

TRƯỜNG ĐẠI HỌC KIẾN TRÚC ĐÀ NẴNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI:

PHÁT TRIỂN ỨNG DỤNG ĐẶT VÉ SỰ KIỆN ĐA NỀN TẢNG SỬ DỤNG FLUTTER VÀ FIREBASE

Người hướng dẫn: TS.TRƯỜNG NGỌC CHÂU

Sinh viên thực hiện: LÊ VĂN NHẬT

Mã số sinh viên: 2151220007

Lớp: 21CT1

Đà Nẵng 12/2025

TRƯỜNG ĐẠI HỌC KIẾN TRÚC ĐÀ NẴNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP
NGÀNH: CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI:

**PHÁT TRIỂN ỨNG DỤNG ĐẶT VÉ SỰ
KIỆN ĐA NỀN TẢNG SỬ DỤNG
FLUTTER VÀ FIREBASE**

Người hướng dẫn: **TS. TRƯƠNG NGỌC CHÂU**
Sinh viên thực hiện: **LÊ VĂN NHẬT**
Mã số sinh viên: **2151220007**
Lớp: **21CT1**

Đà Nẵng, 12/2025

**BẢN NHẬN XÉT CỦA GIÁNG VIÊN PHẢN BIỆN/ ĐÁNH GIÁ
ĐỒ ÁN/ KHÓA LUẬN TỐT NGHIỆP**

(GVPB nộp Bản nhận xét này về Văn phòng Khoa)

1. Họ và tên sinh viên/ nhóm sinh viên được giao đề tài (sĩ số trong nhóm.....):

(1)..... MSSV: Lớp:

(2)..... MSSV: Lớp:

(3)..... MSSV: Lớp:

2. Tên đề tài:

3. Nhận xét:

a) Những kết quả đạt được của ĐA/KLTN:

.....

.....

.....

.....

.....

.....

b) Những hạn chế của ĐA/KLTN:

.....

.....

.....

.....

.....

.....

4. Đề nghị:

Được bảo vệ ☐

Bổ sung thêm để bảo vệ ☐

Không được bảo vệ ☐

5. Các câu hỏi sinh viên cần trả lời trước Hội đồng:

(1)

.....

(2)

.....

(3)

.....

Đà Nẵng, ngày ... tháng ... năm

Giảng viên phản biện

(Ký và ghi rõ họ tên)

TÓM TẮT

Tên đề tài: Phát triển ứng dụng đặt vé sự kiện đa nền tảng sử dụng Flutter và Firebase

Sinh viên thực hiện : Lê Văn Nhật

Mã số SV: 2151220007 - Lớp: 21CT1

Đề tài sẽ tập trung vào việc áp dụng các công nghệ di động hiện đại như Flutter cho giao diện người dùng , Firebase và Supabase làm backend (bao gồm Authentication, Firestore, Cloud Functions). Mục tiêu là xây dựng một nền tảng di động cho phép người dùng dễ dàng tìm kiếm , đặt vé , và quản lý vé cá nhân. Hệ thống sẽ được tối ưu hóa cho cả giao diện người dùng (UX/UI) và hiệu suất vận hành , tích hợp các chức năng cốt lõi như thanh toán , check-in bằng mã QR, đồng thời đảm bảo tính bảo mật thông qua hệ thống phân quyền admin và người dùng.

Ứng dụng đặt vé EventGo sẽ là một công cụ hỗ trợ đắc lực cho cả người dùng cuối lẫn các nhà tổ chức sự kiện. Hệ thống giúp người dùng tiết kiệm thời gian, tối ưu hóa quy trình tìm kiếm và mua vé, đồng thời cung cấp cho quản trị viên một công cụ mạnh mẽ để quản lý sự kiện, phê duyệt vé và theo dõi thống kê, qua đó tạo dựng sự hài lòng cho người dùng.

LỜI MỞ ĐẦU

Để hoàn thành được Đồ án tốt nghiệp này em xin chân thành cảm ơn quý thầy cô trong khoa Công Nghệ Thông Tin - Trường Đại Học Kiến Trúc Đà Nẵng. Với vốn kiến thức được tiếp thu trong quá trình học không chỉ là nền tảng cho quá trình làm đồ án tốt nghiệp mà còn là hành trang quý báu để em bước vào đời một cách vững chắc và tự tin.

Em cũng xin gửi lời cảm ơn chân thành và sự tri ân sâu sắc với thầy TS.Trương Ngọc Châu trong suốt thời gian vừa qua đã nhiệt tình chỉ dạy, giúp đỡ em để có thể hoàn thành tốt đồ án tốt nghiệp. Trong quá trình làm bài đồ án tốt nghiệp, do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên không thể tránh khỏi những thiếu sót, em rất mong nhận được ý kiến đóng góp của quý báu của các thầy cô để em có thể học hỏi thêm được nhiều kinh nghiệm, kiến thức.

Em xin trân trọng cảm ơn!

CAM ĐOAN

Em xin cam đoan kết quả đạt được trong đồ án tốt nghiệp là sản phẩm của riêng cá nhân dưới sự hướng dẫn của TS.Trương Ngọc Châu, không sao chép lại của người khác. Trong toàn bộ nội dung của đồ án, những điều được trình bày hoặc là của bản thân hoặc là được tổng hợp từ nhiều nguồn tài liệu. Tất cả các tài liệu tham khảo đều có xuất xứ rõ ràng và được trích dẫn hợp pháp. Em xin hoàn toàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan của mình.

Sinh viên thực hiện

Lê Văn Nhật

MỤC LỤC

TÓM TẮT.....	
LỜI MỞ ĐẦU.....	i
CAM ĐOAN.....	ii
MỤC LỤC	iii
DANH MỤC CÁC BẢNG	vi
DANH MỤC CÁC HÌNH VẼ	vii
DANH SÁCH CÁC CHỮ VIẾT TẮT	ix
CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN.....	1
1.1. Lý do chọn đề tài.....	1
1.2. Lịch sử nghiên cứu.....	1
1.3. Mục tiêu đề tài.....	2
1.4. Đối tượng, phạm vi nghiên cứu.....	3
1.4.1. Đối tượng nghiên cứu	3
1.4.2. Phạm vi nghiên cứu.....	4
1.5. Cơ sở lý thuyết	5
1.5.1. Giới thiệu về ngôn ngữ Dart	5
1.5.2. Giới thiệu về framework Flutter	7
1.5.3. Giới thiệu về Firebase	10
1.5.4. Giới thiệu về Supabase.....	12
CHƯƠNG 2: PHÂN TÍCH, THIẾT KẾ HỆ THỐNG	16
2.1. Xác định yêu cầu	16
2.2. Phân tích yêu cầu.....	17
2.2.1 Chức năng đăng nhập.....	17
2.2.2 Chức năng đặt vé sự kiện.....	18
2.2.3 Chức năng thêm , sửa, xóa thông tin sự kiện	18
2.2.4 Chức năng tìm kiếm sự kiện	18

2.2.5 Chức năng quản lý order	19
2.2.6 Chức năng check-in sự kiện	19
2.2.7 Chức năng quản lý khách hàng	19
2.2.8 Chức năng quản lý nhân viên	19
2.3. Mô hình hóa yêu cầu	20
2.3.1 Chức năng đăng nhập	20
2.3.2 Chức năng đặt vé sự kiện	22
2.3.3 Chức năng quản lý sự kiện	25
2.3.4 Chức năng tìm kiếm sự kiện	29
2.3.5 Chức năng quản lý khách hàng	31
2.3.6 Chức năng quản lý order	34
2.3.7 Chức năng check-in sự kiện	37
2.3.8 Chức năng quản lý nhân viên	39
2.4. Thiết kế cơ sở dữ liệu	45
2.4.1 Mô hình quan hệ	45
2.4.2 Mô hình sơ đồ lớp	45
2.5. Thiết kế giao diện	46
2.5.1 Giao diện Admin	46
2.5.2 Giao diện người dùng	50
CHƯƠNG 3 : CÀI ĐẶT VÀ PHÁT TRIỂN ỨNG DỤNG	56
3.1. Cấu trúc thư mục code	56
3.1.1. Cấu trúc tổng thể thư mục dự án Flutter	56
3.2. Xử lý thanh toán qua Zalopay	57
3.2.1. Vấn đề	57
3.2.2. Cách giải quyết	57
3.2.3. Module code	58
3.3. Xử lý vấn đề bảo mật thông tin	63
3.3.1. Vấn đề	63

3.3.2. Cách giải quyết.....	63
3.3.3. Module code.....	64
KẾT LUẬN	68
I. KẾT QUẢ ĐẠT ĐƯỢC	68
II. HẠN CHẾ.....	69
III. HƯỚNG PHÁT TRIỂN	69
TÀI LIỆU THAM KHẢO	71

DANH MỤC CÁC BẢNG

Bảng 2.1 Mô tả Use Case đăng nhập	21
Bảng 2.2 Dòng chính Use Case đăng nhập.....	21
Bảng 2.3 Mô tả Use Case đặt vé sự kiện	23
Bảng 2.4 Dòng chính Use Case đặt vé sự kiện.....	23
Bảng 2.5 Mô tả Use Case quản lý sự kiện	26
Bảng 2.6 Dòng chính Use Case quản lý sự kiện.....	26
Bảng 2.7 Mô tả Use Case tìm kiếm sự kiện.....	29
Bảng 2.8 Dòng chính Use Case tìm kiếm sự kiện	30
Bảng 2.9 Mô tả Use Case quản lý khách hàng	32
Bảng 2.10 Dòng chính Use Case quản lý khách hàng.....	32
Bảng 2.11 Mô tả Use Case quản lý order	35
Bảng 2.12 Dòng chính Use Case quản lý order.....	35
Bảng 2.13 Mô tả Use Case check-in sự kiện	38
Bảng 2.14 Dòng chính Use Case check-in sự kiện.....	38
Bảng 2.15 Mô tả Use Case quản lý nhân viên.....	40
Bảng 2.16 Dòng chính Use Case quản lý nhân viên.....	40

DANH MỤC CÁC HÌNH VẼ

Hình 2.1 Biểu đồ Use Case tổng quát.....	20
Hình 2.2 Biểu đồ Use Case đăng nhập	21
Hình 2.3 Biểu đồ tuần tự đăng nhập	22
Hình 2.4 Biểu đồ Use Case đặt vé sự kiện.....	23
Hình 2.5 Biểu đồ tuần tự đặt vé sự kiện	25
Hình 2.6 Biểu đồ Use Case quản lý sự kiện	26
Hình 2.7 Biểu đồ tuần tự thêm sự kiện	27
Hình 2.8 Biểu đồ tuần tự sửa sự kiện.....	28
Hình 2.9 Biểu đồ tuần tự xóa sự kiện	29
Hình 2.10 Biểu đồ Use Case tìm kiếm sự kiện.....	29
Hình 2.11 Biểu đồ tuần tự tìm kiếm sự kiện.....	31
Hình 2.12 Biểu đồ Use Case quản lý khách hàng.....	31
Hình 2.13 Biểu đồ tuần tự xem danh sách khách hàng.....	33
Hình 2.14 Biểu đồ tuần tự xóa khách hàng.....	34
Hình 2.15 Biểu đồ Use Case quản lý order	35
Hình 2.16 Biểu đồ tuần tự xem danh sách order	36
Hình 2.17 Biểu đồ tuần tự xem chi tiết order	37
Hình 2.18 Biểu đồ Use Case check-in sự kiện	37
Hình 2.19 Biểu đồ tuần tự check-in sự kiện	39
Hình 2.20 Biểu đồ Use Case quản lý nhân viên	40
Hình 2.21 Biểu đồ tuần tự thêm nhân viên	42
Hình 2.22 Biểu đồ tuần tự sửa nhân viên	43
Hình 2.23 Biểu đồ tuần tự xóa nhân viên	44
Hình 2.24 Mô hình quan hệ	45
Hình 2.25 Mô hình sơ đồ lớp.....	46
Hình 2.26 Màn hình dashboard	46
Hình 2.27 Màn hình quản lý sự kiện	47
Hình 2.28 Màn hình quản lý order.....	48
Hình 2.29 Màn hình quản lý khách hàng.....	49
Hình 2.30 Màn hình trang chủ.....	50
Hình 2.31 Màn hình chi tiết sự kiện	51
Hình 2.32 Màn hình xác minh bot.....	52
Hình 2.33 Màn hình thanh toán	53

Hình 2.34 Màn hình vé của tôi	54
Hình 2.35 Màn hình tài khoản	55

DANH SÁCH CÁC CHỮ VIẾT TẮT

Từ viết tắt	Tên đầy đủ	Diễn giải
AI	Artificial Intelligence	Trí tuệ nhân tạo, công nghệ mô phỏng trí thông minh con người được đề xuất tích hợp để gợi ý sự kiện.
AOT	Ahead-of-Time	Cơ chế biên dịch mã nguồn thành mã máy trước khi chạy, giúp tối ưu hiệu suất ứng dụng khi xuất bản.
API	Application Programming Interface	Giao diện lập trình ứng dụng, cho phép các thành phần phần mềm giao tiếp với nhau (ví dụ: ZaloPay API).
ARM	Advanced RISC Machines	Kiến trúc vi xử lý phổ biến trên các thiết bị di động mà Dart hỗ trợ biên dịch.
BaaS	Backend-as-a-Service	Mô hình dịch vụ đám mây cung cấp sẵn hạ tầng backend như xác thực, cơ sở dữ liệu (Firebase, Supabase).
BLoC	Business Logic Component	Một mẫu thiết kế quản lý trạng thái giúp tách biệt logic nghiệp vụ khỏi giao diện người dùng.
CLI	Command Line Interface	Giao diện dòng lệnh dùng để tương tác với hệ thống hoặc công cụ lập trình.
CSDL	Cơ sở dữ liệu	Hệ thống lưu trữ và quản lý dữ liệu có tổ chức (như Firestore, PostgreSQL).
DI	Dependency Injection	Kỹ thuật tiêm sự phụ thuộc giúp quản lý các thành phần và giảm sự phụ thuộc chặt chẽ giữa các module code.

FK	Foreign Key	Khóa ngoại, dùng để tạo liên kết quan hệ giữa các bảng trong cơ sở dữ liệu.
FPS	Frames Per Second	Số khung hình trên giây, chỉ số đo độ mượt mà của chuyển động hình ảnh trong ứng dụng.
ID	Identifier	Mã định danh duy nhất dùng để phân biệt các đối tượng như người dùng, sự kiện, đơn hàng.
I/O	Input/Output	Nhập/Xuất dữ liệu, các tác vụ liên quan đến mạng hoặc đọc ghi tệp tin.
JIT	Just-in-Time	Cơ chế biên dịch tức thời trong quá trình chạy, hỗ trợ tính năng Hot Reload giúp xem nhanh thay đổi.
JSON	JavaScript Object Notation	Định dạng dữ liệu văn bản nhẹ dùng để lưu trữ và trao đổi thông tin giữa client và server.
JWT	JSON Web Token	Chuẩn mở dùng để tạo token truy cập an toàn, phục vụ xác thực người dùng trong Supabase.
ML	Machine Learning	Học máy, một lĩnh vực của AI giúp hệ thống tự học từ dữ liệu lịch sử.
MVP	Minimum Viable Product	Sản phẩm khả dụng tối thiểu, phiên bản sản phẩm với các tính năng cơ bản nhất để triển khai nhanh.
MVVM	Model-View-ViewModel	Mô hình kiến trúc phần mềm tách biệt logic xử lý (ViewModel) và giao diện hiển thị (View).
NoSQL	Not Only SQL	Loại cơ sở dữ liệu phi quan hệ, lưu trữ dữ liệu linh hoạt (ví dụ: Cloud Firestore).
OOP	Object-Oriented Programming	Lập trình hướng đối tượng với các đặc tính như kế thừa, đa hình, đóng gói.

PDF	Portable Document Format	Định dạng tài liệu di động, dùng để tạo và gửi vé điện tử cho người dùng.
PK	Primary Key	Khóa chính, giá trị định danh duy nhất cho mỗi bản ghi trong bảng cơ sở dữ liệu.
QR	Quick Response	Mã phản hồi nhanh (mã vạch 2 chiều), dùng để tạo vé điện tử và check-in tại sự kiện.
RLS	Row-Level Security	Tính năng bảo mật cấp hàng của Supabase/PostgreSQL, kiểm soát quyền truy cập từng dòng dữ liệu.
SDK	Software Development Kit	Bộ công cụ phát triển phần mềm tích hợp sẵn (ví dụ: ZaloPay SDK cho thanh toán).
SQL	Structured Query Language	Ngôn ngữ truy vấn có cấu trúc dùng để quản lý và thao tác với cơ sở dữ liệu quan hệ.
SSL	Secure Sockets Layer	Tiêu chuẩn an ninh công nghệ toàn cầu tạo liên kết được mã hóa giữa máy chủ và trình duyệt.
SV	Sinh viên	Người thực hiện đề tài tốt nghiệp.
TS	Tiến sĩ	Học vị của giảng viên hướng dẫn đồ án.
UI	User Interface	Giao diện người dùng, nơi tương tác trực tiếp giữa người sử dụng và hệ thống.
UX	User Experience	Trải nghiệm người dùng, cảm nhận và mức độ hài lòng của người dùng khi sử dụng sản phẩm.
VM	Virtual Machine	Máy ảo, môi trường phần mềm mô phỏng hệ thống máy tính (ví dụ: Dart VM).

CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN

1.1. Lý do chọn đề tài

Trong thời đại kỹ thuật số, nhu cầu tìm kiếm, tham gia và quản lý các sự kiện như concert, hòa nhạc, hội thảo, triển lãm ngày càng tăng cao. Tuy nhiên, các nền tảng hiện có như Ticketmaster hay Eventbrite thường tập trung vào thị trường quốc tế, chưa thực sự tối ưu cho các sự kiện nội địa tại Việt Nam. Điều này tạo ra một khoảng trống cho một ứng dụng thân thiện, thông minh và tập trung vào trải nghiệm người dùng trong nước.

Đề tài này đề xuất xây dựng ứng dụng di động EventGo, một nền tảng toàn diện cho phép người dùng khám phá, đặt vé, và quản lý sự kiện một cách liền mạch. Điểm khác biệt của dự án là việc tích hợp các tính năng thông minh như check-in bằng mã QR tiện lợi, và các tính năng cộng đồng để làm nổi bật ứng dụng so với các đối thủ.

Để hiện thực hóa ý tưởng, đề tài lựa chọn bộ công nghệ hiện đại và mạnh mẽ bao gồm:

- Flutter cho việc xây dựng giao diện người dùng đa nền tảng (cross-platform), đảm bảo trải nghiệm mượt mà và đồng nhất.
- Supabase và Firebase được sử dụng làm backend (Backend-as-a-Service), cung cấp các dịch vụ toàn diện từ xác thực người dùng (Supabase Auth), cơ sở dữ liệu (Firestore), lưu trữ (Storage) cho đến các tác vụ logic phía sau (Cloud Functions).

Dự án không chỉ giải quyết một nhu cầu thực tế của thị trường mà còn là cơ hội để áp dụng và làm chủ các công nghệ phát triển ứng dụng di động tiên tiến.

1.2. Lịch sử nghiên cứu

Trong nước:

– Hiện tại, thị trường ứng dụng (app) và web đặt vé sự kiện tại Việt Nam đã phát triển, nhưng chủ yếu rơi vào hai nhóm chính: các nền tảng chuyên biệt (như Ticketbox) và các siêu ứng dụng có tích hợp tính năng bán vé (như Momo, ZaloPay, VinID).– Hạn

chế của các siêu ứng dụng là việc đặt vé thường chỉ là một tính năng phụ, chưa được tối ưu hóa hoàn toàn trải nghiệm người dùng cho việc khám phá, quản lý vé và kết nối cộng đồng. Các nền tảng chuyên biệt tuy mạnh nhưng có thể chưa tập trung sâu vào các tính năng gợi ý sự kiện bằng AI hoặc các tính năng xã hội hóa. – Một số ứng dụng tương tự: Ticketbox, Momo (mục Sự kiện), ZaloPay (mục Vé),...

Ngoài nước:

– Thị trường quốc tế có rất nhiều nền tảng đặt vé sự kiện lớn và rất chuyên nghiệp, đã hoạt động lâu năm và chiếm thị phần lớn, ví dụ như Eventbrite và Ticketmaster. – Các hệ thống này cực kỳ mạnh mẽ về quy trình đặt vé, quản lý số lượng lớn người dùng và các sự kiện quy mô toàn cầu. – Hạn chế: Các nền tảng này tập trung vào thị trường quốc tế và có thể không tối ưu trải nghiệm cho các sự kiện mang tính nội địa, quy mô nhỏ tại Việt Nam. Ngoài ra, việc tích hợp sâu các cổng thanh toán nội địa (như MoMo, ZaloPay mà EventGo hướng tới) có thể không phải là ưu tiên của họ, gây ít nhiều bất tiện cho người dùng trong nước.

1.3. Mục tiêu đề tài

Mục tiêu chính của đề tài là xây dựng một ứng dụng di động hoàn chỉnh có tên EventGo, cung cấp một giải pháp đặt vé sự kiện hiệu quả, thông minh và thân thiện cho người dùng.

Các mục tiêu cụ thể:

- Quản lý Sự kiện và Người dùng:

- Xây dựng hệ thống đăng ký, đăng nhập, quên mật khẩu và đặt lại mật khẩu an toàn cho người dùng sử dụng Supabase Auth.

- Phân quyền rõ ràng giữa người dùng thông thường và Quản trị viên (Admin) để quản lý hệ thống.

- Khám phá và Tìm kiếm Sự kiện:

- Hiện thị danh sách sự kiện nổi bật, theo xu hướng và gần vị trí người dùng.

- Cung cấp chức năng tìm kiếm và lọc sự kiện nâng cao theo tên, địa điểm, thời gian và thể loại.

- Đặt vé và Thanh toán:
 - Xây dựng quy trình đặt vé trực quan, cho phép chọn loại vé, số lượng..
 - Tích hợp cổng thanh toán trực tuyến (mô phỏng Stripe, MoMo, ZaloPay) để xử lý giao dịch an toàn.
- Quản lý vé và Check-in:
 - Cung cấp cho người dùng vé điện tử dưới dạng mã QR sau khi đặt thành công.
 - Phát triển tính năng quét mã QR để check-in tại sự kiện, giúp quy trình diễn ra nhanh chóng.

Mục tiêu công nghệ:

- Áp dụng kiến trúc Clean Architecture (MVVM hoặc Provider) để đảm bảo mã nguồn dễ bảo trì và mở rộng..
- Sử dụng Supabase cho việc xác thực người dùng và tích hợp các dịch vụ của Firebase, Supabase cho các phần còn lại (Firestore, Functions, Messaging, Storage, Auth).
- Công cụ phát triển: Android Studio, Firebase, Supabase.

