

ĐẠI HỌC TRÀ VINH

TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ



ĐỒ ÁN KẾT THÚC HỌC PHẦN

CÔNG NGHỆ PHẦN MỀM

TÊN ĐỀ TÀI

HỆ THỐNG ĐẶT PHÒNG KHÁCH SẠN TRỰC TUYẾN

Giáo viên hướng dẫn

Họ và tên: ThS. Nguyễn Bảo Ân

Sinh viên thực hiện:

Sơn Ngọc Tân 110122154

Trần Thị Yến Nhi 110122133

Đinh Hoài Phương 110122145

Mã lớp: DA22TTB

Trà Vinh, tháng 7 năm 2025

ĐẠI HỌC TRÀ VINH
TRƯỜNG KỸ THUẬT VÀ CÔNG NGHỆ



ĐỒ ÁN KẾT THÚC HỌC PHẦN
CÔNG NGHỆ PHẦN MỀM

TÊN ĐỀ TÀI

HỆ THỐNG
ĐẶT PHÒNG KHÁCH SẠN TRỰC TUYẾN

Giáo viên hướng dẫn

Họ và tên: ThS. Nguyễn Bảo Ân

Sinh viên thực hiện:

Sơn Ngọc Tân 110122154

Trần Thị Yến Nhi 110122133

Đinh Hoài Phương 110122145

Mã lớp: DA22TTB

NHẬN XÉT

Của giảng viên hướng dẫn

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Giảng viên hướng dẫn
(ký và ghi rõ họ tên)

NHẬN XÉT

Của giảng viên phản biện thứ nhất

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Giảng viên phản biện

MỤC LỤC

1. GIỚI THIỆU	1
1.1. Tên dự án và chủ đề	1
1.2. Mục tiêu của ứng dụng	1
1.3. Lý do chọn đề tài	2
2. PHÂN TÍCH YÊU CẦU	2
2.1. Các chức năng chính của hệ thống (Functional Requirements)	2
2.2. Các yêu cầu phi chức năng (Non-functional Requirements)	6
3. THIẾT KẾ HỆ THỐNG	7
3.1. Kiến trúc tổng thể	7
3.2. Thiết kế cơ sở dữ liệu	12
3.2.2. Mô tả chi tiết các bảng	13
3.3. Thiết kế API	18
3.4. Thiết kế giao diện (UI/UX)	22
4. TRIỂN KHAI VÀ CÔNG NGHỆ SỬ DỤNG	26
4.1. Danh sách công nghệ	26
4.2. Quy trình CI/CD với GitHub Actions	28
4.2. Cấu hình Docker và quy trình triển khai	29
4.3 Build và Deploy ứng dụng	31
5. QUẢN LÝ DỰ ÁN	34
5.1. Cách sử dụng Jira để lập kế hoạch và theo dõi tiến độ	34
5.2. Phân công nhiệm vụ của từng thành viên trong nhóm	36
6. KIỂM THỬ	37
6.1. Chiến lược kiểm thử và công cụ sử dụng	37
6.2. Kết quả kiểm thử API	42
7. ĐÁNH GIÁ VÀ KẾT LUẬN	43
7.1. Những khó khăn gặp phải trong quá trình thực hiện	43
7.2. Bài học rút ra và đề xuất cải thiện trong tương lai	44
8. PHỤ LỤC	47
8.1. Hướng dẫn cài đặt và chạy ứng dụng	47
8.2. Liên kết GitHub repository và link demo	52

DANH MỤC HÌNH ẢNH

Hình 1: Tạo EC2.....	31
Hình 2: Tạo một máy chủ Ubuntu ảo	32
Hình 3: Cài đặt Instance	32
Hình 4: Kiểm thử Postman đăng kí người dùng.....	38
Hình 5: Kiểm thử Postman đăng nhập người dùng	39
Hình 6: Kiểm thử Postman tạo phòng	39
Hình 7: Swagger UI người dùng	40
Hình 8: Swagger UI Khách sạn	40
Hình 9: Swagger UI của phòng	41
Hình 10: Swagger UI của đặt phòng	41
Hình 11: Swagger UI của thanh toán	41
Hình 12: UNIT TEST cho Frontend.....	42
Hình 13: UNIT TEST cho Backend	42

