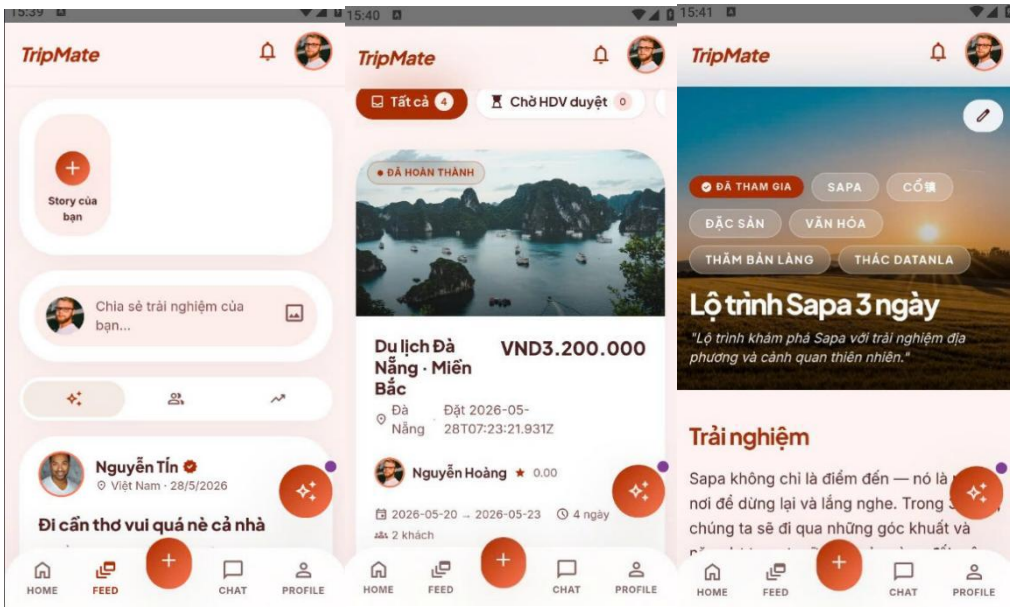

*Hình 3.21. Màn hình chức năng xử lý rút tiền*

- Xem chi tiết doanh thu hệ thống: Hoa hồng theo từng booking, top HDV và toàn bộ dòng tiền nạp/rút trên hệ thống.




*Hình 3.22. Màn hình chức năng xử lý rút tiền*

### 5.2.3. Một số hình ảnh trên ứng dụng Android




*Hình 3.23. Kết quả trên giao ứng dụng*

# KẾT LUẬN

Đề tài ***"Xây dựng hệ thống hỗ trợ du lịch cá nhân và chia sẽ trải nghiệm du lịch giữa người dùng"*** đã được hoàn thành với đầy đủ các mục tiêu đề ra ban đầu. Sản phẩm là một nền tảng full-stack hiện đại, vận hành ổn định, sẵn sàng demo và có tiềm năng triển khai thực tế.

Qua quá trình thực hiện đề tài, em đã:

- Nắm vững kiến trúc full-stack hiện đại với React + NestJS + PostgreSQL.
- Hiểu sâu về REST API, JWT auth, ORM, validation, error handling.
- Thực hành triển khai realtime với Socket.IO.
- Tự host và vận hành mô hình ngôn ngữ (LLM) cục bộ kết hợp RAG, thiết kế kiến trúc đa nhà cung cấp; tích hợp các dịch vụ (SePay, Firebase, Leaflet/OpenStreetMap).
- Lần đầu đóng gói thành công ứng dụng Android từ codebase web qua Capacitor.
- Rèn luyện tư duy phân tích – thiết kế hệ thống đầy đủ từ Use Case đến CSDL.
- Trang bị kỹ năng làm việc độc lập với đề tài lớn, biết phân chia công việc thành các milestone nhỏ.

***Hạn chế***

Ngoài những kết quả đã đạt được trong quá trình thực hiện đề tài. Hệ thống còn một số hạn chế:

- Chưa triển khai được ứng dụng trên hệ điều hành IOS
- Chưa cho mô hình tự học và tự train trong quá trình sử dụng của người dùng
- Xây dựng hệ thống với nhiều chức năng nhưng chưa hoàn toàn đầy đủ nghiệp vụ

***Hướng phát triển***

***- Cải tiến ngắn hạn***

- Rate limit + CAPTCHA : để tránh việc người dùng hoặc bên thứ ba phá
- i18n: thêm bản dịch tiếng Anh, mở rộng cho khách quốc tế.

***- Mở rộng tính năng***

- iOS app: đóng gói thêm bản iOS qua Capacitor (cần Mac + Apple Developer).
- Recommendation engine: nâng cấp từ chấm điểm có trọng số (hiện tại) lên ML model (collaborative filtering với matrix factorization) để học hành vi ngầm.
- Video & Reels: cho phép đăng video ngắn dạng feed.

# TÀI LIỆU THAM KHẢO


[1]  Masse, Mark. REST API design rulebook: designing consistent RESTful web service interfaces. " O'Reilly Media, Inc.", 2011.

[2] React, “React Documentation,” [Trực tuyến]. Xem tại: https://react.dev/. [Truy cập: Mar. 16, 2026].

[3]Vite, “Vite Documentation,” [Trực tuyến]. Xem tại: https://200lab.io/blog/vite-la-gi . [Truy cập: Mar. 16, 2026].

[4]TanStack Query, “TanStack Query Documentation,” [Trực tuyến]. Xem tại: https://tanstack.com/query/latest/docs/framework/react/overview. [Truy cập: Mar. 16, 2026].

[5] DRAKE, Joshua D.; WORSLEY, John C. Practical PostgreSQL. " O'Reilly Media, Inc.", 2002.

[6] Zhadko-Bazilevych, Serhii. "Analysis of ORM framework approaches for Node. js." Journal of Computer Sciences Institute 37 (2025): 426-430.

[7] Arslan, Muhammad, et al. "A Survey on RAG with LLMs." Procedia computer science 246 (2024): 3781-3790.

[8] Jing, Jin, Abdelsalam Sumi Helal, and Ahmed Elmagarmid. "Client-server computing in mobile environments." ACM computing surveys (CSUR) 31.2 (1999): 117-157.

[9] JWT Token , “JWT Documentation,” [Trực tuyến]. Xem tại: https://devwork.vn/blog/json-web-token-jwt-la-gi . [Truy cập: Mar. 17, 2026].

[10] Socket.IO , “Socket.IO  Documentation,” [Trực tuyến]. Xem tại: https://socket.io/docs/v4/ . [Truy cập: Mar. 17, 2026].

# PHỤ LỤC

Mã nguồn của đề tài được công khai trên GitHub tại địa chỉ sau:

https://github.com/nghiaprsv1/DoAnTotNghiepv3