1.4. Đối tượng, phạm vi nghiên cứu

1.4.1. Đối tượng nghiên cứu

Để đảm bảo dự án được xây dựng trên một nền tảng vững chắc và đáp ứng đúng nhu cầu thực tế, đề tài sẽ tập trung nghiên cứu sâu vào các đối tượng sau:

- Quy trình nghiệp vụ của hệ thống đặt vé sự kiện:
 - Phân tích chi tiết các luồng nghiệp vụ cốt lõi từ đầu đến cuối, bao gồm: quy trình đăng tải và duyệt sự kiện, luồng đặt vé và thanh toán của người dùng (booking flow), cơ chế tạo và xác thực vé điện tử (QR code), và quy trình check-in tại địa điểm tổ chức.
- Hành vi và nhu cầu của người dùng mục tiêu:

- Tiến hành nghiên cứu hành trình người dùng (user journey) để xác định các "điểm chạm" và những khó khăn họ gặp phải với các giải pháp hiện có. Đối tượng nghiên cứu chính là sinh viên, người trẻ và nhân viên văn phòng tại các thành phố lớn, nhằm xây dựng một sản phẩm có trải nghiệm người dùng (UX) vượt trội.:

- Kiến trúc và các mẫu thiết kế phần mềm hiện đại:

- Nghiên cứu và áp dụng các mô hình kiến trúc tiên tiến như Clean Architecture để đảm bảo mã nguồn có khả năng bảo trì, mở rộng và kiểm thử dễ dàng. Các mẫu quản lý trạng thái (State Management) như BLoC hoặc Provider sẽ được phân tích để lựa chọn giải pháp tối ưu cho ứng dụng.

- Công nghệ và hệ sinh thái Backend-as-a-Service (BaaS):

- Flutter: Nghiên cứu sâu về khả năng xây dựng giao diện người dùng hiệu suất cao, đồng nhất trên cả hai nền tảng Android và iOS.

- Supabase: Tập trung nghiên cứu và triển khai giải pháp xác thực mạnh mẽ, an toàn và linh hoạt của Supabase, bao gồm cả các luồng đăng nhập qua mạng xã hội và đặt lại mật khẩu.

- Firebase: Tập trung nghiên cứu và triển khai các dịch vụ quan trọng như Firestore, Cloud Functions, Storage và Messaging để phục vụ lưu trữ dữ liệu sự kiện, đơn hàng, gửi thông báo và xử lý các tác vụ backend của hệ thống.

1.4.2. Phạm vi nghiên cứu

Phạm vi chức năng của hệ thống

Để đảm bảo tính khả thi của đề án trong khung thời gian cho phép, dự án sẽ có một số giới hạn sau:

Nền tảng: Ứng dụng chỉ tập trung phát triển cho nền tảng di động (Android và iOS) bằng Flutter. Phiên bản web cho người dùng cuối sẽ không được phát triển trong phạm vi của đề án này.

Triển khai: Phạm vi của dự án không bao gồm việc triển khai chính thức sản phẩm lên Google Play Store và Apple App Store, mà tập trung vào việc xây dựng một sản phẩm hoàn chỉnh, được kiểm thử kỹ lưỡng và sẵn sàng để triển khai .

1.5. Cơ sở lý thuyết

1.5.1. Giới thiệu về ngôn ngữ Dart

Dart là một ngôn ngữ lập trình hướng đối tượng, mạnh về kiểu (statically typed), được Google phát triển từ năm 2011 và hiện đã đạt phiên bản Dart 3.x (tính đến năm 2025). Dart được thiết kế với mục tiêu chính là tối ưu cho việc phát triển giao diện người dùng (UI), đặc biệt trong hệ sinh thái Flutter, đồng thời vẫn đủ linh hoạt để sử dụng cho backend (server-side) thông qua Dart Server hoặc Dart VM.

Dart hỗ trợ hai cơ chế biên dịch linh hoạt:

- Just-in-Time (JIT): được sử dụng trong quá trình phát triển, cho phép Hot Reload – tính năng cốt lõi giúp lập trình viên thấy ngay kết quả thay đổi mã nguồn chỉ trong vài trăm mili giây, từ đó tăng tốc độ phát triển và thử nghiệm giao diện lên gấp nhiều lần.
- Ahead-of-Time (AOT): khi xuất bản ứng dụng, mã Dart được biên dịch thành mã máy native cho từng nền tảng (ARM, x64, iOS, Android, Web), đảm bảo hiệu suất tối ưu, khởi động nhanh và không cần runtime bổ sung.

Một trong những tính năng quan trọng nhất của Dart là Null Safety (giới thiệu từ Dart 2.12), cho phép hệ thống kiểu dữ liệu phân biệt rõ ràng giữa biến có thể null (String?) và biến không null (String). Nhờ đó, lỗi null pointer exception – vốn là nguyên nhân phổ biến gây crash ứng dụng – được phát hiện và loại bỏ ngay tại thời điểm biên dịch, giúp mã nguồn an toàn hơn, dễ bảo trì hơn và giảm đáng kể thời gian debug.

Dart cung cấp đầy đủ các đặc trưng của ngôn ngữ lập trình hiện đại, bao gồm:

- Lập trình bất đồng bộ mạnh mẽ với cú pháp async/await, giúp xử lý các tác vụ mạng, I/O, gọi API một cách rõ ràng, dễ đọc và ít lỗi hơn so với callback hay Promise.
- Quản lý bộ nhớ tự động (Garbage Collection) hiệu quả, tối ưu cho ứng dụng di động có tài nguyên hạn chế.

- Hỗ trợ OOP toàn diện: kế thừa (inheritance), đa hình (polymorphism), đóng gói (encapsulation), abstract class, interface.
- Mixin: cơ chế tái sử dụng code linh hoạt mà không cần kế thừa đa lớp.
- Generic: hỗ trợ kiểu dữ liệu tổng quát, tăng tính an toàn và tái sử dụng.
- Extension methods: cho phép mở rộng chức năng cho các lớp có sẵn mà không cần sửa đổi mã gốc.

Từ Dart 3.0, ngôn ngữ được nâng cấp với các tính năng hiện đại và mạnh mẽ hơn:

- Records: kiểu dữ liệu bất biến, nhẹ, dùng để trả về nhiều giá trị từ hàm mà không cần tạo class.
- Patterns: hỗ trợ destructuring, matching mạnh mẽ, giúp xử lý dữ liệu phức tạp ngắn gọn hơn.
- Class Modifiers: như final, sealed, base, interface – giúp kiểm soát kế thừa và triển khai một cách rõ ràng, tránh lạm dụng OOP.

Với cú pháp ngắn gọn, gần gũi với JavaScript, Java và C#, Dart rất dễ học đối với sinh viên và lập trình viên mới, đồng thời đủ mạnh mẽ để xây dựng các ứng dụng doanh nghiệp phức tạp. Nhờ sự kết hợp giữa hiệu suất cao, an toàn kiểu dữ liệu và năng suất phát triển vượt trội, Dart đã trở thành ngôn ngữ nền tảng không thể thiếu trong hệ sinh thái Flutter và đang ngày càng được mở rộng sang các lĩnh vực như backend (Dart Frog, Shelf) và CLI tools.

```
// hello.dart

void main() {

  print('Hello, Dart!');

  // Biến có kiểu (type inference)

  String name = "Nguyen Van A";

  int age = 20;
```

```
// Null safety

String? email; // có thể null

print(email ?? "Chưa có email");

}
```

Lợi ích của Dart:

- An toàn kiểu dữ liệu với Null Safety Dart loại bỏ hoàn toàn lỗi null pointer exception tại thời điểm biên dịch, giúp ứng dụng ổn định hơn, giảm thiểu bug trong quá trình vận hành.
- Hỗ trợ lập trình bất đồng bộ mạnh mẽ Cú pháp async/await được tích hợp sẵn, giúp xử lý các tác vụ như gọi API, đọc database, upload file trở nên dễ đọc, dễ bảo trì và ít lỗi hơn so với callback hay Promise.
- Hiệu suất cao và tối ưu cho UI Dart được thiết kế riêng cho Flutter, với garbage collector hiệu quả và JIT/AOT linh hoạt, đảm bảo ứng dụng vừa nhanh khi phát triển, vừa mượt khi chạy.
- Cú pháp hiện đại, dễ học Gần giống C#, JavaScript và Java → sinh viên và lập trình viên mới có thể học nhanh trong vòng 1–2 tuần.

1.5.2. Giới thiệu về framework Flutter

Flutter là một framework phát triển giao diện người dùng (UI) mã nguồn mở do Google phát triển, được công bố lần đầu vào năm 2017 và đạt phiên bản ổn định 1.0 vào tháng 12 năm 2018. Flutter cho phép lập trình viên xây dựng các ứng dụng di động, web và máy tính để bàn từ một codebase duy nhất với hiệu suất gần tương đương ứng dụng native.

Flutter sử dụng ngôn ngữ lập trình Dart và được thiết kế dựa trên nguyên tắc "mọi thứ là widget". Toàn bộ giao diện người dùng được xây dựng từ các widget có thể là StatelessWidget (không thay đổi trạng thái) hoặc StatefulWidget (có trạng thái động). Điểm nổi bật của Flutter nằm ở cơ chế Hot Reload, cho phép lập trình viên xem trước thay đổi giao diện ngay lập tức mà không làm mất trạng thái ứng dụng, từ đó tăng đáng kể năng suất phát triển.

Về mặt hiệu suất, Flutter biên dịch mã Dart thành mã máy (native code) thông qua cơ chế Ahead-of-Time (AOT), giúp ứng dụng đạt tốc độ xử lý cao, gần tương đương với ứng dụng được viết bằng Kotlin hoặc Swift. Ngoài ra, Flutter sử dụng công cụ rendering riêng viết bằng C++, không phụ thuộc vào hệ thống giao diện của thiết bị, đảm bảo trải nghiệm người dùng nhất quán trên mọi nền tảng.

Hiện nay, Flutter hỗ trợ phát triển ứng dụng trên 6 nền tảng chính: iOS, Android, Web, Windows, macOS và Linux. Với hệ sinh thái widget phong phú, tài liệu đầy đủ và sự hỗ trợ mạnh mẽ từ Google, Flutter đã trở thành lựa chọn hàng đầu cho các dự án phát triển ứng dụng đa nền tảng.

```
// lib/main.dart

import 'package:flutter/material.dart';

void main() {

  runApp(const MyApp());

}

class MyApp extends StatelessWidget {

  const MyApp({super.key});

  @override

  Widget build(BuildContext context) {

    return MaterialApp(

      title: 'Flutter Demo',

      theme: ThemeData(primarySwatch: Colors.blue),

      home: const HomePage(),

    );

  }

}
```

```
}

class HomePage extends StatefulWidget {

  const HomePage({super.key});

  @override

  State<HomePage> createState() => _HomePageState();

}

class _HomePageState extends State<HomePage> {

  String _message = "Chào mừng đến với Flutter!";

  @override

  Widget build(BuildContext context) {

    return Scaffold(

      appBar: AppBar(title: const Text("Flutter Cơ Bản")),

      body: Center(

        child: Column(

          mainAxisAlignment: MainAxisAlignment.center,

          children: [

            Text(_message, style: const TextStyle(fontSize: 20)),

            ElevatedButton(

              onPressed: () {

                setState(() {

                  _message = "Bạn đã nhấn nút!";

                });

              },
```

```
    },  
    child: const Text("Nhấn tôi"),  
  ),  
], ), ),);}}
```

Lợi ích của Flutter:

- Phát triển đa nền tảng từ một codebase duy nhất Flutter cho phép xây dựng ứng dụng chạy mượt mà trên iOS, Android, Web, Windows, macOS và Linux chỉ với một bộ mã nguồn. Điều này giúp tiết kiệm 60–70% thời gian và công sức so với việc phát triển riêng cho từng nền tảng bằng Swift/Kotlin hay React Native.
- Hiệu suất gần tương đương ứng dụng native Nhờ cơ chế AOT compilation và Skia rendering engine, Flutter đạt 60 FPS ổn định trên hầu hết thiết bị, không có hiện tượng giật lag hay bridge như các framework cross-platform khác.
- Giao diện đẹp, nhất quán và tùy biến cao Hệ thống widget material design và Cupertino được xây dựng sẵn, kết hợp khả năng tùy chỉnh pixel-perfect, giúp tạo ra trải nghiệm người dùng đồng nhất và chuyên nghiệp trên mọi thiết bị.
- Hot Reload – Tăng năng suất phát triển vượt bậc Lập trình viên có thể xem trước thay đổi giao diện trong vòng dưới 1 giây, giúp quá trình debug và thiết kế UI trở nên nhanh chóng, trực quan, đặc biệt hữu ích trong giai đoạn phát triển giao diện phức tạp.
- Cộng đồng lớn và sự hỗ trợ từ Google Với hơn 150.000 stars trên GitHub, tài liệu chính thức đầy đủ, và hàng nghìn package trên pub.dev, Flutter đảm bảo tính bền vững lâu dài và dễ dàng tìm kiếm giải pháp khi gặp vấn đề.

1.5.3. Giới thiệu về Firebase

Firebase là một nền tảng phát triển ứng dụng Backend-as-a-Service (BaaS) do Google sở hữu, cung cấp bộ công cụ toàn diện giúp lập trình viên xây dựng ứng dụng

nhanh chóng mà không cần quản lý máy chủ. Firebase đặc biệt phù hợp với các dự án cần triển khai nhanh (MVP) và tích hợp tốt với Flutter.

Các dịch vụ chính của Firebase bao gồm:

- Firebase Authentication: Hỗ trợ đăng nhập bằng email/password, Google, Facebook, Apple, v.v.
- Cloud Firestore: Cơ sở dữ liệu NoSQL thời gian thực, đồng bộ dữ liệu tức thì giữa các thiết bị.
- Realtime Database: Cơ sở dữ liệu JSON với khả năng đồng bộ realtime.
- Cloud Storage: Lưu trữ file đa phương tiện (ảnh, video) với dung lượng lớn.
- Cloud Functions: Chạy mã backend không cần server (serverless).
- Firebase Analytics: Phân tích hành vi người dùng miễn phí.

Firebase cung cấp mức miễn phí (Spark Plan) đủ dùng cho các dự án nhỏ và vừa, cùng với hệ thống Firebase Security Rules giúp kiểm soát truy cập dữ liệu một cách linh hoạt và an toàn.

```
// services/product_service.dart

import 'package:cloud_firestore/cloud_firestore.dart';

class ProductService {

  final CollectionReference products =

    FirebaseFirestore.instance.collection('products');

  // Thêm sản phẩm vào collection

  Future<void> addProduct(String name, double price, int quantity) async {

    await products.add({

      'name': name,

      'price': price,
```

```
'quantity': quantity,  
  
  'created_at': FieldValue.serverTimestamp(),  
  
});  
  
}  
  
// Lấy danh sách sản phẩm realtime  
  
Stream<QuerySnapshot> getProducts() {  
  
  return products.orderBy('created_at', descending: true).snapshots();  
  
}  
  
}
```

Lợi ích của Firebase :

- Triển khai nhanh chóng mà không cần server Firebase là Backend-as-a-Service hoàn chỉnh: auth, database, storage, hosting → lập trình viên tập trung 100% vào frontend, phù hợp với đồ án có thời gian giới hạn.
- Đồng bộ dữ liệu thời gian thực Cloud Firestore và Realtime Database cho phép cập nhật dữ liệu tức thì trên tất cả thiết bị người dùng → lý tưởng cho các tính năng chat, thông báo, dashboard.
- Tích hợp sẵn với Flutter Các package như firebase_auth, cloud_firestore được Google duy trì, tài liệu rõ ràng, ví dụ phong phú → giảm thời gian setup xuống dưới 30 phút.

1.5.4. Giới thiệu về Supabase

Supabase là một nền tảng backend mã nguồn mở, được ra mắt vào năm 2020 với mục tiêu trở thành "Firebase thay thế sử dụng PostgreSQL". Khác với Firebase sử dụng cơ sở dữ liệu NoSQL, Supabase xây dựng trên PostgreSQL – một hệ quản trị cơ sở dữ liệu quan hệ mạnh mẽ, hỗ trợ đầy đủ các tính năng như ràng buộc toàn vẹn, transaction, trigger, và full-text search.

Supabase cung cấp các dịch vụ tương đương Firebase nhưng với kiến trúc mở và linh hoạt hơn:

- Authentication: Hỗ trợ email, OAuth (Google, GitHub, v.v.), xác thực bằng JWT.
- PostgreSQL Database: Quản lý dữ liệu quan hệ, hỗ trợ SQL chuẩn.
- Realtime: Đồng bộ dữ liệu thời gian thực qua WebSocket và cơ chế LISTEN/NOTIFY của PostgreSQL.
- Storage: Lưu trữ file tương thích S3.
- Edge Functions: Chạy hàm serverless bằng Deno.

Ưu điểm nổi bật của Supabase là mã nguồn mở, có thể tự host, và miễn phí tier cao hơn Firebase. Đặc biệt, với các ứng dụng cần mô hình dữ liệu phức tạp, quan hệ nhiều bảng, Supabase là lựa chọn vượt trội nhờ sức mạnh của PostgreSQL.

```
// services/auth_service.dart

import 'package:supabase_flutter/supabase_flutter.dart';

final supabase = Supabase.instance.client;

class AuthService {

  // Đăng ký

  Future<String?> signUp(String email, String password) async {

    try {

      final response = await supabase.auth.signUp(

        email: email,

        password: password,

      );

      return response.user != null ? "Đăng ký thành công!" : null;

    } catch (e) {

      // Xử lý lỗi đăng ký

    }

  }

}
```

```
} catch (e) {  
  
    return e.toString();  
  
}  
  
}  
  
// Đăng nhập  
  
Future<String?> signIn(String email, String password) async {  
  
    try {  
  
        final response = await supabase.auth.signInWithPassword(  
  
            email: email,  
  
            password: password,  
  
        );  
  
        return response.user != null ? "Đăng nhập thành công!" : null;  
  
    } catch (e) {  
  
        return "Sai email hoặc mật khẩu";  
  
    }  
  
}  
  
// Đăng xuất  
  
Future<void> signOut() async => await supabase.auth.signOut();  
  
// Lấy user hiện tại  
  
User? get currentUser => supabase.auth.currentUser;  
  
}
```

Lợi ích của Supabase:

- Cơ sở dữ liệu SQL mạnh mẽ (PostgreSQL) Hỗ trợ quan hệ nhiều-một, một-nhiều, transaction, trigger, index → phù hợp với các hệ thống có mô hình dữ liệu phức tạp như quản lý sinh viên, đặt lịch, bán hàng.
- Mã nguồn mở & có thể tự host Toàn bộ hệ thống Supabase có thể cài đặt trên VPS hoặc Docker → không phụ thuộc nhà cung cấp, dễ chuyển đổi khi cần mở rộng hoặc bảo mật dữ liệu.
- Realtime mà không cần viết thêm code Chỉ cần bật Realtime trên bảng → dữ liệu tự động đẩy về client qua WebSocket → tương tự Firebase nhưng dùng SQL.
- Tương lai phát triển tốt Được đầu tư bởi Y Combinator, tăng trưởng nhanh, cộng đồng tích cực → đảm bảo tính bền vững cho đề án và sản phẩm sau này.

CHƯƠNG 2: PHÂN TÍCH, THIẾT KẾ HỆ THỐNG

2.1. Xác định yêu cầu

Tiến hành nghiên cứu sâu các quy trình và luồng hoạt động của một hệ thống ứng dụng đặt vé sự kiện đa nền tảng (iOS, Android, Web), nhằm đảm bảo hiểu rõ cơ cấu tổ chức, cách thức vận hành và các điểm tương tác giữa người dùng, hệ thống và đơn vị tổ chức sự kiện. Nghiên cứu các nhu cầu thực tế của người dùng khi tìm kiếm, xem thông tin và đặt vé tham gia sự kiện trực tuyến, từ đó phân tích và xây dựng các chức năng cốt lõi của ứng dụng. Các chức năng được phát triển dựa trên yêu cầu thực tiễn của người dùng cuối, ban tổ chức và quản trị viên hệ thống.

Chia sẻ thông tin về sự kiện là một trong những chức năng quan trọng nhất, giúp người dùng dễ dàng nhận diện, so sánh và lựa chọn sự kiện phù hợp. Mỗi sự kiện cần được trình bày đầy đủ, rõ ràng với các thông tin như: tên sự kiện đầy đủ, ngắn gọn và phản ánh đúng nội dung; chủ đề hoặc loại hình (hòa nhạc, hội thảo, triển lãm, thể thao, workshop...); thời gian diễn ra chính xác kèm múi giờ; địa điểm cụ thể với địa chỉ, bản đồ tích hợp và hướng dẫn di chuyển; mô tả chi tiết về chương trình, diễn giả, lịch trình và đối tượng tham gia; hình ảnh minh họa, video trailer hoặc tư liệu từ các sự kiện trước; thông tin về đơn vị tổ chức (tên, website, uy tín); và đặc biệt là chính sách vé với các loại vé (VIP, Standard, Early Bird, Group), giá từng loại, số lượng còn lại và thời hạn bán vé.

Chia sẻ thông tin về hệ thống tổ chức sự kiện và quản lý vé đóng vai trò nền tảng để xây dựng một hệ thống đặt vé đáng tin cậy, bảo mật và hiệu quả cao, phục vụ tốt nhu cầu tham gia sự kiện của người dùng và tối ưu hóa quy trình vận hành cho ban tổ chức. Ứng dụng được phát triển đa nền tảng (iOS, Android, Web) bằng Flutter, đảm bảo giao diện thống nhất, trải nghiệm mượt mà trên mọi thiết bị. Phía backend sử dụng Firebase với các dịch vụ như Authentication, Firestore, Cloud Functions, Storage và Messaging để quản lý người dùng, dữ liệu sự kiện, đơn hàng, thanh toán và thông báo. Hệ thống hỗ trợ tạo vé điện tử QR code duy nhất, chống giả mạo, tự động gửi qua email và lưu trữ trong ứng dụng. Tích hợp nhiều cổng thanh toán phổ biến như Momo, ZaloPay, VNPay, thẻ tín dụng và chuyển khoản ngân hàng. Dữ liệu người dùng được mã hóa, tuân thủ các

quy định pháp lý về bảo mật thông tin và an ninh mạng. Ngoài ra, hệ thống cung cấp giao diện quản trị riêng cho Admin để tạo sự kiện, quản lý vé, theo dõi doanh thu .

Đặt vé sự kiện trực tuyến là tính năng cốt lõi, mang lại trải nghiệm tiện lợi và hiện đại cho người dùng, đồng thời giúp ban tổ chức quản lý hiệu quả nguồn lực và tránh tình trạng quá tải. Quy trình đặt vé được thiết kế đơn giản, rõ ràng: người dùng truy cập ứng dụng, tìm kiếm sự kiện theo từ khóa, danh mục, địa điểm hoặc ngày diễn ra; xem chi tiết sự kiện và chọn loại vé phù hợp; đăng nhập bằng email, chọn số lượng vé, kiểm tra tổng tiền; thực hiện thanh toán qua cổng tích hợp; nhận ngay vé điện tử QR code trong mục “Vé của tôi” kèm email xác nhận . Toàn bộ quá trình diễn ra nhanh chóng, minh bạch, giúp người dùng yên tâm và tránh tình trạng chông chéo lịch hoặc đặt vé giả.

Hệ thống mở rộng các chức năng liên quan như tìm kiếm sự kiện, quản lý đơn hàng (xem lại, hủy vé nếu được phép), và trong tương lai có thể bổ sung check-in tại sự kiện bằng cách quét mã QR.