DANH MỤC BẢNG BIỂU

Bảng 1: Bảng User.....	14
Bảng 2: Bảng user.....	14
Bảng 3: Bảng rooms	15
Bảng 4: Bảng bookings	16
Bảng 5: Bảng payments.....	16
Bảng 6: Endpoint users.....	19
Bảng 7: Endpoint hotels	19
Bảng 8: Endpoint rooms	20
Bảng 9: Endpoint bookings	20
Bảng 10: Endpoint payments.....	20

1. GIỚI THIỆU

1.1. Tên dự án và chủ đề

Dự án "Hệ thống đặt phòng khách sạn trực tuyến" (Hotel Booking System) là một ứng dụng web được phát triển nhằm số hóa và tối ưu hóa quy trình đặt phòng khách sạn. Hệ thống được xây dựng với mục tiêu tạo ra một nền tảng kết nối giữa khách hàng có nhu cầu lưu trú và các khách sạn, đồng thời cung cấp công cụ quản lý hiệu quả cho nhà quản trị.

Chủ đề của dự án tập trung vào việc xây dựng một hệ thống quản lý đặt phòng khách sạn toàn diện, bao gồm các chức năng từ tìm kiếm, đặt phòng, thanh toán cho đến quản lý vận hành khách sạn. Hệ thống được thiết kế với kiến trúc microservices hiện đại, tách biệt hoàn toàn giữa frontend và backend, đảm bảo khả năng mở rộng và bảo trì dễ dàng.

1.2. Mục tiêu của ứng dụng

Ứng dụng được phát triển với các mục tiêu chính sau:

Đối với khách hàng:

- Cung cấp giao diện thân thiện, dễ sử dụng để tìm kiếm và đặt phòng khách sạn
- Cho phép xem thông tin chi tiết về khách sạn, phòng, tiện nghi và giá cả
- Hỗ trợ quản lý lịch sử đặt phòng và thông tin cá nhân
- Đảm bảo quy trình thanh toán an toàn, tiện lợi

Đối với quản trị viên:

- Cung cấp dashboard quản lý tổng quan với các thống kê quan trọng
- Cho phép quản lý thông tin khách sạn, phòng, giá cả một cách linh hoạt
- Theo dõi và xử lý các đơn đặt phòng, thanh toán
- Hỗ trợ báo cáo doanh thu và phân tích kinh doanh

Về mặt kỹ thuật:

- Xây dựng hệ thống với kiến trúc RESTful API chuẩn

- Đảm bảo tính bảo mật cao với JWT authentication
- Tối ưu hiệu năng và khả năng mở rộng
- Áp dụng các công nghệ hiện đại trong phát triển web

1.3. Lý do chọn đề tài

Việc lựa chọn đề tài "Hệ thống đặt phòng khách sạn trực tuyến" xuất phát từ nhiều lý do thực tiễn:

1.Nhu cầu thực tế của thị trường:

Ngành du lịch và khách sạn đang phát triển mạnh mẽ tại Việt Nam và trên thế giới. Việc số hóa quy trình đặt phòng không chỉ là xu hướng mà đã trở thành nhu cầu thiết yếu của cả khách hàng lẫn nhà cung cấp dịch vụ.

2. Cơ hội học tập và phát triển kỹ năng:

Đề tài cho phép áp dụng đa dạng các công nghệ web hiện đại. Tìm hiểu về kiến trúc microservices và RESTful API. Thực hành xây dựng hệ thống với nhiều vai trò người dùng (role-based access control). Làm việc với các vấn đề thực tế như xác thực, phân quyền, xử lý đồng thời.

3. Tính phức tạp phù hợp:

Hệ thống đặt phòng khách sạn có độ phức tạp vừa phải, bao gồm nhiều module chức năng khác nhau như quản lý người dùng, khách sạn, phòng, đặt phòng, thanh toán. Điều này giúp sinh viên có cơ hội thực hành toàn diện các kỹ năng lập trình web.

4. Khả năng mở rộng và phát triển:

Dự án có thể được mở rộng với nhiều tính năng nâng cao như tích hợp bản đồ, đánh giá khách sạn, hệ thống khuyến mãi, chatbot hỗ trợ... tạo nền tảng cho việc phát triển lâu dài.