Về yêu cầu phi chức năng, hệ thống đảm bảo: thời gian tải danh sách sự kiện dưới 2 giây, xử lý thanh toán dưới 5 giây; khả năng mở rộng hỗ trợ hàng chục nghìn người dùng đồng thời; tỷ lệ lỗi giao dịch dưới 0,1%; bảo mật cao với mã hóa SSL, xác thực an toàn, không lưu thông tin thẻ; giao diện thân thiện, responsive, tuân thủ Material Design.

2.2. Phân tích yêu cầu

2.2.1 Chức năng đăng nhập

a. Khách hàng

- Khi đăng nhập vào ứng dụng, sẽ hiện thông tin cá nhân của mình (họ tên, email, số điện thoại, lịch sử đặt vé) và hiển thị đầy đủ các danh mục dành cho khách hàng như: tìm kiếm sự kiện, đặt vé, xem vé đã mua và quản lý hồ sơ.

b. Nhân viên

- Khi đăng nhập, sẽ hiển thị giao diện check-in sự kiện với danh sách sự kiện đang diễn ra và công cụ quét mã QR để kiểm tra vé tại cổng.

c. Admin

- Khi đăng nhập, sẽ hiển thị toàn bộ chức năng quản trị, có quyền thêm, sửa, xóa và cập nhật thông tin sự kiện, khách hàng, đơn hàng và người dùng hệ thống.
- Đồng thời quản lý toàn bộ lịch bán vé và check-in của các sự kiện.

2.2.2 Chức năng đặt vé sự kiện

- Người dùng truy cập ứng dụng, sử dụng bộ lọc để tìm kiếm sự kiện theo tên, loại hình, địa điểm, ngày diễn ra hoặc mức giá.
- Chọn sự kiện : Xem chi tiết (mô tả, hình ảnh, lịch trình, loại vé, số lượng còn lại).
- Chọn loại vé và số lượng : Hệ thống tự động tính tổng tiền.
- Chọn phương thức thanh toán : Xác nhận đơn hàng.
- Sau khi thanh toán thành công, hệ thống tự động:
 - Tạo vé điện tử QR code duy nhất cho từng vé.
 - Lưu vé vào mục “Vé của tôi” trong ứng dụng.
 - Gửi email xác nhận và mã QR.

2.2.3 Chức năng thêm , sửa, xóa thông tin sự kiện

- Chức năng này cho phép Admin thêm mới thông tin chi tiết về sự kiện vào hệ thống.
- Thông tin bao gồm: tên sự kiện, loại hình, thời gian, địa điểm, mô tả, hình ảnh/video, chính sách vé, giá vé, số lượng vé từng loại, thời gian mở/đóng bán vé.
- Dữ liệu sự kiện sẽ được hiển thị công khai cho người dùng trong quá trình tìm kiếm và đặt vé, giúp họ lựa chọn sự kiện phù hợp.

2.2.4 Chức năng tìm kiếm sự kiện

- Chức năng này cho phép tìm kiếm sự kiện theo từ khóa, loại hình, ngày diễn ra, địa điểm, mức giá.

- Kết quả hiển thị rõ ràng với ảnh bìa, tên sự kiện, thời gian, địa điểm và giá vé thấp nhất, giúp người dùng dễ dàng chọn lựa sự kiện phù hợp trước khi đặt vé.

2.2.5 Chức năng quản lý order

- Admin vào mục Quản lý order để xem danh sách order (mã đơn, sự kiện, khách hàng, số lượng, tổng tiền, trạng thái).
- Có thể xem chi tiết order (người tham dự, loại vé, mã QR, hóa đơn).
- Hỗ trợ xóa order (nếu chưa thanh toán hoặc hủy hợp lệ), cập nhật lại số lượng vé .

2.2.6 Chức năng check-in sự kiện

- Nhân viên: có thể xem danh sách vé đã đặt theo sự kiện đang diễn ra, thực hiện quét mã QR để kiểm tra, xác nhận khách vào cổng và cập nhật trạng thái “Đã check-in” theo thời gian thực.
- Hệ thống ngăn chặn việc sử dụng lại vé và hỗ trợ check-in ngoại tuyến (đồng bộ sau khi có mạng).

2.2.7 Chức năng quản lý khách hàng

- Admin vào mục Quản lý khách hàng để xem danh sách khách hàng (họ tên, email, số điện thoại, ngày đăng ký, tổng vé đã mua).
- Có thể xóa khách hàng (khóa tài khoản vĩnh viễn nếu vi phạm chính sách hoặc theo yêu cầu).

2.2.8 Chức năng quản lý nhân viên

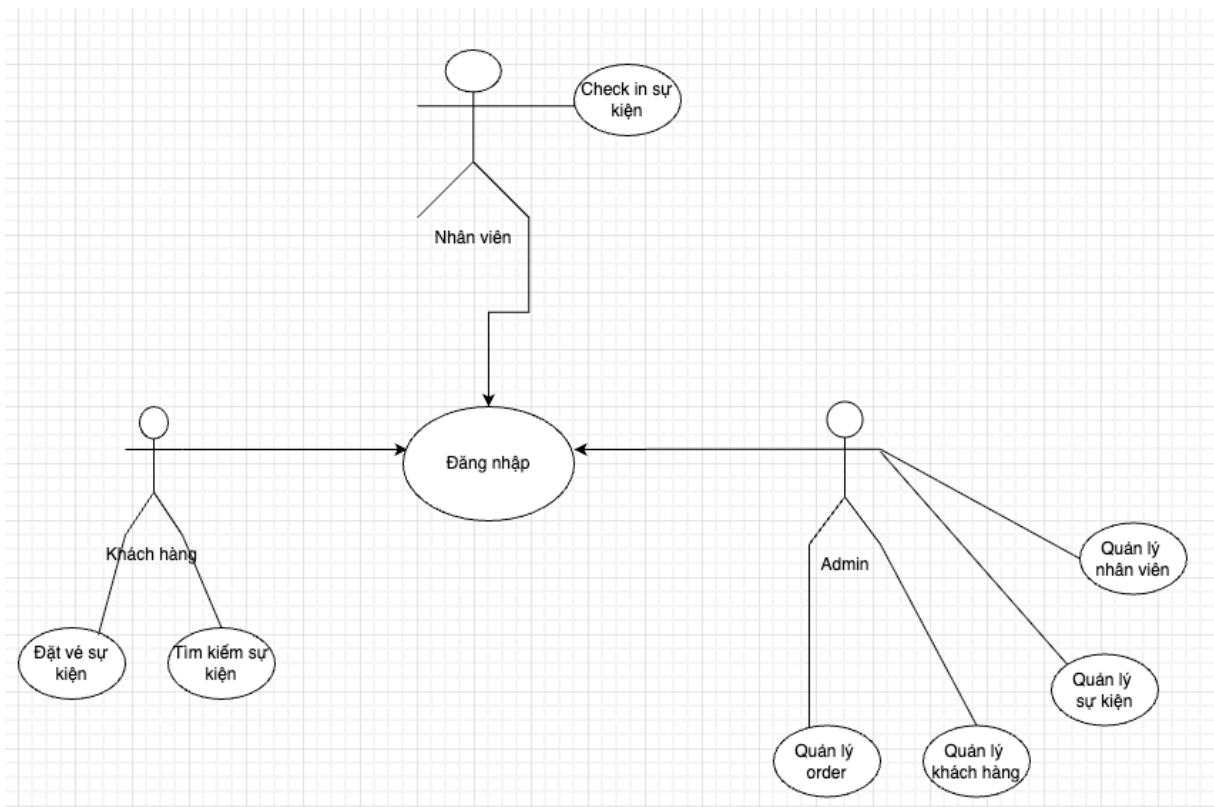
Admin vào mục Quản lý nhân viên để thực hiện các thao tác quản trị bao gồm:

- Xem danh sách nhân viên (họ tên, email, số điện thoại, ngày tạo tài khoản, vai trò).
- Thêm nhân viên mới.

- Cập nhật thông tin nhân viên (sửa).
- Xóa nhân viên (khóa tài khoản nếu cần thiết).

2.3. Mô hình hóa yêu cầu

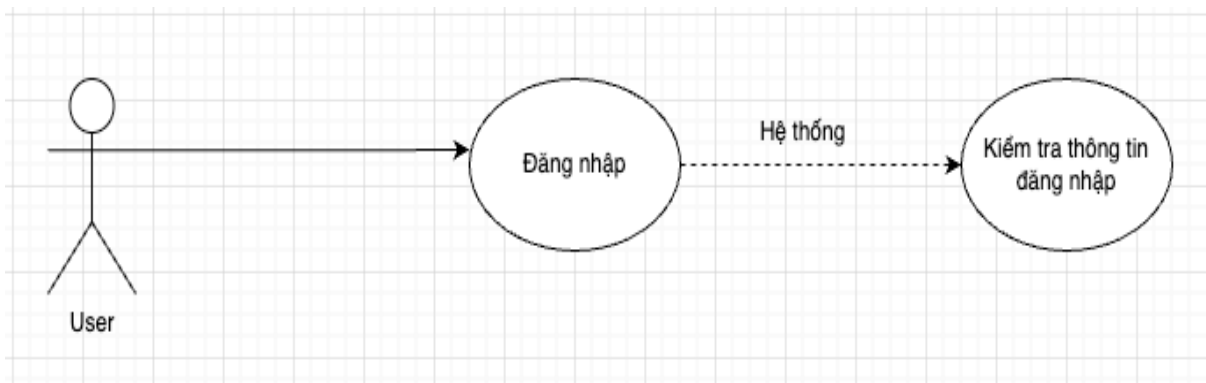
- Usecase tổng quát



Hình 2.1 Biểu đồ Use Case tổng quát

2.3.1 Chức năng đăng nhập

- Biểu đồ Use Case đăng nhập



Hình 2.2 Biểu đồ Use Case đăng nhập

- Mô tả Use Case đăng nhập

Bảng 2.1 Mô tả Use Case đăng nhập

Tên Use Case	Đăng nhập
Tác nhân	Admin, nhân viên và người dùng
Điều kiện đầu vào	Admin, nhân viên và người dùng chưa đăng nhập
Kết quả đầu ra	Admin, nhân viên và người dùng đăng nhập thành công
Mô tả	Use Case cho phép Admin, nhân viên và người dùng đăng nhập vào hệ thống.

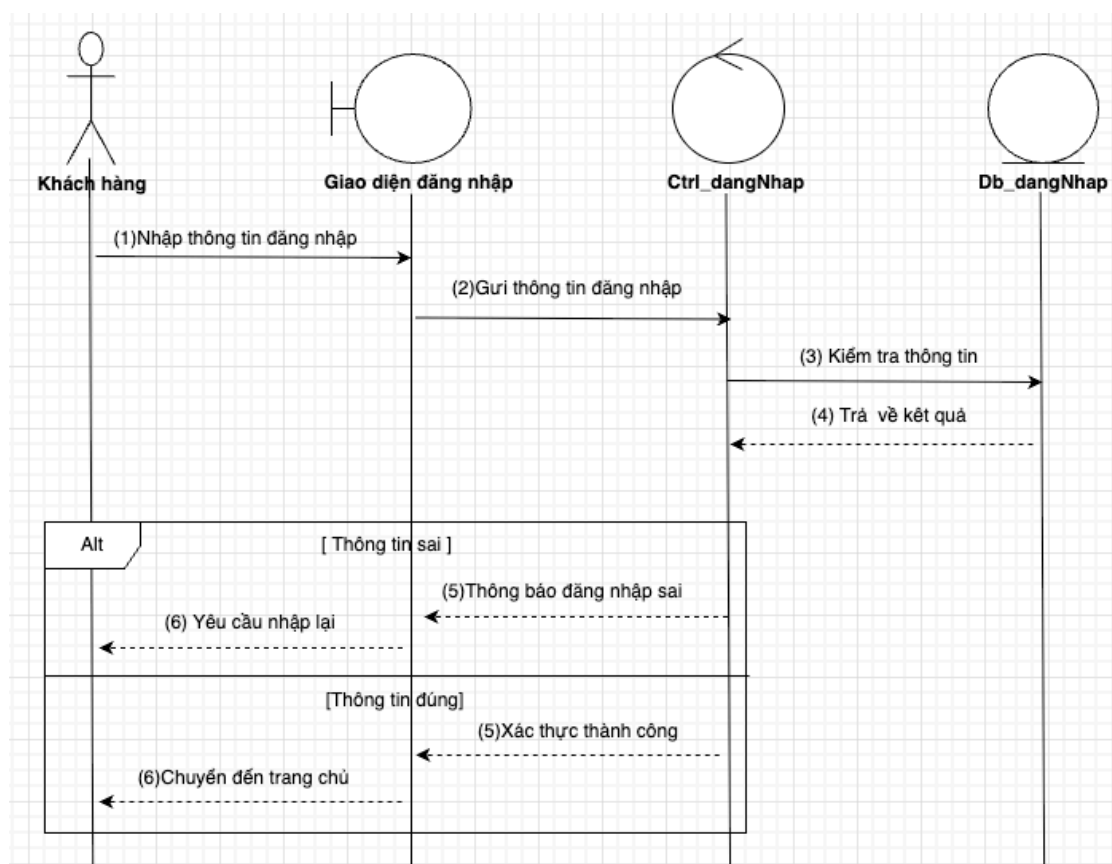
- Dòng chính Use Case đăng nhập

Bảng 2.2 Dòng chính Use Case đăng nhập

STT	Thực hiện	Hành động
1	Tác nhân	Admin, nhân viên và người dùng truy cập vào ứng dụng.
2	Hệ thống	Kiểm tra email và mật khẩu có đúng không
3	Tác nhân	Nhập email, mật khẩu

4	Hệ thống	Xác thực thành công và chuyển hướng đến giao diện tương ứng với vai trò (Admin, nhân viên và người dùng)
---	----------	--

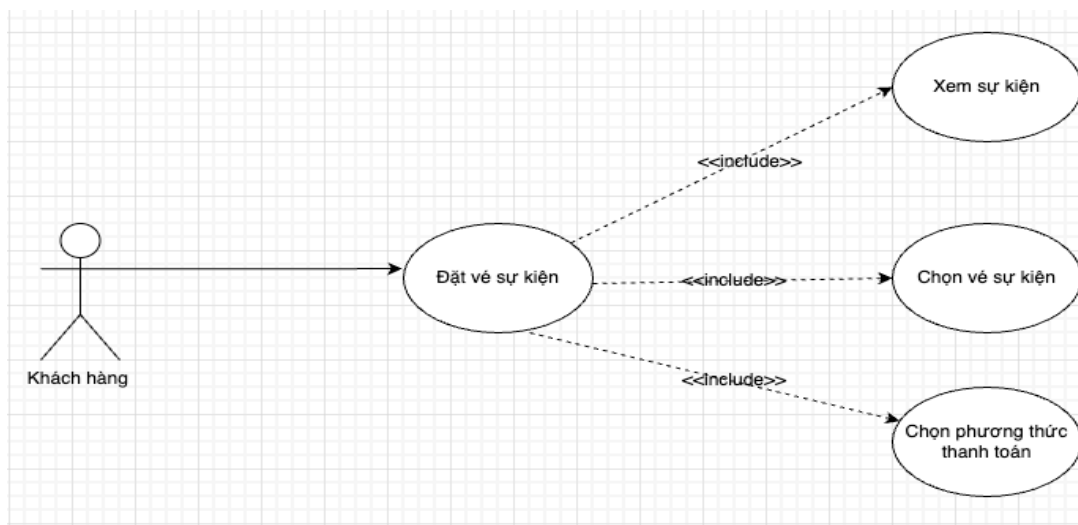
- Biểu đồ tuần tự đăng nhập



Hình 2.3 Biểu đồ tuần tự đăng nhập

2.3.2 Chức năng đặt vé sự kiện

- Biểu đồ Use Case đặt vé sự kiện



Hình 2.4 Biểu đồ Use Case đặt vé sự kiện

- Mô tả Use Case đặt vé sự kiện

Bảng 2.3 Mô tả Use Case đặt vé sự kiện

Tên Use Case	Đặt vé sự kiện
Tác nhân	Người dùng
Điều kiện đầu vào	Khách hàng đã truy cập vào ứng dụng
Kết quả đầu ra	Đặt vé thành công, nhận được vé điện tử
Mô tả	Use Case cho phép khách hàng thực hiện toàn bộ quy trình đặt vé, bao gồm xem sự kiện, chọn sự kiện và chọn phương thức thanh toán.

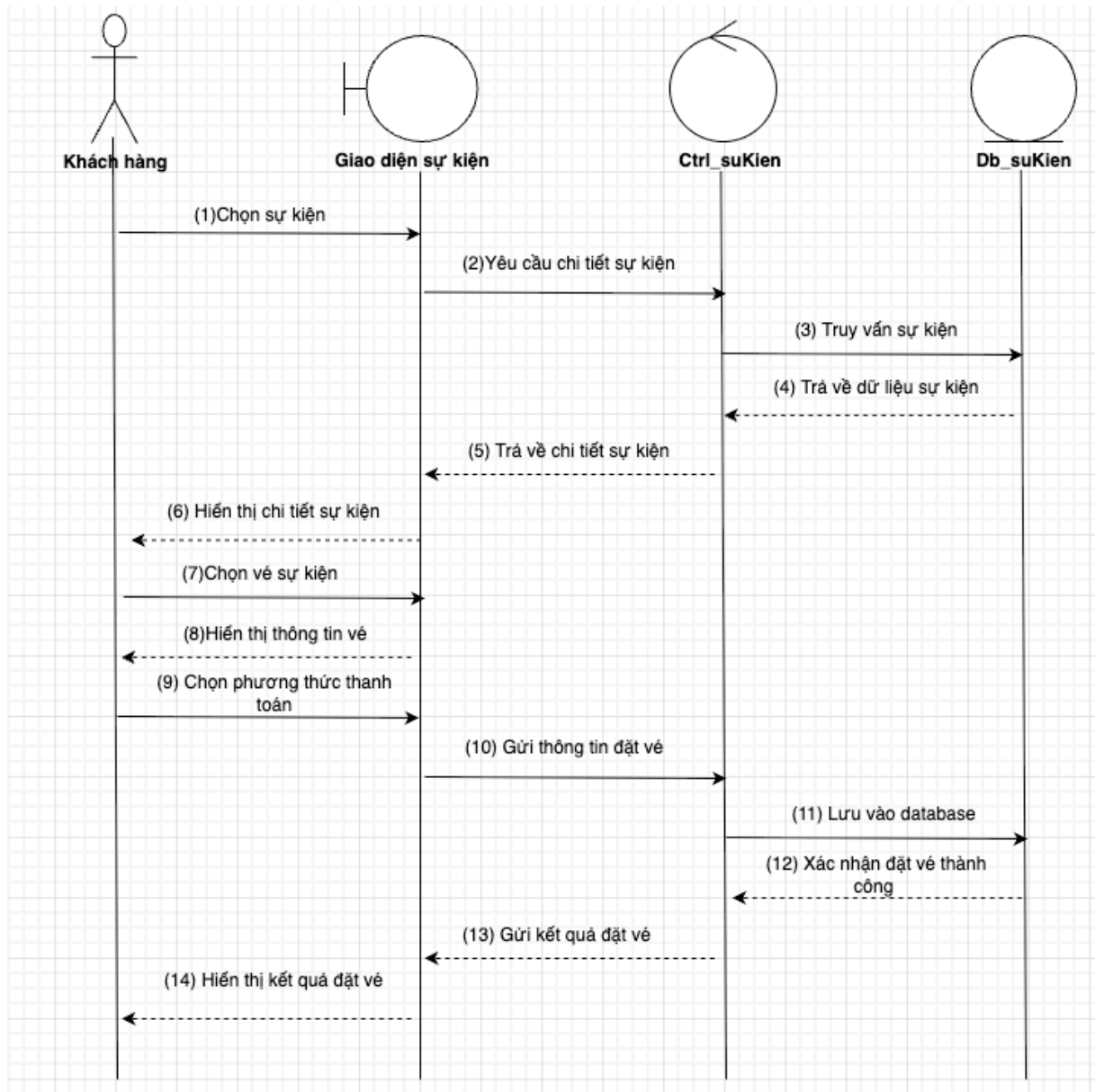
- Dòng chính Use Case đặt vé sự kiện

Bảng 2.4 Dòng chính Use Case đặt vé sự kiện

STT	Thực hiện	Hành động
1	Tác nhân	Khách hàng bắt đầu đặt vé sự kiện.
2	Hệ thống	Hiển thị danh sách sự kiện đang mở bán.

3	Tác nhân	Xem chi tiết sự kiện (tên, mô tả, thời gian, địa điểm, hình ảnh, loại vé, giá, số lượng còn lại).
4	Tác nhân	Chọn sự kiện cần tham gia.
5	Hệ thống	Hiển thị form đặt vé với các loại vé khả dụng.
6	Tác nhân	Chọn loại vé, số lượng vé, nhập thông tin người tham dự.
7	Tác nhân	Chọn phương thức thanh toán.
8	Hệ thống	Chuyển hướng đến cổng thanh toán và xác nhận giao dịch thành công.
9	Hệ thống	• Tạo vé điện tử với mã QR duy nhất cho từng người tham dự. • Lưu đơn hàng vào cơ sở dữ liệu. • Gửi email xác nhận kèm vé (PDF + mã QR). • Cập nhật số lượng vé còn lại theo thời gian thực.

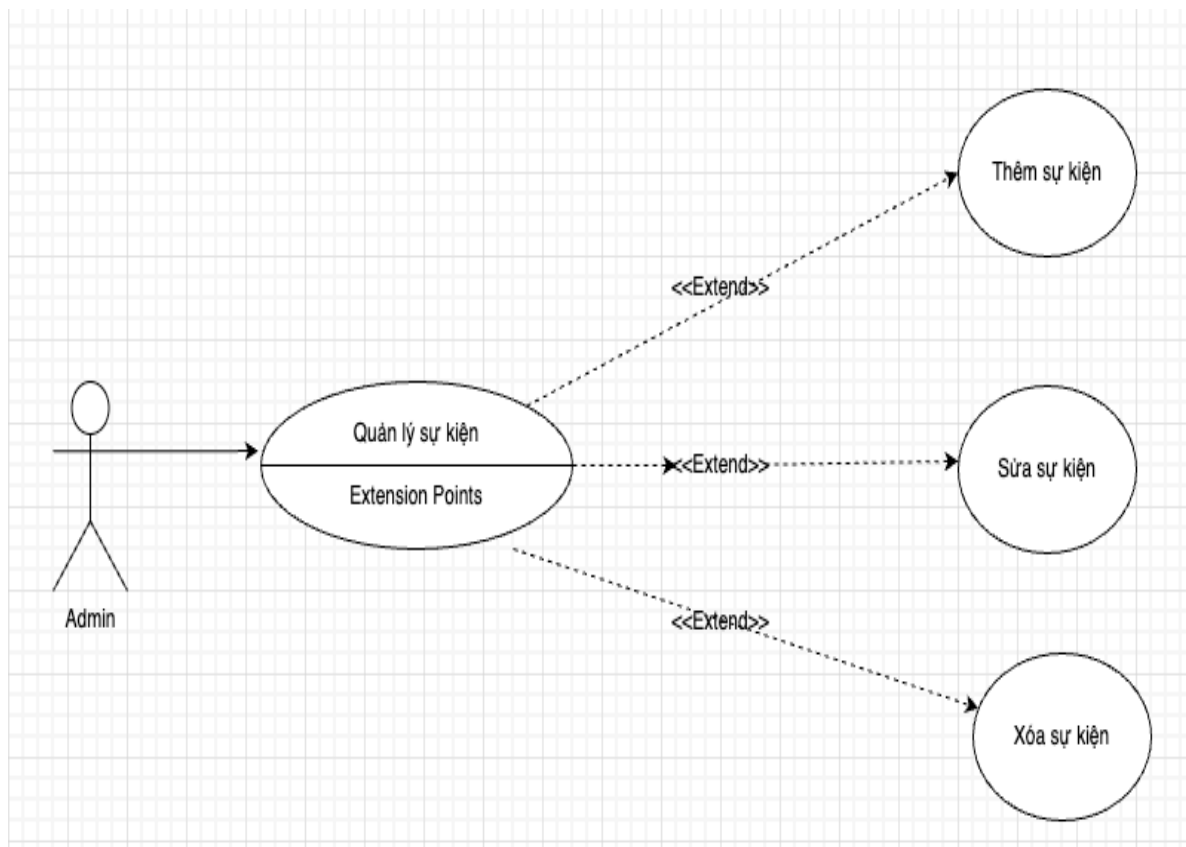
- Biểu đồ tuần tự đặt vé sự kiện



Hình 2.5 Biểu đồ tuần tự đặt vé sự kiện

2.3.3 Chức năng quản lý sự kiện

- Biểu đồ Use Case quản lý sự kiện



Hình 2.6 Biểu đồ Use Case quản lý sự kiện

- Mô tả Use Case quản lý sự kiện

Bảng 2.5 Mô tả Use Case quản lý sự kiện

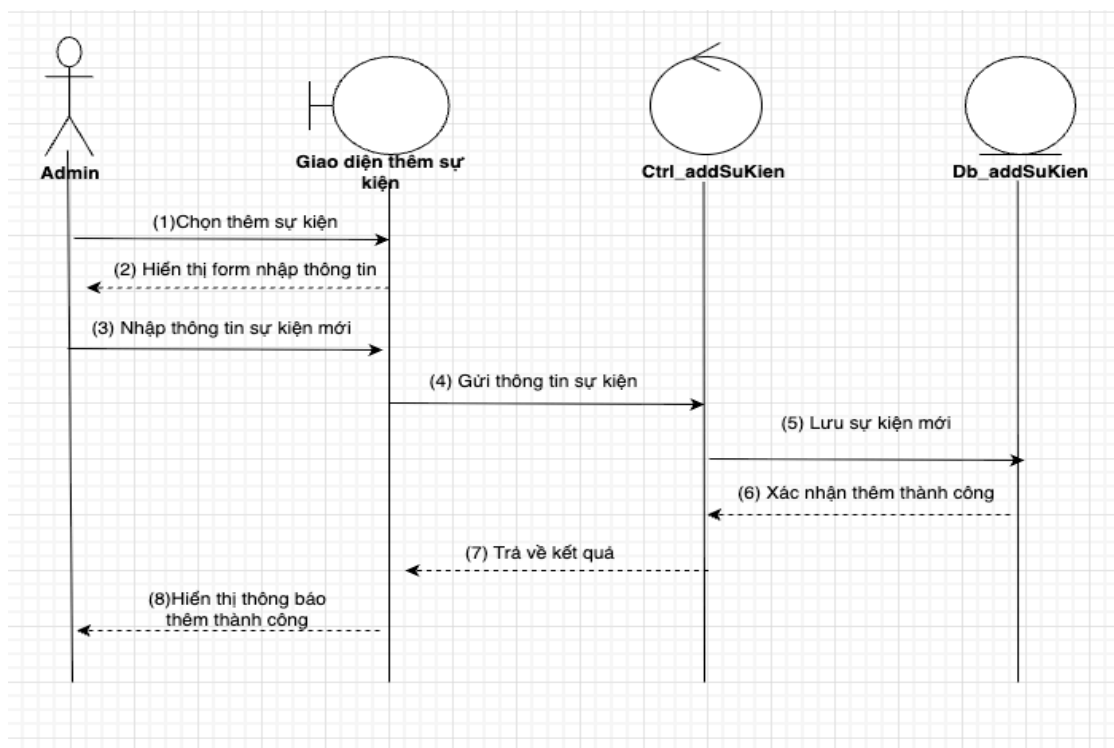
Tên Use Case	Quản lý sự kiện
Tác nhân	Admin
Điều kiện đầu vào	Admin đã đăng nhập vào hệ thống
Kết quả đầu ra	Sự kiện được thêm, sửa hoặc xóa thành công
Mô tả	Use Case cho phép Admin thực hiện các thao tác quản lý toàn diện đối với sự kiện, bao gồm thêm mới, sửa đổi và xóa sự kiện

- Dòng chính Use Case quản lý sự kiện

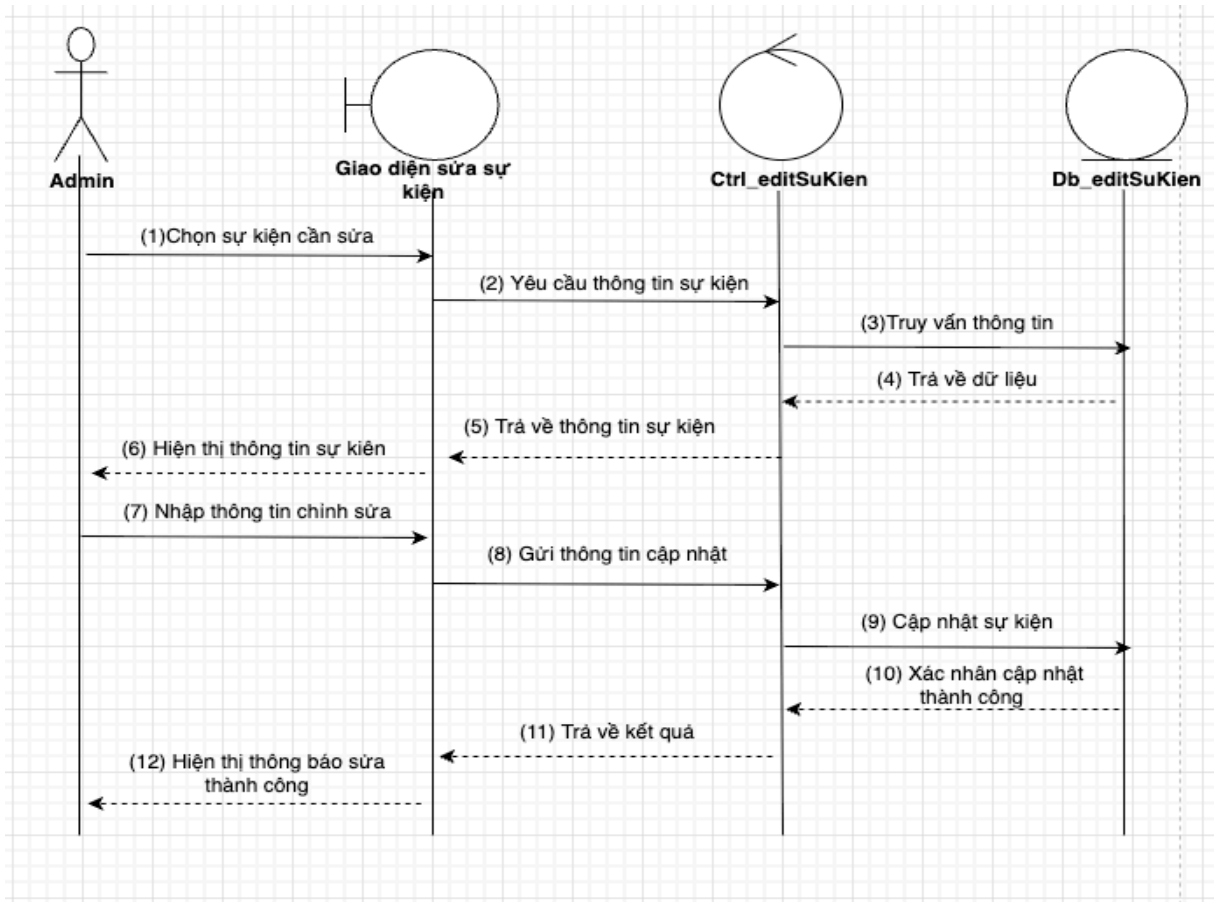
Bảng 2.6 Dòng chính Use Case quản lý sự kiện

STT	Thực hiện	Hành động
1	Tác nhân	Chọn chức năng Quản lý sự kiện
2	Hệ thống	Hiển thị giao diện danh sách sự kiện
3	Tác nhân	Chọn thao tác (Thêm / Sửa / Xóa sự kiện)
4	Hệ thống	Hiển thị form tương ứng với thao tác đã chọn
5	Tác nhân	Nhập hoặc chỉnh sửa thông tin sự kiện / xác nhận thao tác xóa.
6	Hệ thống	Kiểm tra dữ liệu và cập nhật vào cơ sở dữ liệu.
7	Hệ thống	Hiển thị thông báo thao tác thành công.
8	Hệ thống	Cập nhật lại danh sách sự kiện.

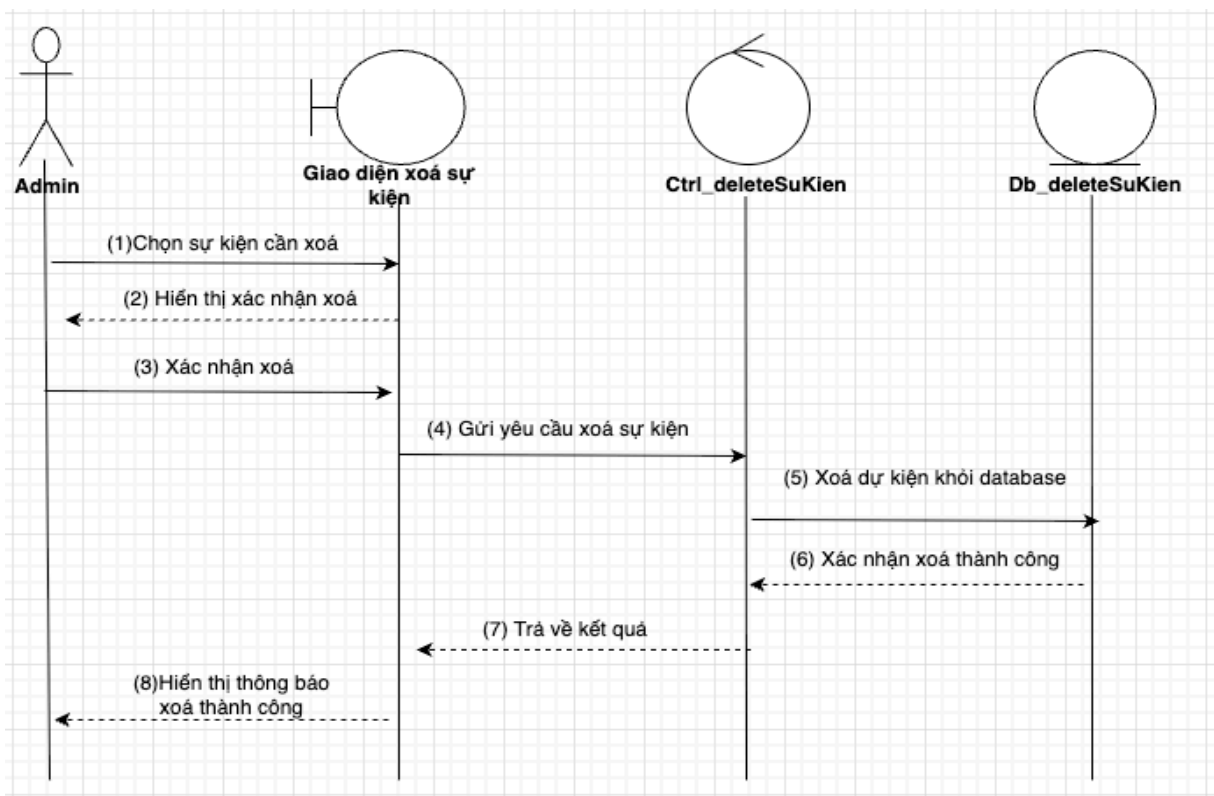
- Biểu đồ tuần tự quản lý sự kiện



Hình 2.7 Biểu đồ tuần tự thêm sự kiện



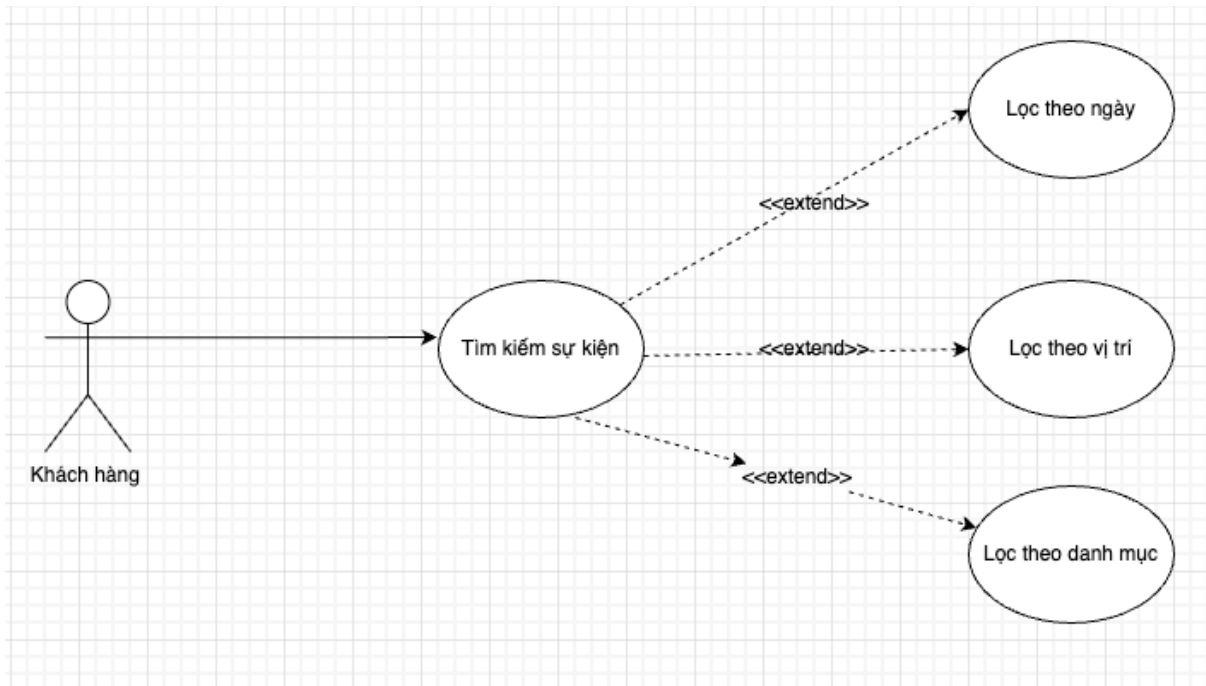
Hình 2.8 Biểu đồ tuần tự sửa sự kiện



Hình 2.9 Biểu đồ tuần tự xóa sự kiện

2.3.4 Chức năng tìm kiếm sự kiện

- Biểu đồ Use Case tìm kiếm sự kiện



Hình 2.10 Biểu đồ Use Case tìm kiếm sự kiện

- Mô tả Use Case tìm kiếm sự kiện

Bảng 2.7 Mô tả Use Case tìm kiếm sự kiện

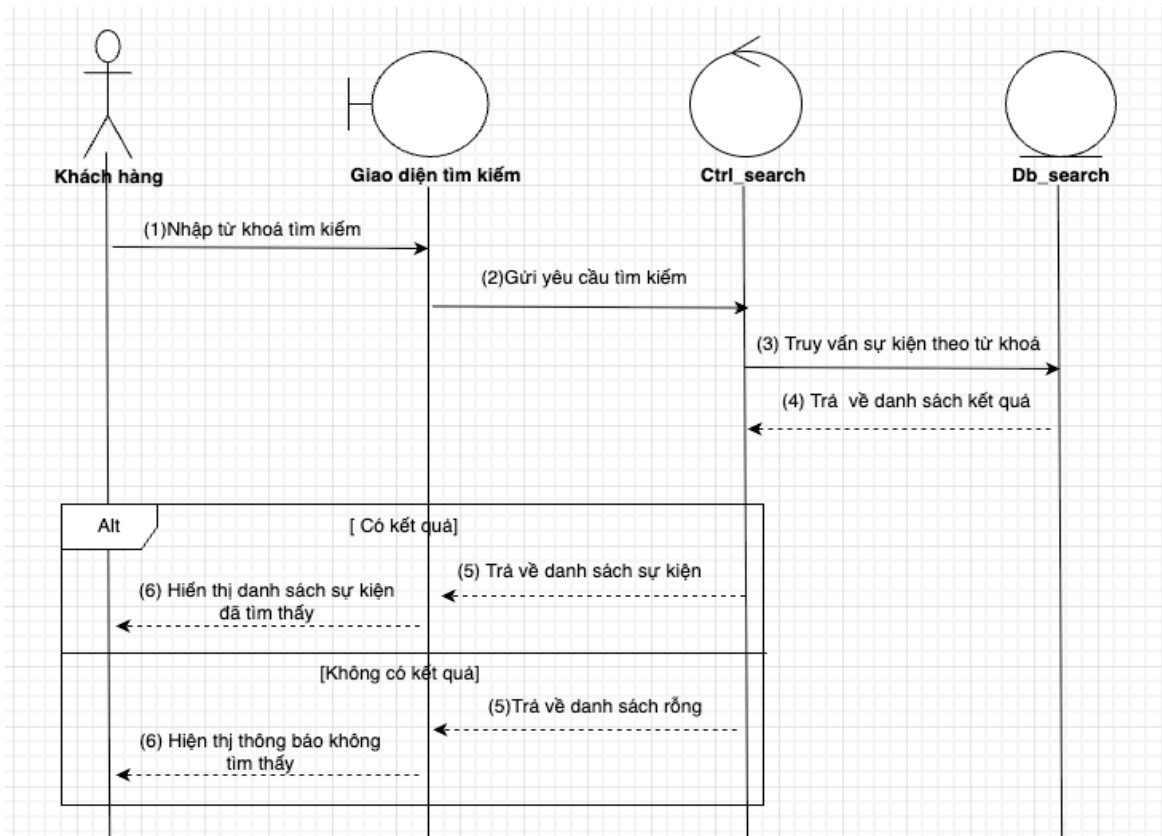
Tên Use Case	Tìm kiếm sự kiện
Tác nhân	Khách hàng
Điều kiện đầu vào	Khách hàng đăng nhập và truy cập vào chức năng tìm kiếm sự kiện
Kết quả đầu ra	Danh sách sự kiện được tìm thấy và có thể lọc theo ngày, vị trí, danh mục.
Mô tả	Use Case cho phép khách hàng tìm kiếm sự kiện và thực hiện lọc nâng cao theo ngày, vị trí hoặc danh mục.

- Dòng chính Use Case tìm kiếm sự kiện

Bảng 2.8 Dòng chính Use Case tìm kiếm sự kiện

STT	Thực hiện	Hành động
1	Tác nhân	Chọn chức năng Tìm kiếm sự kiện
2	Hệ thống	Hiển thị giao diện tìm kiếm sự kiện
3	Tác nhân	Nhập từ khóa tìm kiếm (tên sự kiện)
4	Hệ thống	Hiển thị danh sách sự kiện phù hợp với từ khóa
5	Tác nhân	Chọn tùy chọn lọc (ngày / vị trí / danh mục)
6	Hệ thống	Áp dụng bộ lọc và hiển thị danh sách sự kiện sau khi lọc

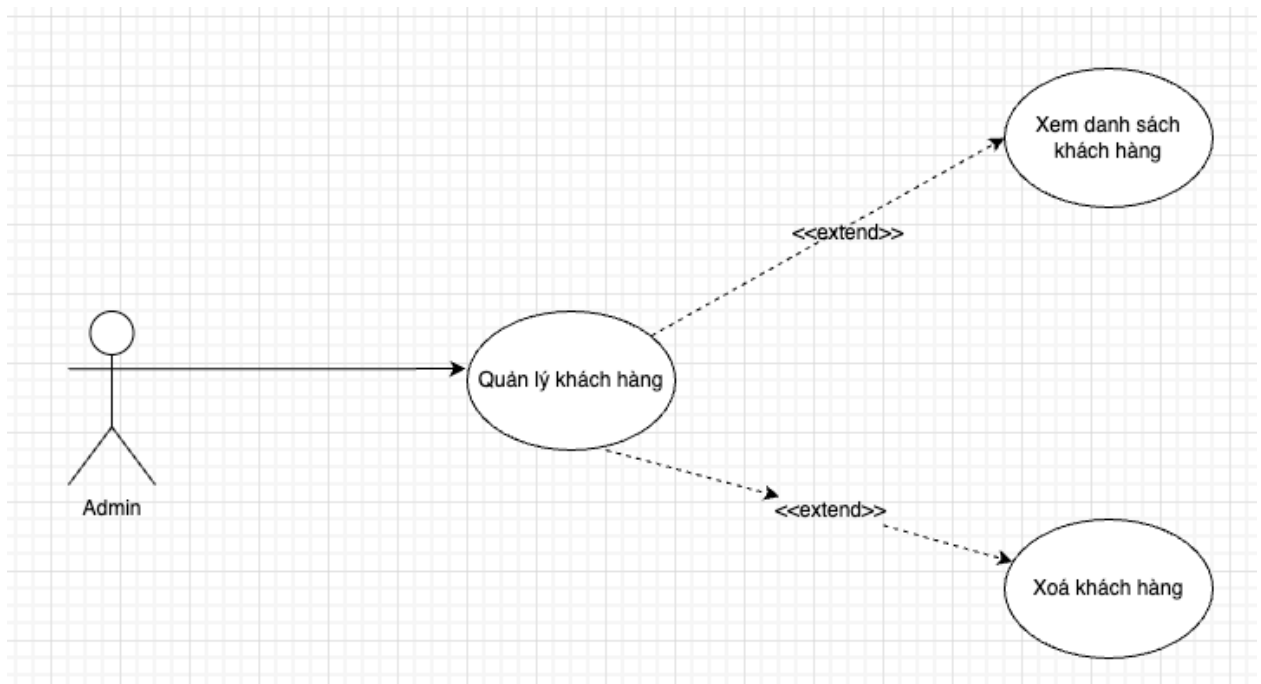
- Biểu đồ tuần tự tìm kiếm sự kiện



Hình 2.11 Biểu đồ tuần tự tìm kiếm sự kiện

2.3.5 Chức năng quản lý khách hàng

- Biểu đồ Use Case quản lý khách hàng



Hình 2.12 Biểu đồ Use Case quản lý khách hàng

- Mô tả Use Case quản lý khách hàng

Bảng 2.9 Mô tả Use Case quản lý khách hàng

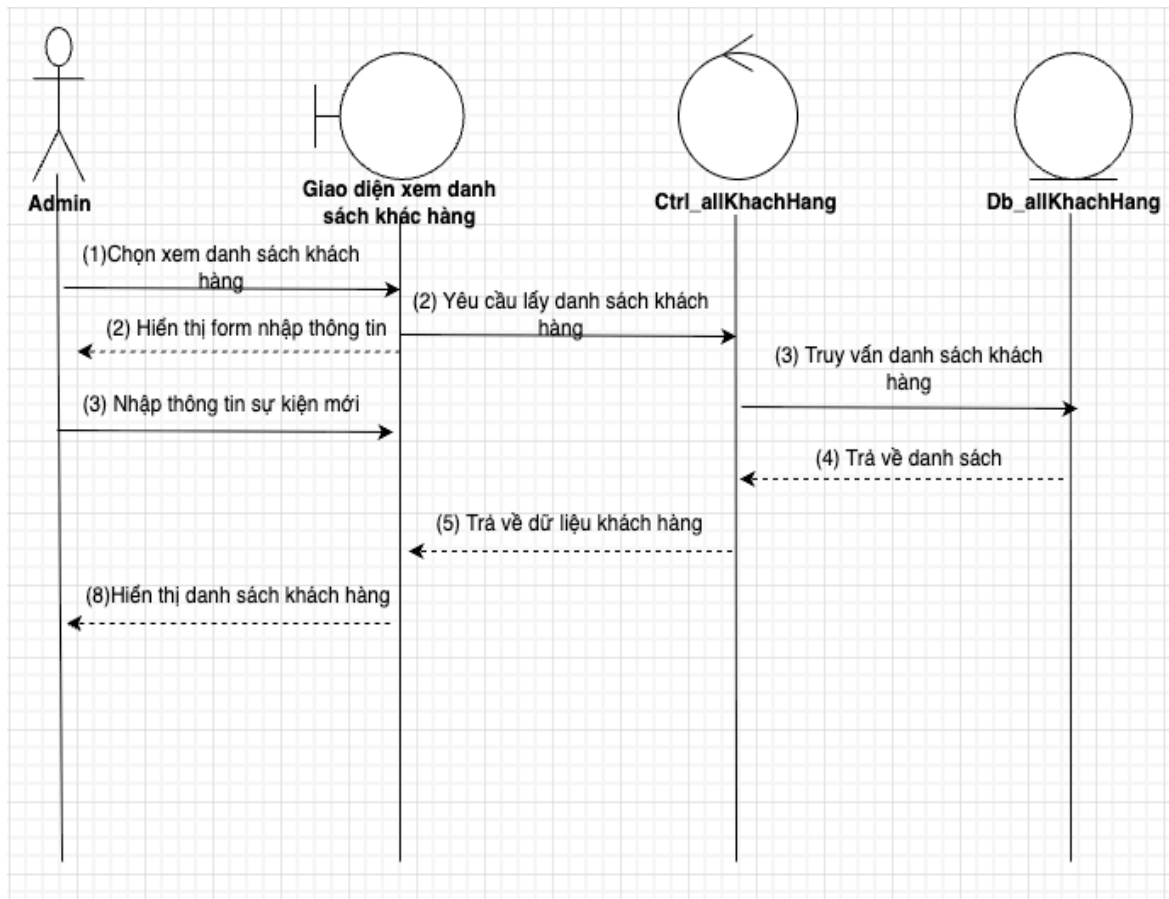
Tên Use Case	Quản lý khách hàng
Tác nhân	Admin
Điều kiện đầu vào	Admin đăng nhập và truy cập chức năng quản lý khách hàng
Kết quả đầu ra	Admin được xem danh sách khách hàng, hoặc xóa thành công
Mô tả	Use Case cho phép Admin xem danh sách khách hàng, xóa thông tin khách hàng

- Dòng chính Use Case quản lý khách hàng

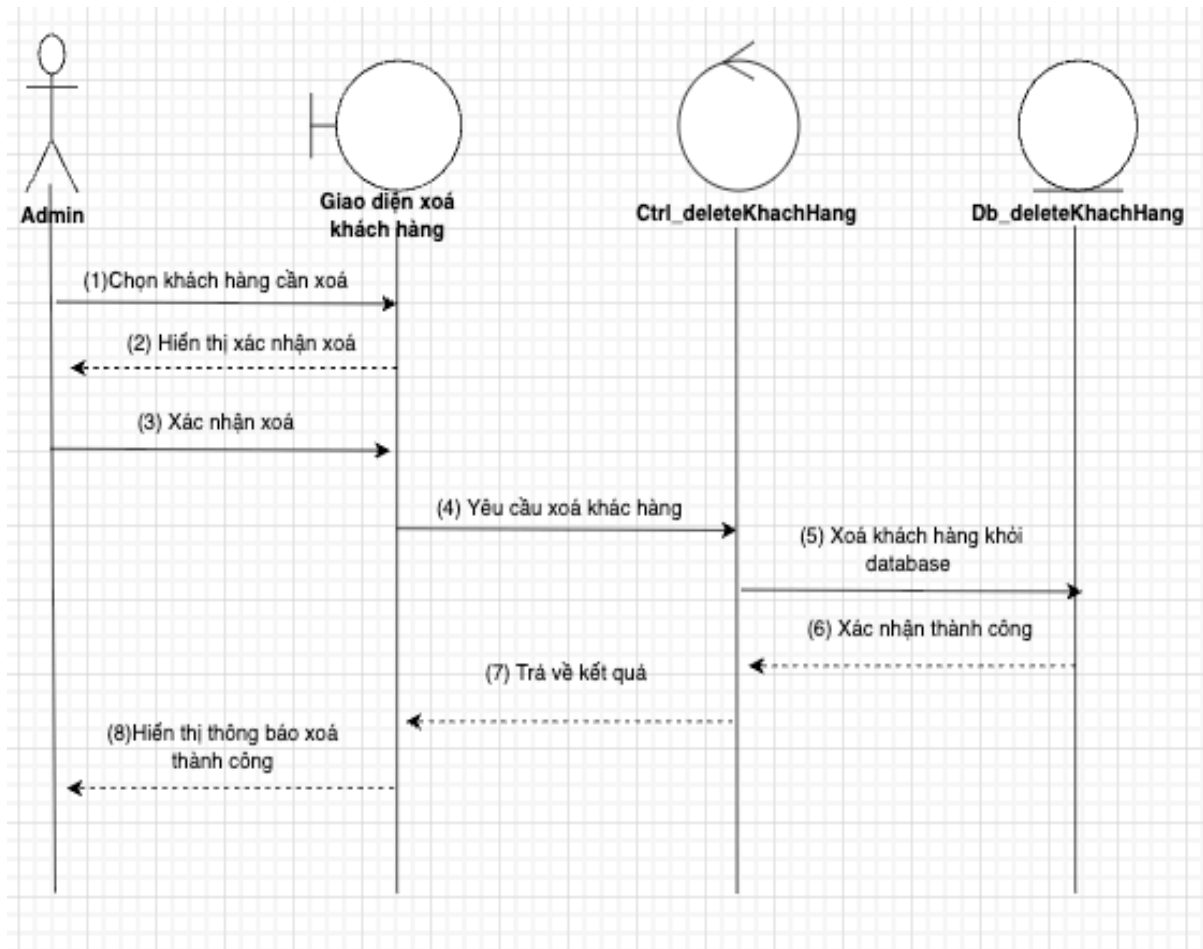
Bảng 2.10 Dòng chính Use Case quản lý khách hàng

STT	Thực hiện	Hành động
1	Tác nhân	Chọn chức năng Quản lý khách hàng
2	Hệ thống	Hiển thị giao diện quản lý khách hàng
3	Tác nhân	Chọn thao tác (xem danh sách khách hàng/xóa khách hàng)
4	Hệ thống	Thực hiện chức năng tương ứng
5	Hệ thống	Cập nhật lại danh sách khách hàng.

- Biểu đồ tuần tự quản lý khách hàng



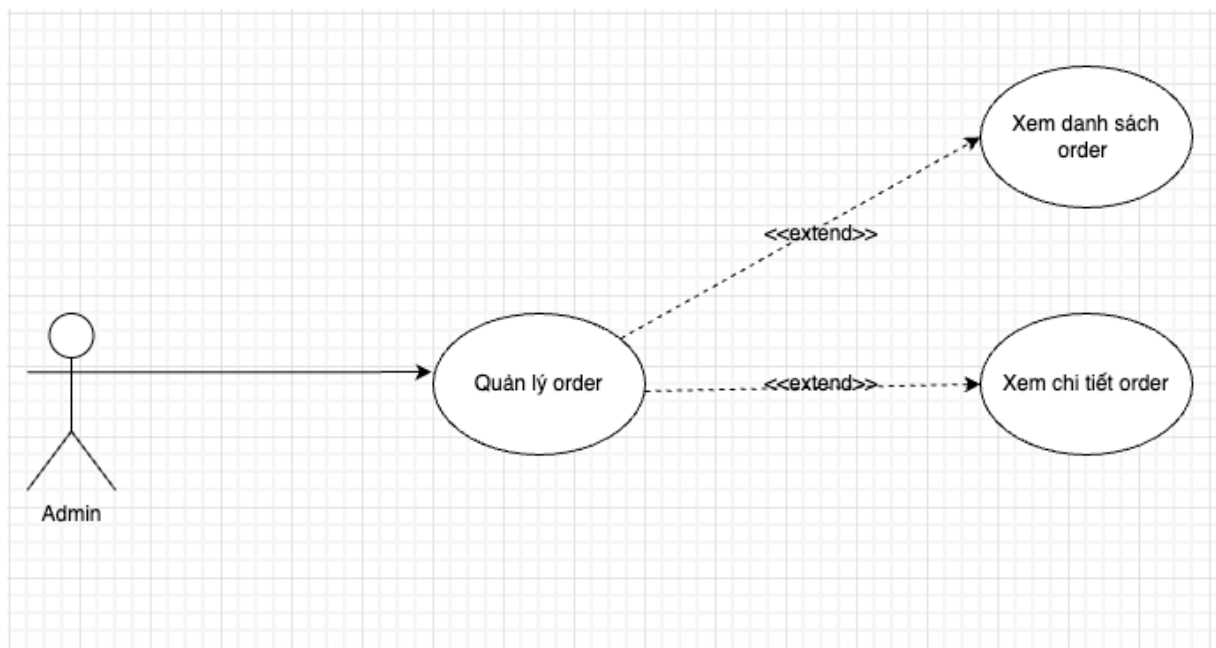
Hình 2.13 Biểu đồ tuần tự xem danh sách khách hàng



Hình 2.14 Biểu đồ tuần tự xóa khách hàng

2.3.6 Chức năng quản lý order

- Biểu đồ Use Case quản lý order



Hình 2.15 Biểu đồ Use Case quản lý order

- Mô tả Use Case quản lý order

Bảng 2.11 Mô tả Use Case quản lý order

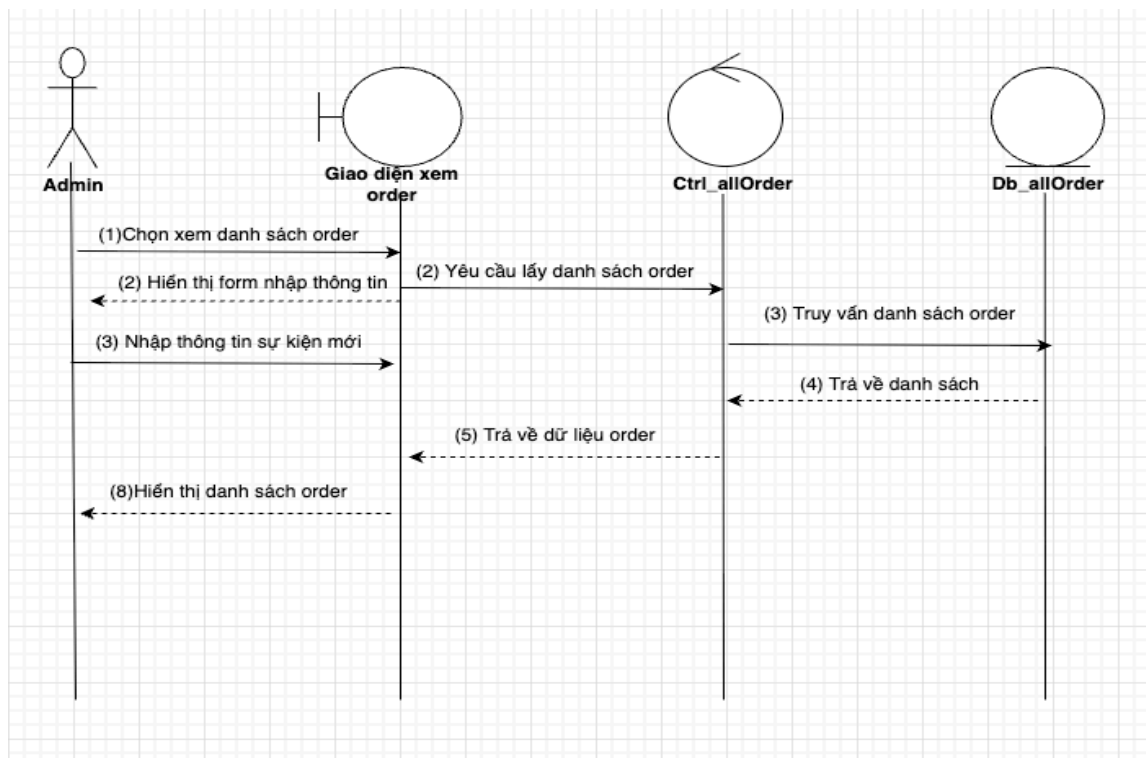
Tên Use Case	Quản lý order
Tác nhân	Admin
Điều kiện đầu vào	Admin đăng nhập và truy cập chức năng quản lý order
Kết quả đầu ra	Order được hiển thị hoặc bị xóa khỏi hệ thống
Mô tả	Use Case cho phép Admin xem danh sách order, xem chi tiết từng order và xóa order nếu cần

- Dòng chính Use Case quản lý order

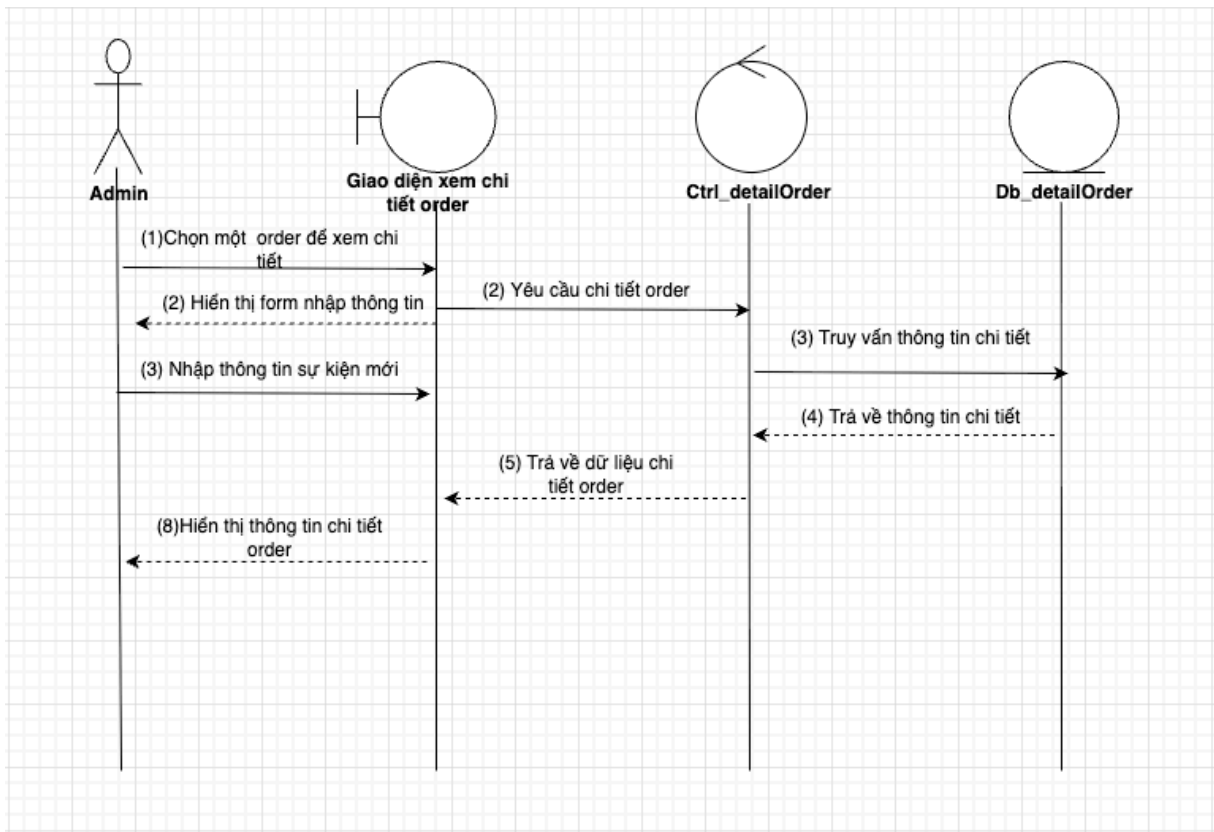
Bảng 2.12 Dòng chính Use Case quản lý order

STT	Thực hiện	Hành động
1	Tác nhân	Chọn chức năng Quản lý order
2	Hệ thống	Hiển thị giao diện quản lý order
3	Tác nhân	Chọn thao tác (Xem danh sách / Xem chi tiết)
4	Hệ thống	Thực hiện chức năng tương ứng
5	Hệ thống	Cập nhật lại giao diện order sau thao tác

- Biểu đồ tuần tự quản lý order



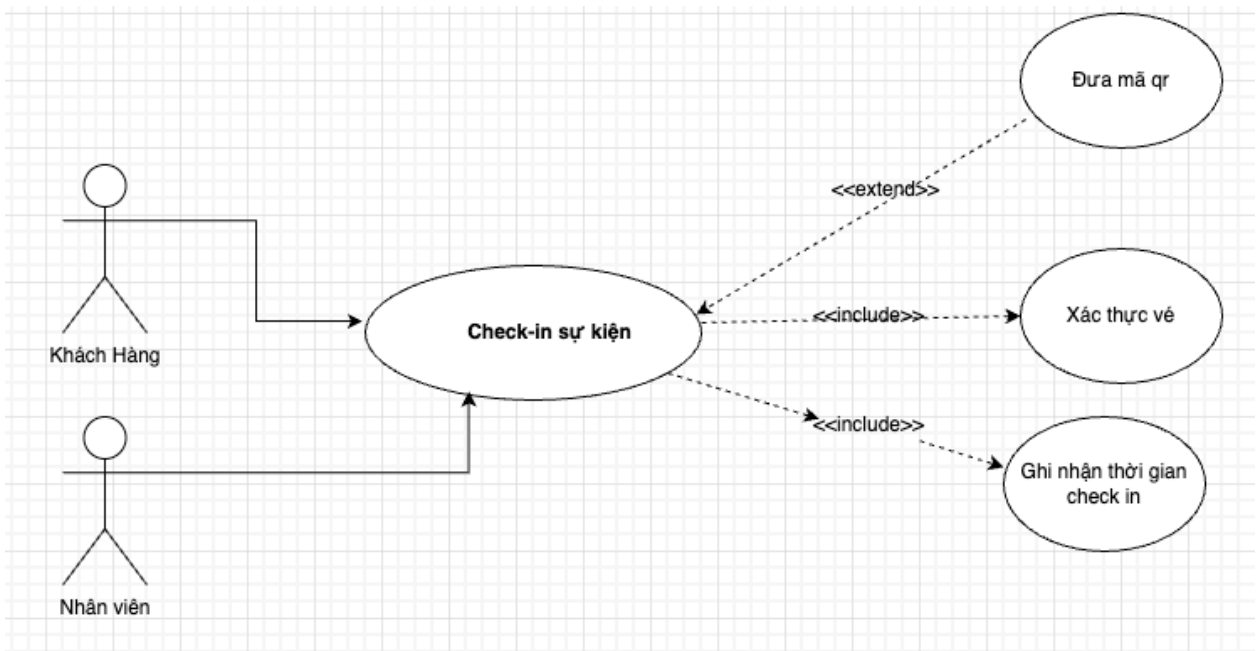
Hình 2.16 Biểu đồ tuần tự xem danh sách order



Hình 2.17 Biểu đồ tuần tự xem chi tiết order

2.3.7 Chức năng check-in sự kiện

- Biểu đồ Use Case check-in sự kiện



Hình 2.18 Biểu đồ Use Case check-in sự kiện

- Mô tả Use Case check-in sự kiện

Bảng 2.13 Mô tả Use Case check-in sự kiện

Tên Use Case	Check-in sự kiện
Tác nhân	Khách hàng, Nhân viên
Điều kiện đầu vào	Khách hàng đã mua vé hợp lệ hoặc nhân viên có quyền check-in
Kết quả đầu ra	Vé được xác thực và ghi nhận thời gian check-in thành công
Mô tả	Use case cho phép khách hàng hoặc nhân viên thực hiện check-in sự kiện bằng mã QR. Hệ thống sẽ xác thực vé và ghi lại thời gian check-in

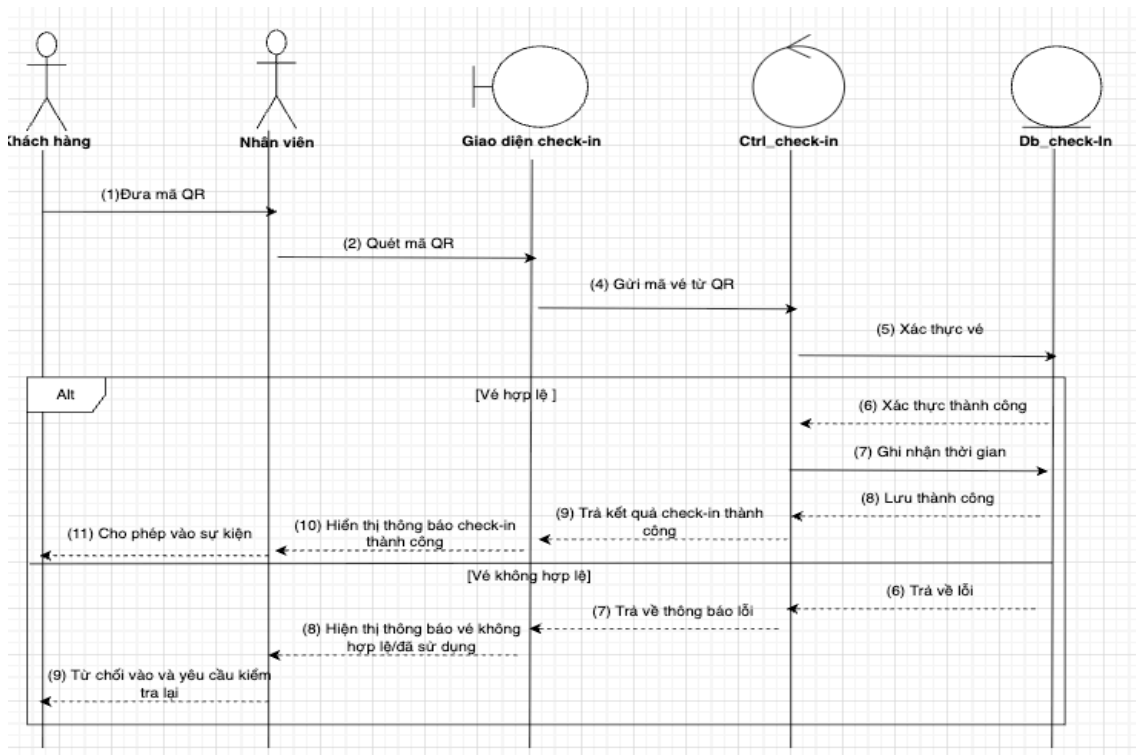
- Dòng chính Use Case check-in sự kiện

Bảng 2.14 Dòng chính Use Case check-in sự kiện

STT	Thực hiện	Hành động
1	Tác nhân	Chọn chức năng Check-in sự kiện
2	Hệ thống	Hiển thị giao diện check-in
3	Tác nhân	Đưa mã QR cho hệ thống quét (<i>Extend</i>)
4	Hệ thống	Thực hiện Xác thực vé (<i>Include</i>)
5	Hệ thống	Thực hiện Ghi nhận thời gian check-in (<i>Include</i>)
6	Hệ thống	Hiển thị thông báo check-in thành công

--	--	--

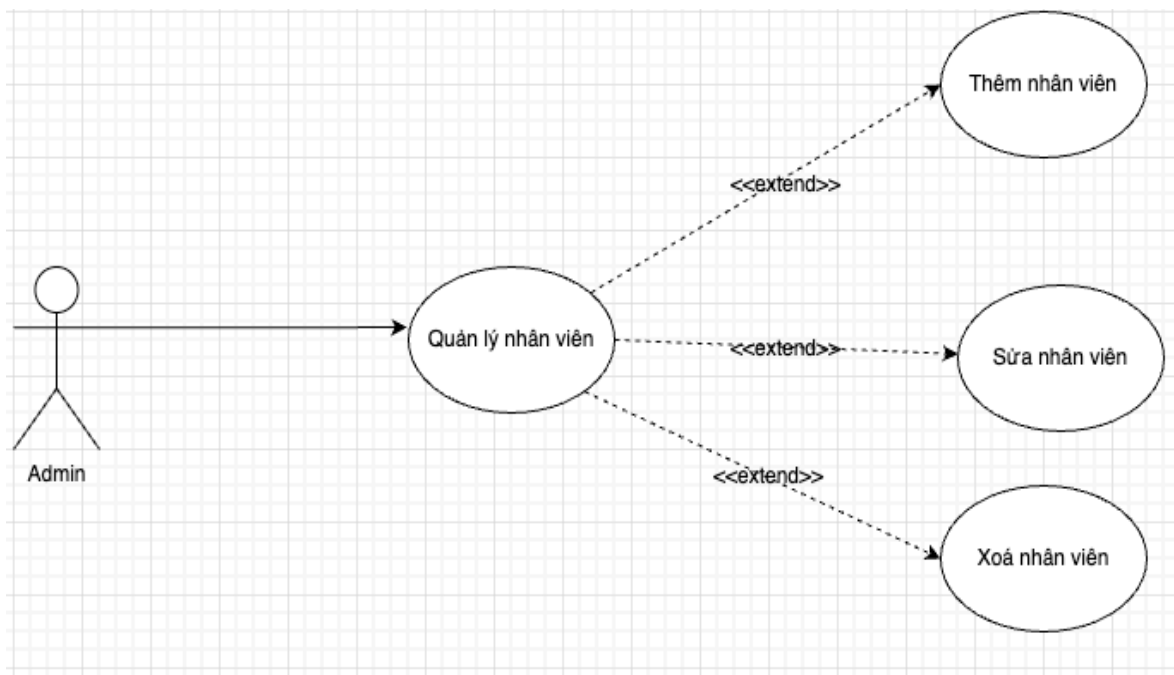
- Biểu đồ tuần tự check-in sự kiện



Hình 2.19 Biểu đồ tuần tự check-in sự kiện

2.3.8 Chức năng quản lý nhân viên

- Biểu đồ Use Case quản lý nhân viên



Hình 2.20 Biểu đồ Use Case quản lý nhân viên

- Mô tả Use Case quản lý nhân viên

Bảng 2.15 Mô tả Use Case quản lý nhân viên

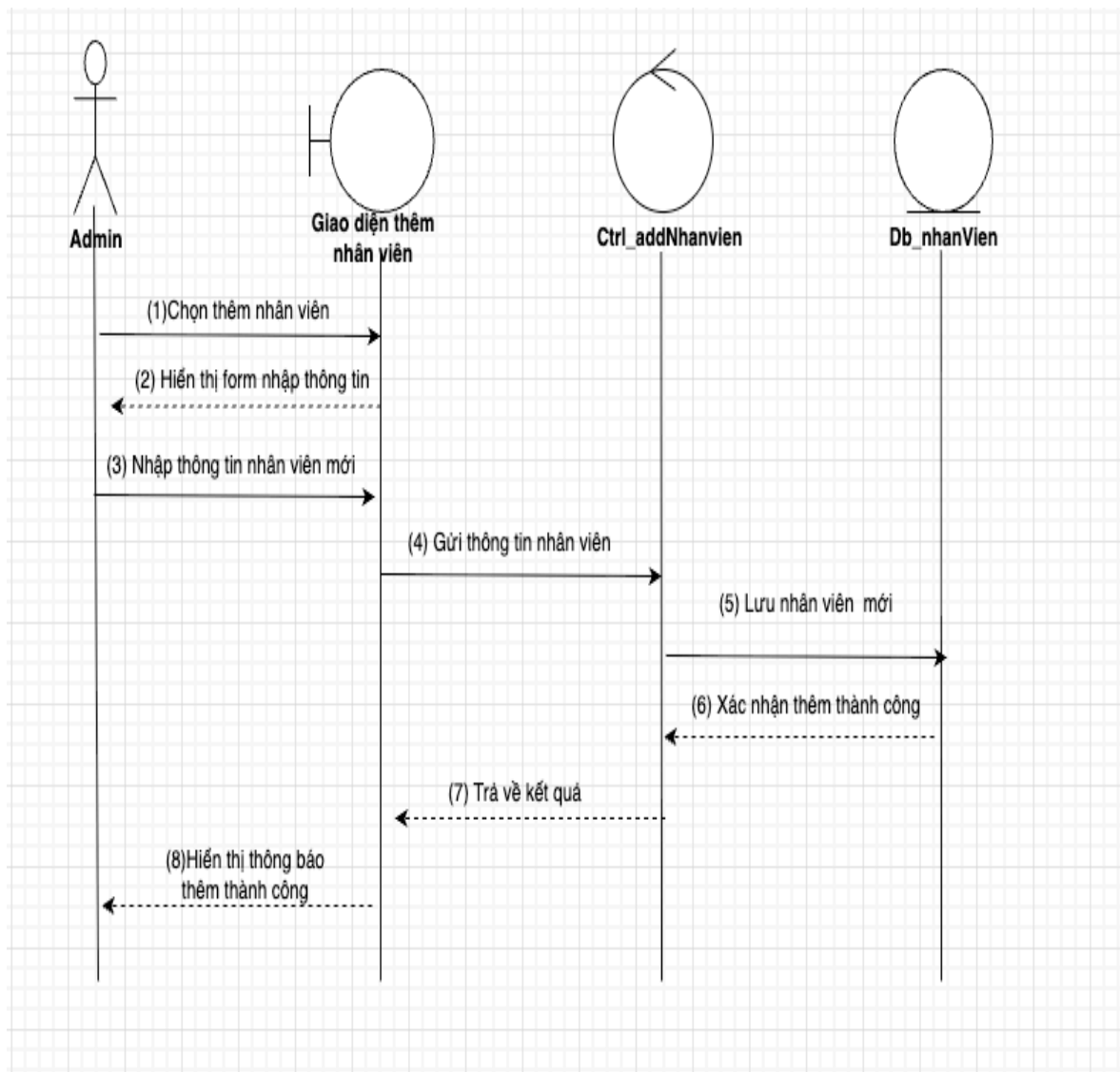
Tên Use Case	Quản lý nhân viên
Tác nhân	Admin
Điều kiện đầu vào	Admin đăng nhập và truy cập chức năng quản lý nhân viên
Kết quả đầu ra	Admin được xem danh sách nhân viên, thêm, sửa hoặc xóa thành công
Mô tả	Use Case cho phép Admin xem danh sách nhân viên, thêm, sửa hoặc xóa thông tin nhân viên

- Dòng chính Use Case quản lý nhân viên

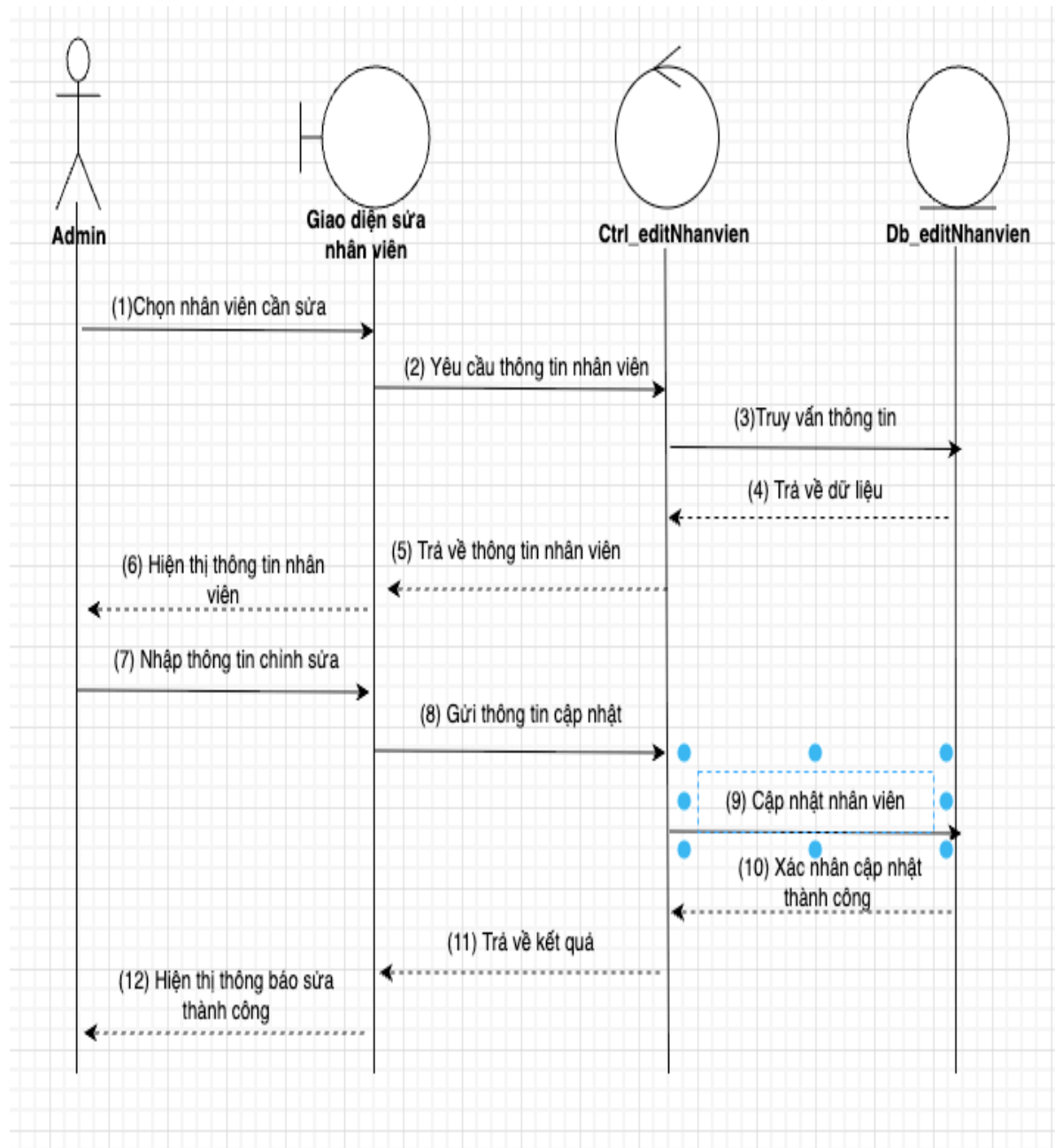
Bảng 2.16 Dòng chính Use Case quản lý nhân viên

STT	Thực hiện	Hành động
1	Tác nhân	Admin truy cập mục Quản lý nhân viên.
2	Hệ thống	Hiển thị danh sách nhân viên (Họ tên, Email, Vai trò).
3	Tác nhân	Admin chọn Thêm nhân viên.
4	Hệ thống	Hiển thị form nhập thông tin nhân viên mới.
5	Tác nhân	Nhập thông tin: Họ tên, Email, Mật khẩu, Số điện thoại, Vai trò.
6	Hệ thống	Kiểm tra dữ liệu hợp lệ và lưu vào cơ sở dữ liệu.
7	Hệ thống	Hiển thị thông báo: “Thêm nhân viên thành công”.
8	Tác nhân	Admin chọn 1 nhân viên → nhấn Sửa.
9	Hệ thống	Hiển thị form cập nhật thông tin.
10	Tác nhân	Chỉnh sửa thông tin và nhấn Lưu.
11	Hệ thống	Cập nhật dữ liệu trong cơ sở dữ liệu.
12	Tác nhân	Admin chọn Xóa nhân viên.
13	Hệ thống	Hiển thị xác nhận xóa.
14	Tác nhân	Xác nhận xóa.
15	Hệ thống	Xóa hoặc khóa tài khoản nhân viên trong hệ thống.

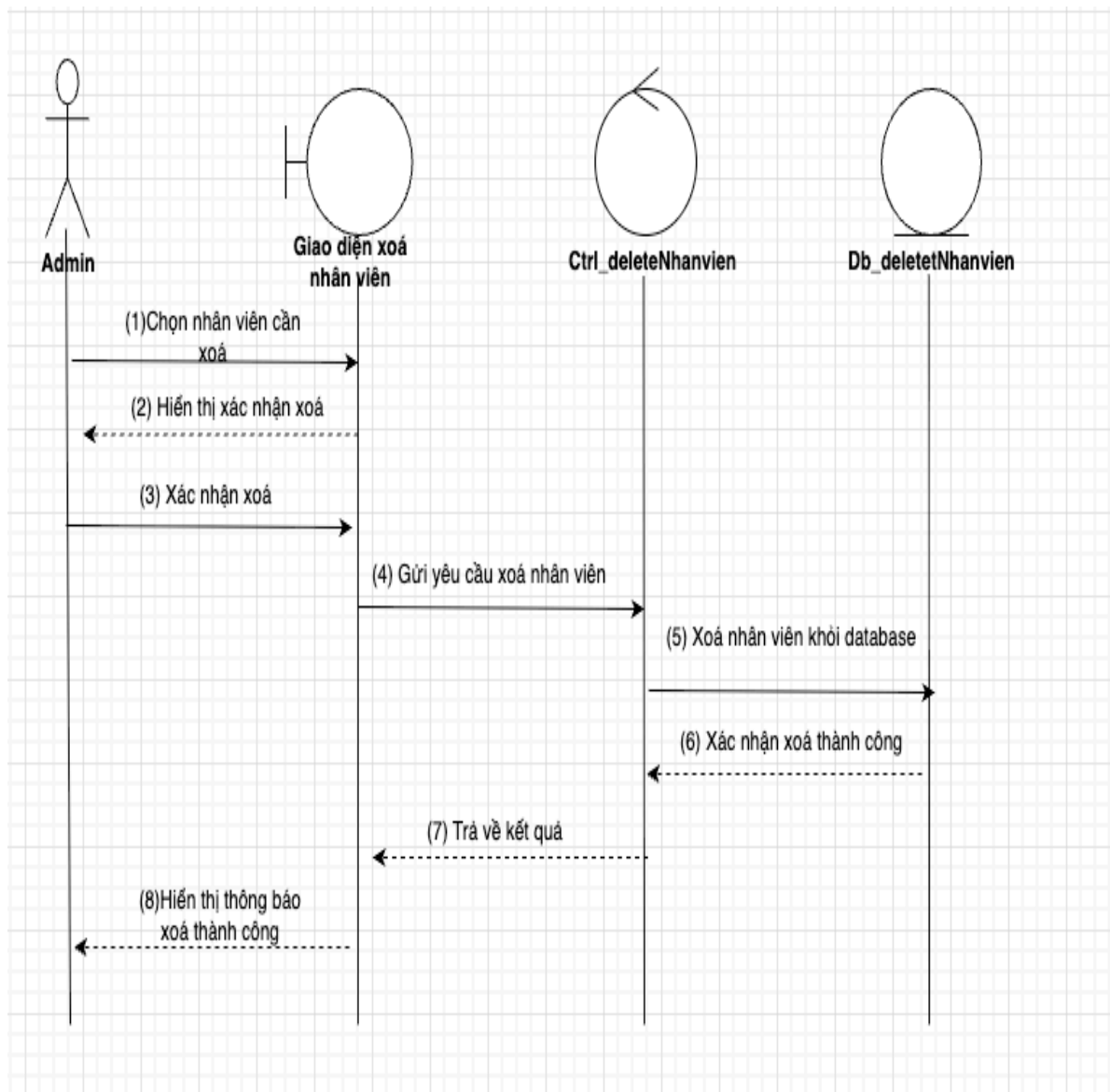
- Biểu đồ tuần tự quản lý nhân viên



Hình 2.21 Biểu đồ tuần tự thêm nhân viên



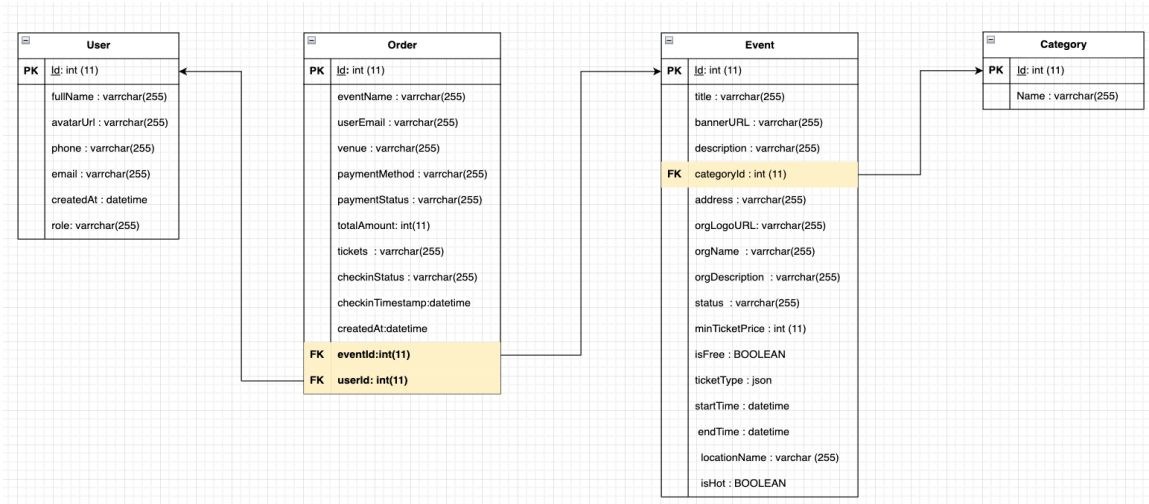
Hình 2.22 Biểu đồ tuần tự sửa nhân viên



Hình 2.23 Biểu đồ tuần tự xóa nhân viên

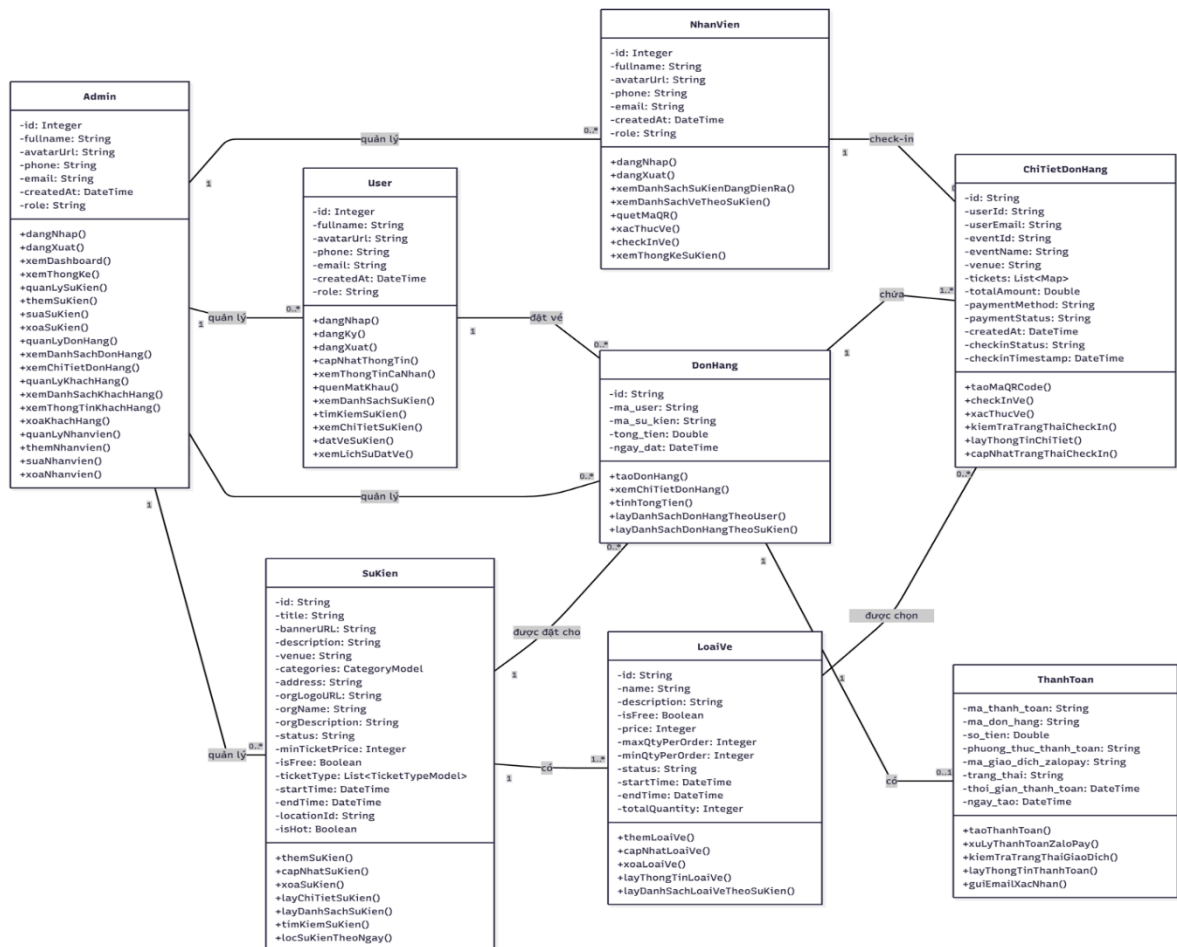
2.4. Thiết kế cơ sở dữ liệu

2.4.1 Mô hình quan hệ



Hình 2.24 Mô hình quan hệ

2.4.2 Mô hình sơ đồ lớp



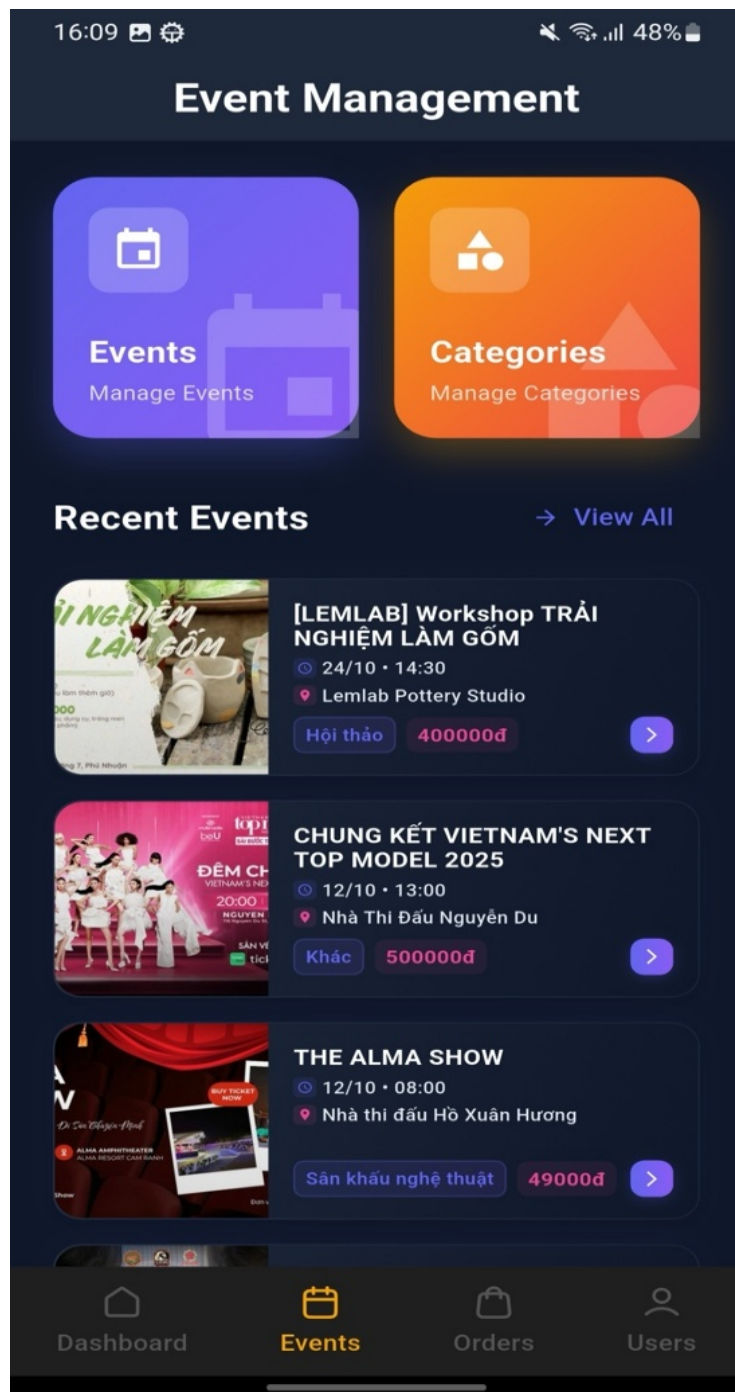
Hình 2.25 Mô hình sơ đồ lớp

2.5. Thiết kế giao diện

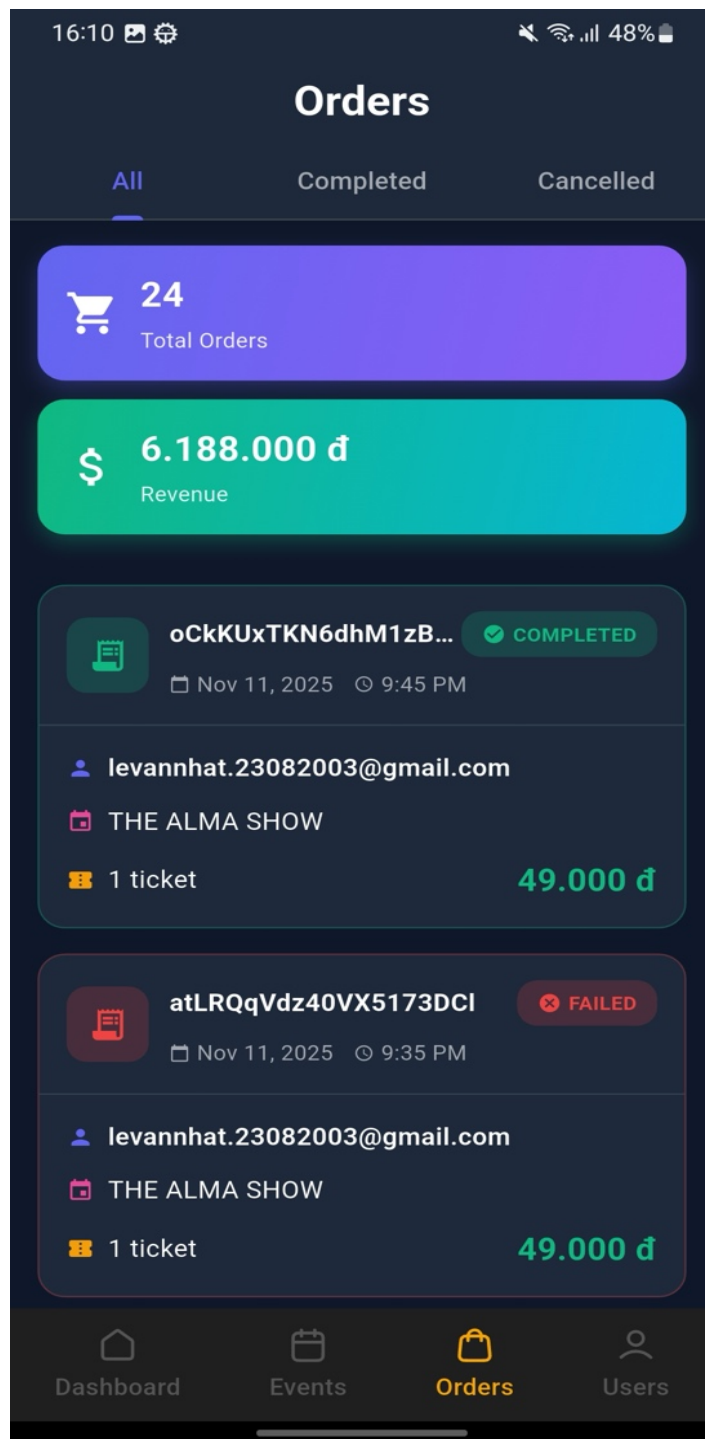
2.5.1 Giao diện Admin



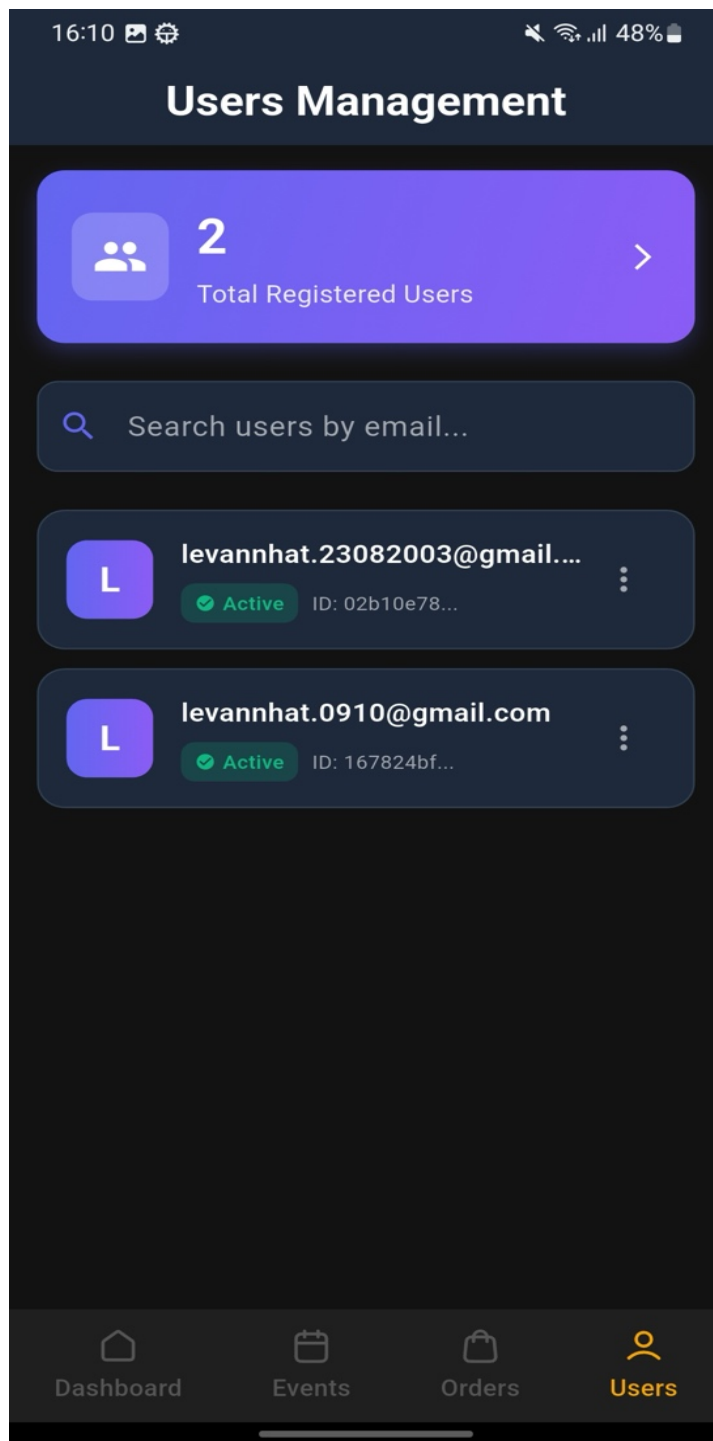
Hình 2.26 Màn hình dashboard



Hình 2.27 Màn hình quản lý sự kiện

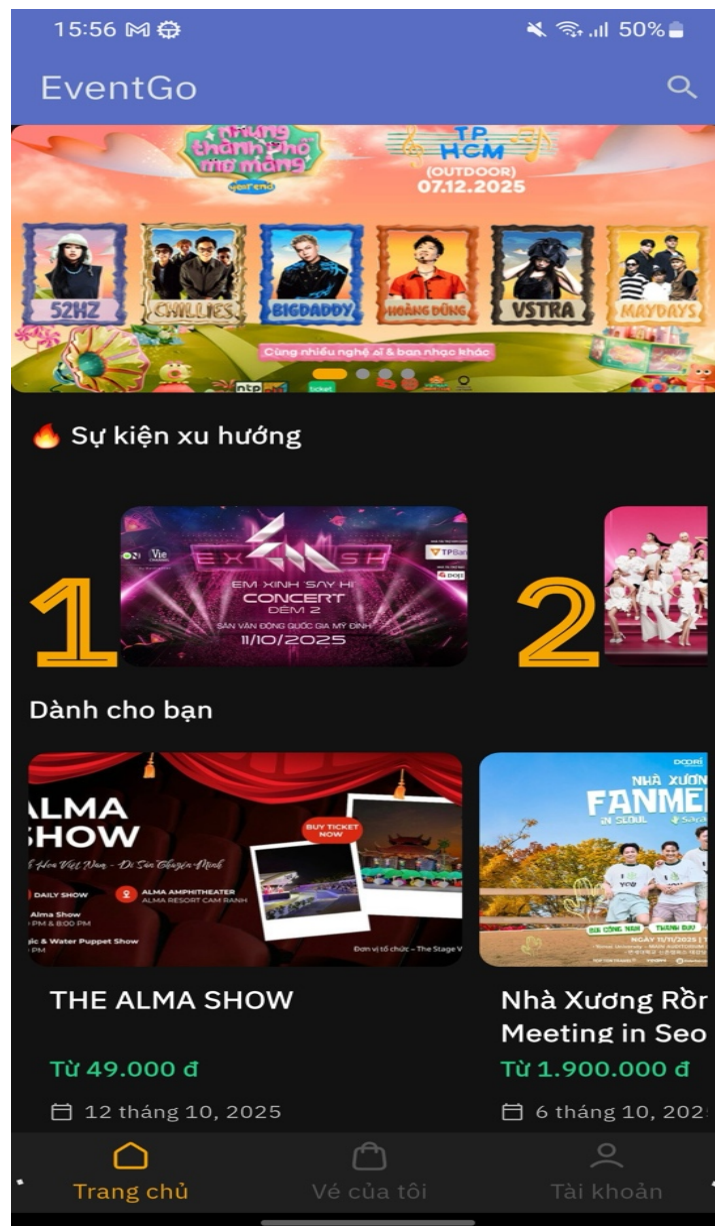


Hình 2.28 Màn hình quản lý order

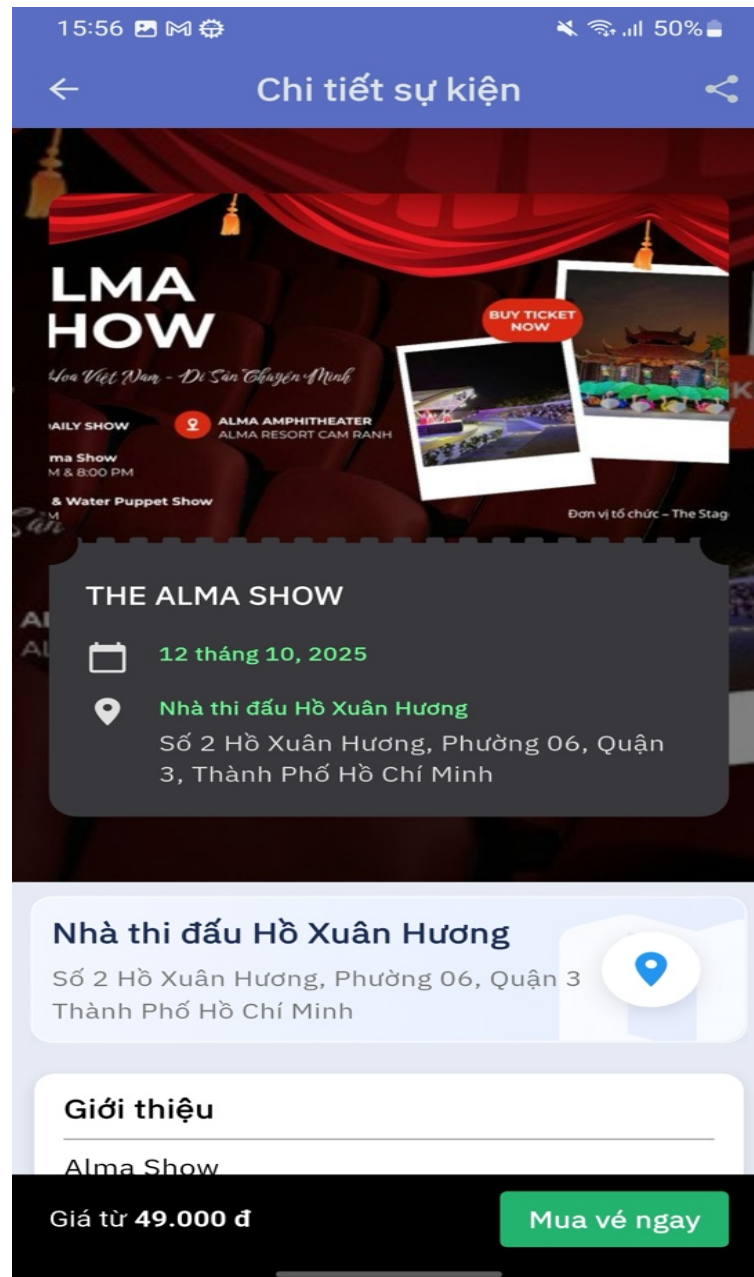


Hình 2.29 Màn hình quản lý khách hàng

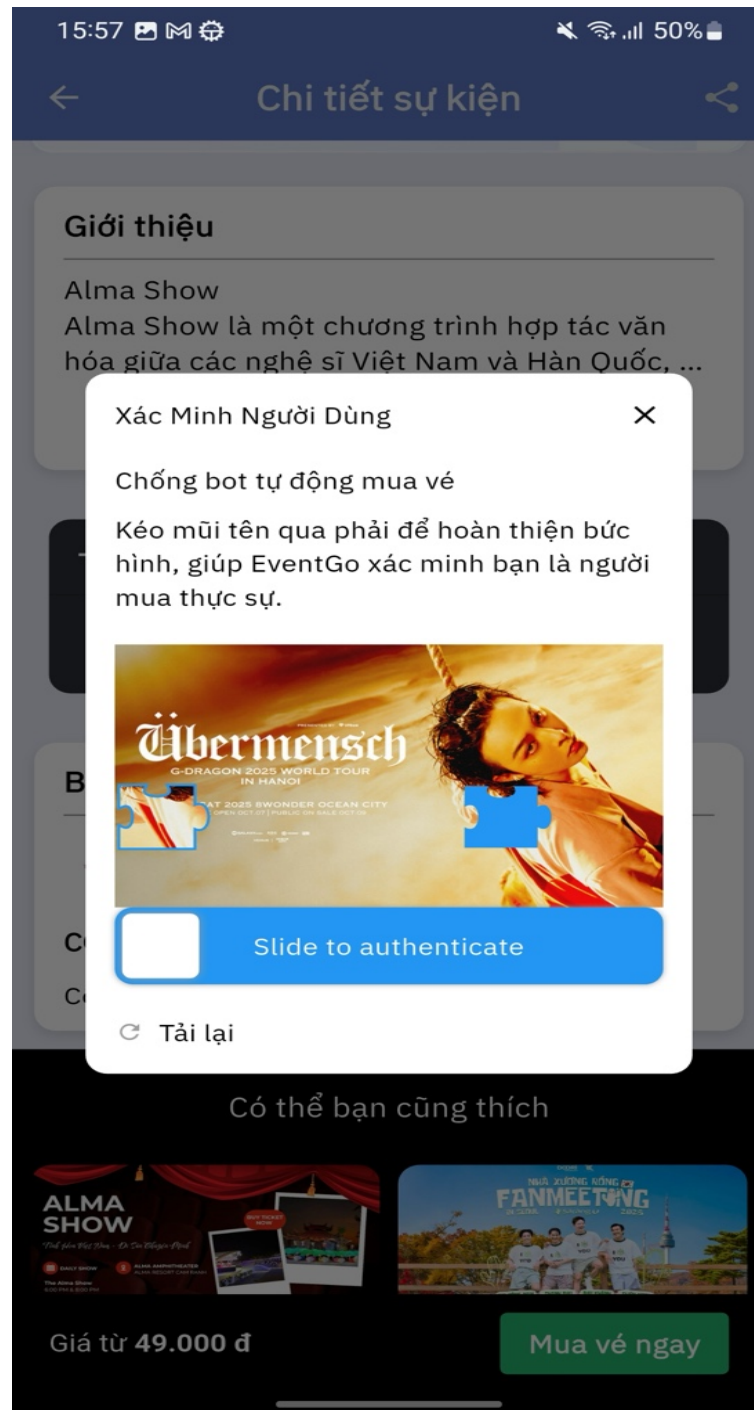
2.5.2 Giao diện người dùng



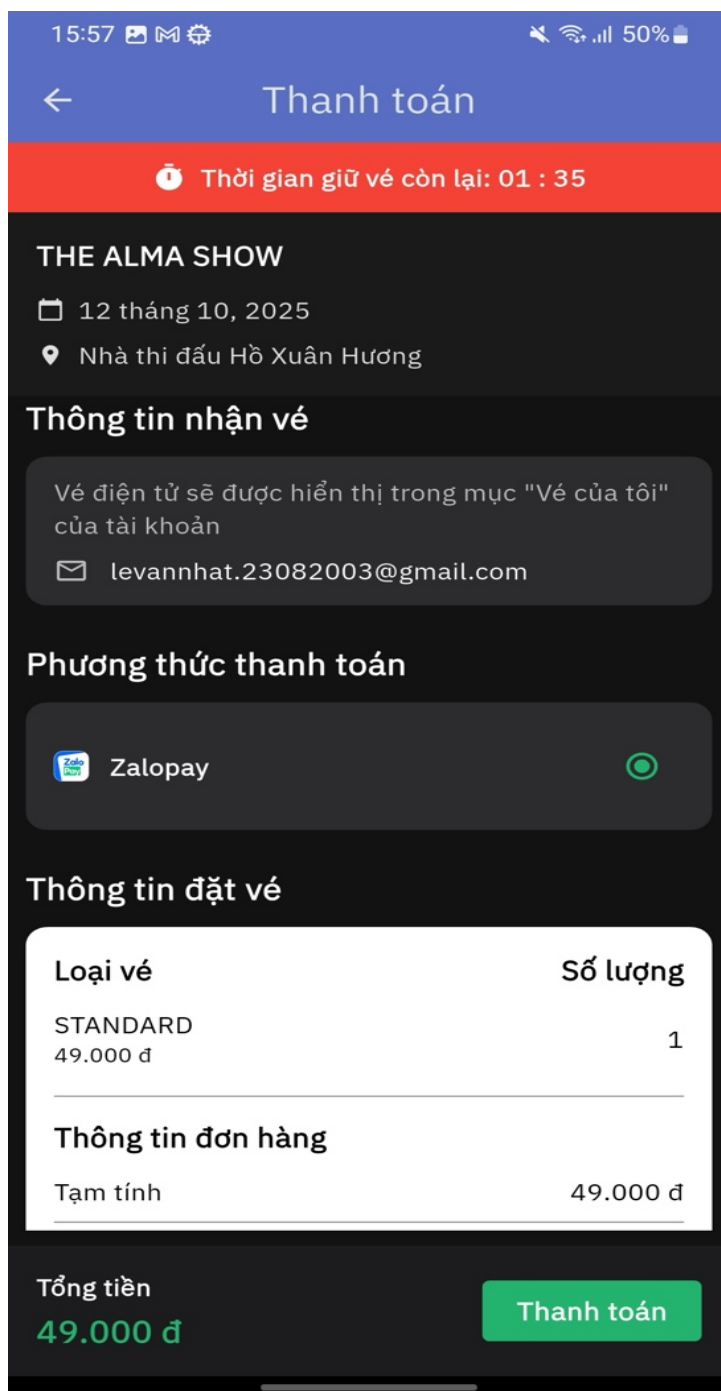
Hình 2.30 Màn hình trang chủ



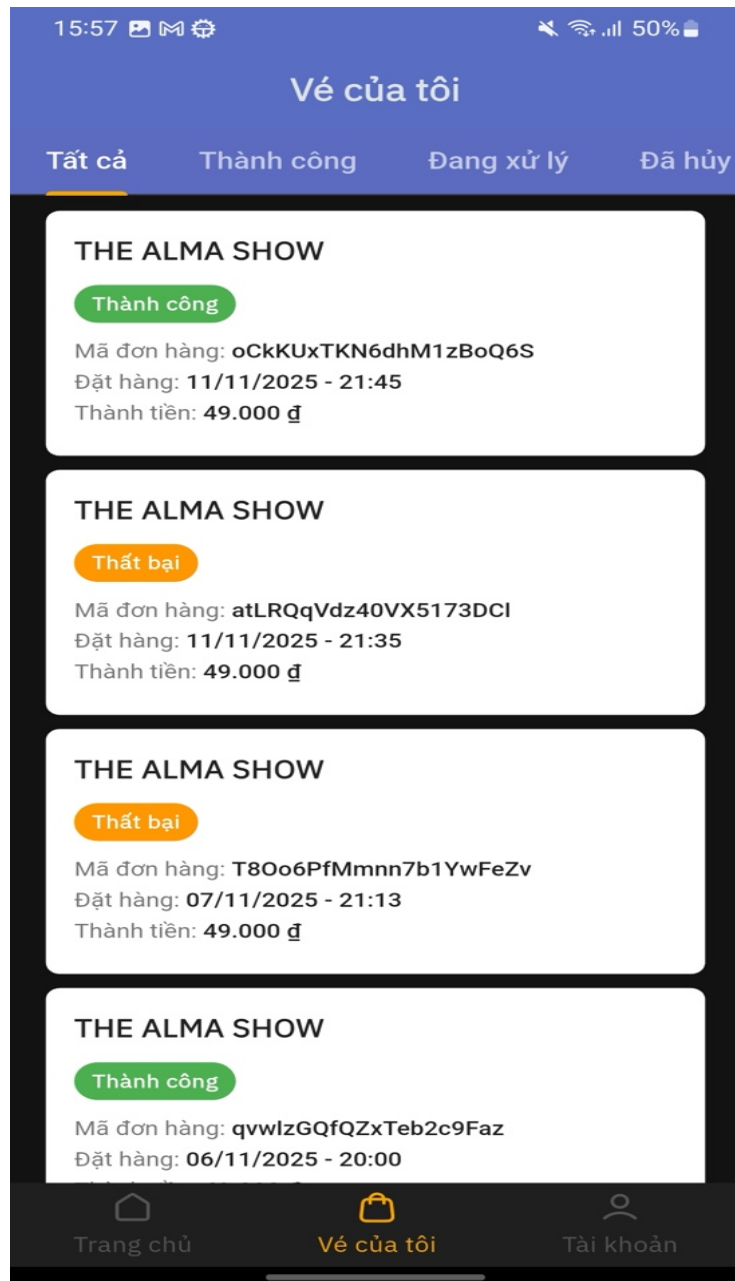
Hình 2.31 Màn hình chi tiết sự kiện



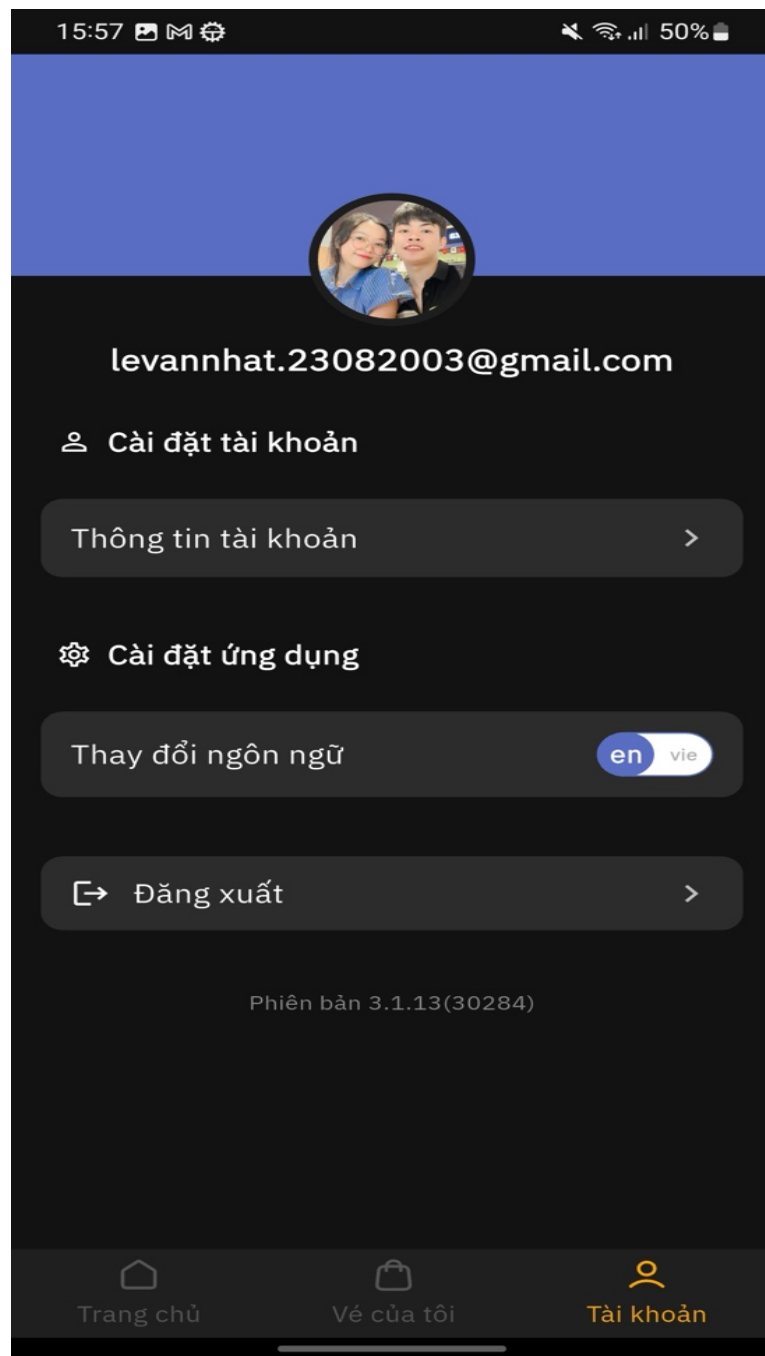
Hình 2.32 Màn hình xác minh bot



Hình 2.33 Màn hình thanh toán



Hình 2.34 Màn hình vé của tôi



Hình 2.35 Màn hình tài khoản

CHƯƠNG 3 : CÀI ĐẶT VÀ PHÁT TRIỂN ỨNG DỤNG

3.1. Cấu trúc thư mục code

3.1.1. Cấu trúc tổng thể thư mục dự án Flutter

— core	# Core layer chứa config, base class, constants,...
— base	# Class cơ bản dùng chung (BaseView, BaseViewModel)
— config	# Tập tin cấu hình (Supabase, ZaloPay, môi trường, API...)
— constants	# Chứa constants (màu, text, spacing, image path...)
— utils	# Hàm tiện ích (formatter, extension, validator...)
— widgets	# Các widget tái sử dụng toàn app (button, dialogs, input field,...)
— data	# Data layer – xử lý dữ liệu
— models	# Định nghĩa model / JSON parser
— repositories	# Triển khai repository (gọi dữ liệu từ API / DB)
— domain	# Business logic thuần (Không phụ thuộc Flutter)
— entities	# Entity chuẩn domain (model thuần - không JSON parser)
— usecase	# UseCase xử lý nghiệp vụ (login, getEvent, updateProfile,...)
— injection	# Khởi tạo và đăng ký dependency (GetIt / Provider Riverpod...)
— presentation	# UI + ViewModels (MVVM)
— pages	# Màn hình UI (View)
— view_models	# ViewModel (ChangeNotifier) xử lý logic cho từng màn

|— routers # File router điều hướng (Route name, GoRouter, etc)

|— main.dart # Entry point – chạy app, inject DI, runApp()

3.2. Xử lý thanh toán qua Zalopay

3.2.1. Vấn đề

Trong hệ thống đặt vé sự kiện trực tuyến, việc tích hợp phương thức thanh toán là yếu tố không thể thiếu nhằm đảm bảo người dùng có thể thanh toán và nhận vé một cách tự động, nhanh chóng và an toàn. Tuy nhiên, quá trình xử lý thanh toán đối mặt với một số thách thức đáng chú ý:

Xác nhận thanh toán theo thời gian thực: Khi người dùng tiến hành thanh toán, hệ thống cần xác định chính xác liệu giao dịch có thành công hay không. Điều này giúp đảm bảo vé chỉ được phát hành nếu thanh toán được xác nhận, tránh trường hợp đặt vé ảo hoặc bán trùng suất.

Bảo mật thông tin thanh toán: Khóa bảo mật (key1, key2) do ZaloPay cung cấp không được lưu trữ trực tiếp trên ứng dụng Flutter, vì mã nguồn có thể bị decompile và lộ thông tin. Việc ký dữ liệu và xác thực callback cần thực hiện ở phía backend để đảm bảo an toàn theo tiêu chuẩn từ ZaloPay.

Đồng bộ trạng thái giữa ZaloPay – Backend – CSDL: Sau khi người dùng thanh toán, ZaloPay gửi callback về server, nhưng có thể xảy ra trường hợp callback bị chậm hoặc thất bại. Do đó, hệ thống cần cơ chế dự phòng, chẳng hạn như chủ động gọi API truy vấn trạng thái từ ZaloPay nếu sau khoảng thời gian nhất định chưa nhận được phản hồi. Điều này giúp bảo toàn tính nhất quán dữ liệu cho hệ thống.

3.2.2. Cách giải quyết

Để giải quyết các vấn đề trên, hệ thống áp dụng mô hình xử lý thanh toán Client–Server với sự tham gia của ba thành phần: ứng dụng Flutter (client), backend API (server) và dịch vụ ZaloPay. Quy trình xử lý thanh toán được đề xuất như sau:

Tạo giao dịch thanh toán trên backend: Khi người dùng chọn sự kiện và bấm "Thanh toán bằng ZaloPay", ứng dụng Flutter gửi yêu cầu tới backend với thông tin đơn hàng (mã đặt chỗ, số tiền, người dùng...). Backend sau đó tạo ra mã giao dịch theo chuẩn yymmdd_x (ví dụ: 250117_001) để đồng bộ với ZaloPay.

Gọi API tạo đơn hàng của ZaloPay: Backend sử dụng các tham số cần thiết như app_id, amount, app_trans_id... và ký mac bằng key1 (MAC key) để tạo đơn hàng qua API /v2/create. Nếu giao dịch hợp lệ, ZaloPay phản hồi với zp_trans_token.

Thanh toán qua SDK ZaloPay: Token `zp_trans_token` được gửi về ứng dụng Flutter. Flutter sẽ sử dụng gói `flutter_zalopay_sdk` để mở ứng dụng ZaloPay và chuyển người dùng sang giao diện thanh toán.

Xử lý callback từ ZaloPay: Sau khi thanh toán, ZaloPay gửi dữ liệu callback về backend. Backend sẽ tiến hành xác thực chữ ký mac bằng `key2` (callback key). Nếu dữ liệu hợp lệ, trạng thái giao dịch được cập nhật thành PAID trong cơ sở dữ liệu.

Xử lý dự phòng với API truy vấn: Trong trường hợp backend không nhận được callback, hệ thống sẽ chủ động gọi API `/v2/query` để kiểm tra giao dịch. Điều này giúp đảm bảo trạng thái thanh toán luôn chính xác ngay cả khi callback bị lỗi.

Cập nhật giao diện người dùng: Sau khi trạng thái giao dịch được cập nhật (qua callback hoặc truy vấn), ứng dụng Flutter sẽ tự động lấy dữ liệu mới từ Firebase/Supabase hoặc gọi API để hiển thị vé điện tử cho người dùng.

3.2.3. *Module code*

```
Future<String?> createPaymentOrder() async {  
  
  setLoading(true);  
  
  int amount = grandTotal.toInt();  
  
  if (amount < 1000 || amount > 1000000) {  
  
    _zpTransToken = "Invalid Amount";  
  
    setLoading(false);  
  
    return null;  
  
  }  
  
  try {  
  
    var result = await createOrder(amount);  
  
    if (result != null) {  
  
      _zpTransToken = result.zptranstoken;  
  
      setLoading(false);  

```

```
return _zpTransToken;

} else {

  setLoading(false);

  return null;

}

} catch (e) {

  _zpTransToken = "Error: $e";

  setLoading(false);

  return null;

}

}
```

```
AppElevatedButton(

  text: buttonText,

  onPressed: () async {

    if (hasTickets) {

      showDialog(

        context: context,

        barrierDismissible: false,

        builder: (BuildContext context) {

          return Consumer<HomeViewModel>(
```

```
builder: (context, vm, child) => vm.isLoading
```

```
? Center(child: CircularProgressIndicator())
```

```
: SizedBox.shrink(),
```

```
);
```

```
},
```

```
);
```

```
final token = await vmReader.createPaymentOrder();
```

```
if (!mounted) return;
```

```
context.pop();
```

```
if (token != null) {
```

```
context.push(RouterPath.payment, extra: token);
```

```
} else {
```

```
ScaffoldMessenger.of(context).showSnackBar(
```

```
SnackBar(content: Text(vmReader.zpTransToken)),
```

```
);
```

```
}
```

```
}
```

```
},
```

```
Future<CreateOrderResponse?> createOrder(int price) async {

  final Dio dio = Dio();

  var body = new Map<String, String>();

  body["app_id"] = ZaloPayConfig.appId;

  body["app_user"] = ZaloPayConfig.appUser;

  body["app_time"] = DateTime.now().millisecondsSinceEpoch.toString();

  body["amount"] = price.toStringAsFixed(0);

  body["app_trans_id"] = utils.getAppTransId();

  body["embed_data"] = "{}";

  body["item"] = "[]";

  body["bank_code"] = utils.getBankCode();

  body["description"] = utils.getDescription(body["app_trans_id"]!);


  var dataGetMac = sprintf("%s|%s|%s|%s|%s|%s|%s", [

    body["app_id"],

    body["app_trans_id"],

    body["app_user"],

    body["amount"],

    body["app_time"],

    body["embed_data"],
```

```
body["item"],

]);

body["mac"] = utils.getMacCreateOrder(dataGetMac);

print("mac: ${body["mac"]}");


print("body_request: $body");


try {

Response response = await dio.post(

Endpoints.createOrderUrl,

data: body, // Dio dùng 'data' thay vì 'body'

options: Options(

contentType: Headers.formUrlEncodedContentType,

),

);

var data = response.data;

print("data_response: $data");


return CreateOrderResponse.fromJson(data);

} on DioException catch (e) {

print("DioError: ${e.message}");

if (e.response != null) {
```

```

print("DioError response: ${e.response?.data}");

}

return null;

} catch (e) {

print("Unexpected error: $e");

return null;

}

}

```

3.3. Xử lý vấn đề bảo mật thông tin

3.3.1. Vấn đề

Trong quá trình phát triển ứng dụng đặt vé sự kiện đa nền tảng, một trong những yếu tố quan trọng nhất là đảm bảo bảo mật thông tin người dùng, đặc biệt khi ứng dụng có khả năng truy cập dữ liệu trực tiếp từ phía client (Flutter). Nếu dữ liệu không được bảo vệ hợp lý:

Người dùng có thể xem, sửa dữ liệu của người khác.

Kẻ xấu có thể truy vấn bảng người dùng thông qua Supabase API public.

Dữ liệu quan trọng như thông tin cá nhân, lịch sử thanh toán, email... có thể bị lộ.

Mặc dù Supabase cung cấp sẵn tính năng Authentication, nhưng để kiểm soát truy cập các bản ghi (rows) trong cơ sở dữ liệu, cần bật Row-Level Security (RLS) và cấu hình chính sách (policies) cụ thể. Nếu không bật RLS, tất cả dữ liệu trong bảng có thể bị lộ qua các truy vấn đơn giản.

3.3.2. Cách giải quyết

Supabase Authentication :

Supabase cung cấp cơ chế đăng ký, đăng nhập, quản lý phiên (session) tự động.

Sau mỗi lần đăng nhập, ứng dụng Flutter có thể gọi trực tiếp Supabase SDK và Supabase sẽ tự đính kèm thông tin phiên (session) vào tất cả các truy vấn đến cơ sở dữ liệu.

Không cần xử lý thủ công token hay xây dựng backend trung gian.

Row-Level Security (RLS) :

RLS là cơ chế bảo mật dữ liệu cấp hàng của Supabase.

Khi RLS được bật, mỗi truy vấn từ client sẽ bị giới hạn theo chính sách mà developer định nghĩa.

Nếu không có policy phù hợp, bảng đó sẽ không thể truy cập từ client.

Supabase giúp kích hoạt, tạo policy dễ dàng ngay trong Dashboard → Table → Policies.

3.3.3. Module code

```
Future<bool> login(String email, String password) async {  
  
  try {  
  
    _setLoading(true);  
  
    _clearError();  
  
  
    final result = await _loginUseCase(email, password);  
  
  
    if (result.isSuccess && result.user != null) {  
  
      final profile = await Supabase.instance.client  
        .from('profiles')  
        .select('role')  
        .eq('id', result.user!.id)  
        .maybeSingle();  
  
      if (profile == null) {  
  
        await _createUserProfile(result.user!);  

```

```
}

if (profile == null || profile['role'] != 'admin') {

  _currentUser = result.user;

  await _saveLoginState(true);

  _setLoading(false);

  return true;

} else {

  await logout();

  _setLoading(false);

  _setError("Admin vui lòng đăng nhập ở trang quản trị.");

  return false;

}

} else {

  _setLoading(false);

  _setError(result.errorMessage ?? AppStrings.loginFailed);

  return false;

}

} catch (e) {

  _setLoading(false);

  _setError(AppStrings.unknownError);

  return false;

}
```

```
}  
  
}  
  
Future<bool> register(String email, String password) async {  
  
  try {  
  
    _setLoading(true);  
  
    _clearError();  
  
  
    final result = await _registerUseCase(email, password);  
  
  
    if (result.isSuccess && result.user != null) {  
  
      await Supabase.instance.client.from('profiles').insert({  
  
        'id': result.user!.id,  
  
        'role': 'user',  
  
        'email': result.user!.email,  
  
      });  
  
  
      _currentUser = result.user;  
  
      await _saveLoginState(true);  
  
      _setLoading(false);  
  
      return true;  
  
    } else {
```

```
    _setLoading(false);

    _setError(result.errorMessage ?? AppStrings.signUpFailed);

    return false;

  }

} catch (e) {

  _setLoading(false);

  _setError(AppStrings.unknownError);

  return false;

}

}
```

KẾT LUẬN

Trên đây là toàn bộ nội dung báo cáo đồ án tốt nghiệp. Sau một thời gian nỗ lực nghiên cứu, thiết kế và lập trình, em đã hoàn thành đồ án tốt nghiệp với đề tài “Phát triển ứng dụng đặt vé sự kiện đa nền tảng sử dụng Flutter và Firebase” mang tên sản phẩm EventGo.

I. KẾT QUẢ ĐẠT ĐƯỢC

Về công nghệ:

- Nắm vững và thành thạo việc phát triển ứng dụng đa nền tảng bằng Flutter kết hợp ngôn ngữ Dart, từ việc xây dựng giao diện phức tạp đến quản lý trạng thái hiệu quả bằng Provider và Clean Architecture.
- Hiểu sâu và áp dụng thành công các dịch vụ Backend-as-a-Service hiện đại: Supabase (Authentication + Realtime + RLS) và Firebase (Firestore, Storage, Cloud Functions).
- Tích hợp thành công thanh toán nội địa qua ZaloPay SDK với cơ chế xử lý an toàn (tạo đơn hàng và xác thực callback ở backend).
- Làm chủ quy trình triển khai bảo mật cấp độ hàng (Row Level Security) và phân quyền người dùng chặt chẽ trên Supabase.
- Biết cách thiết kế, phân tích yêu cầu, xây dựng kiến trúc phần mềm sạch và triển khai một ứng dụng di động hoàn chỉnh từ A-Z.

Về chương trình:

- Hoàn thành đầy đủ các mục tiêu chức năng đã đề ra ban đầu: đăng ký/đăng nhập, tìm kiếm và lọc sự kiện, đặt vé, thanh toán trực tuyến, quản lý vé điện tử (QR code), check-in bằng mã QR, giao diện quản trị cho Admin.
- Đảm bảo tính bảo mật thông tin người dùng và vé điện tử, chống giả mạo hiệu quả nhờ cơ chế RLS và mã QR duy nhất.
- Sản phẩm đã đạt trạng thái hoàn thiện, sẵn sàng để kiểm thử thực tế và triển khai lên các kho ứng dụng.

II. HẠN CHẾ

Mặc dù đã cố gắng hết sức, đồ án vẫn còn tồn tại một số hạn chế nhất định:

- Chưa triển khai phiên bản web và desktop dù Flutter hỗ trợ (hiện chỉ tập trung mobile).
- Tính năng thanh toán hiện mới chỉ tích hợp ZaloPay, chưa hỗ trợ đầy đủ các cổng thanh toán phổ biến khác như MoMo, VNPay, Stripe.
- Chưa có cơ chế thông báo đẩy (push notification) khi sự kiện sắp diễn ra hoặc khi có vé mới mở bán.
- Chưa triển khai tính năng gợi ý sự kiện bằng AI/ML hoặc tìm kiếm theo vị trí người dùng (geolocation).
- Hiệu suất khi có lượng lớn người dùng đồng thời truy cập cùng lúc (hàng chục nghìn) chưa được kiểm thử thực tế trên quy mô lớn.
- Giao diện quản trị Admin hiện chỉ là màn hình trong ứng dụng mobile, chưa có phiên bản web dashboard riêng biệt chuyên nghiệp.

III. HƯỚNG PHÁT TRIỂN

Dựa trên những hạn chế đã nhận diện, em xin đề xuất một số hướng phát triển trong tương lai để hoàn thiện và mở rộng sản phẩm EventGo:

- Mở rộng nền tảng: Phát triển thêm phiên bản Web và Desktop bằng cùng codebase Flutter để người tổ chức sự kiện có thể quản lý dễ dàng trên mọi thiết bị.
- Tích hợp thêm nhiều cổng thanh toán nội địa (MoMo, VNPay, ShopeePay) và quốc tế (Stripe, PayPal) để tăng tính tiện lợi cho người dùng.
- Xây dựng hệ thống push notification (Firebase Cloud Messaging) và thông báo trong ứng dụng để nhắc nhở sự kiện, mở bán vé mới.
- Phát triển dashboard quản trị web riêng chuyên nghiệp cho Admin và ban tổ chức sự kiện.
- Ứng dụng trí tuệ nhân tạo: Tích hợp gợi ý sự kiện cá nhân hóa dựa trên lịch sử tham gia và vị trí người dùng (geolocation + machine learning).
- Tối ưu hiệu suất và mở rộng quy mô bằng cách chuyển một phần logic nặng sang Cloud Functions hoặc Edge Functions của Supabase.

Triển khai chính thức lên Google Play Store và Apple App Store, kết hợp chiến lược marketing để đưa EventGo đến gần hơn với người dùng và các nhà tổ chức sự kiện tại Việt Nam.

TÀI LIỆU THAM KHẢO

- [1] <https://flutter.dev/>
- [2] <https://dart.dev/>
- [3] <https://firebase.google.com/>
- [4] <http://supabase.com/>
- [5] <https://firebase.flutter.dev/>