2. PHÂN TÍCH YÊU CẦU

2.1. Các chức năng chính của hệ thống (Functional Requirements)

Hệ thống đặt phòng khách sạn được phân tích và thiết kế với các nhóm chức năng chính sau:

2.1.1. Quản lý người dùng và xác thực

Đăng ký tài khoản:

- Người dùng có thể đăng ký tài khoản mới với các thông tin: email, username, mật khẩu, họ tên, số điện thoại
- Hệ thống kiểm tra tính hợp lệ của email và username không trùng lặp
- Mật khẩu được mã hóa bằng thuật toán bcrypt trước khi lưu vào database

Đăng nhập và xác thực:

- Đăng nhập bằng username/password
- Hệ thống cấp JWT token cho phiên làm việc
- Token có thời hạn và được refresh tự động
- Hỗ trợ phân quyền Admin và Guest (khách hàng)

Quản lý thông tin cá nhân:

- Xem và cập nhật thông tin profile
- Thay đổi mật khẩu với xác thực mật khẩu cũ
- Xem lịch sử đặt phòng cá nhân

2.1.2. Quản lý khách sạn

Dành cho Admin:

- Thêm mới khách sạn với đầy đủ thông tin: tên, mô tả, địa chỉ, thành phố, quốc gia, điện thoại, email, website
- Cập nhật thông tin khách sạn
- Xóa khách sạn (kiểm tra ràng buộc với phòng và booking)
- Quản lý danh sách tiện nghi của khách sạn
- Upload và quản lý hình ảnh khách sạn thông qua Google Drive API

Dành cho khách hàng:

- Xem danh sách khách sạn với phân trang
- Tìm kiếm khách sạn theo thành phố
- Lọc khách sạn theo đánh giá sao (1-5 sao)
- Xem chi tiết thông tin khách sạn và hình ảnh

2.1.3. Quản lý phòng**Dành cho Admin:**

- Thêm phòng mới cho khách sạn với thông tin: số phòng, loại phòng, sức chứa, giá/đêm, mô tả, tiện nghi
- Cập nhật thông tin phòng
- Quản lý trạng thái phòng (available/maintenance)
- Xóa phòng (kiểm tra ràng buộc với booking)
- Upload hình ảnh phòng

Dành cho khách hàng:

- Xem danh sách phòng của khách sạn
- Lọc phòng theo: loại phòng (single, double, suite, deluxe), khoảng giá, sức chứa
- Kiểm tra phòng trống theo ngày check-in/check-out
- Xem chi tiết phòng với hình ảnh và tiện nghi

2.1.4. Quản lý đặt phòng (Booking)**Tạo booking:**

Khách hàng chọn phòng và nhập thông tin: ngày check-in/out, số lượng khách, yêu cầu đặc biệt. Hệ thống kiểm tra tính khả dụng của phòng trong khoảng thời gian đã chọn. Tự động tính tổng số đêm và tổng tiền. Tạo mã booking reference duy nhất.

Quản lý booking:

- Xem danh sách booking cá nhân (khách hàng)

- Xem tất cả booking (admin)
- Cập nhật thông tin booking (chỉ booking đang pending)
- Hủy booking (trước ngày check-in)
- Admin có thể xác nhận hoặc từ chối booking

Trạng thái booking:

- PENDING: Chờ xác nhận
- CONFIRMED: Đã xác nhận
- CANCELLED: Đã hủy
- COMPLETED: Đã hoàn thành

2.1.5. Quản lý thanh toán

Tạo thanh toán:

- Khách hàng tạo thanh toán cho booking đã được xác nhận
- Hỗ trợ nhiều phương thức: Credit Card, Bank Transfer, Cash, PayPal, MoMo
- Lưu thông tin giao dịch và trạng thái

Xử lý thanh toán:

- Admin xác nhận thanh toán đã nhận
- Cập nhật trạng thái: PENDING → COMPLETED/FAILED
- Hỗ trợ hoàn tiền (refund) khi cần

2.1.6. Dashboard và báo cáo (Admin)

Dashboard tổng quan:

- Thống kê số lượng: người dùng, khách sạn, phòng, booking
- Doanh thu theo thời gian
- Booking sắp tới và khách đang lưu trú

Báo cáo chi tiết:

- Báo cáo doanh thu theo khách sạn

- Thống kê tỷ lệ lấp đầy phòng
- Danh sách booking theo trạng thái
- Export báo cáo (tính năng mở rộng)

2.2. Các yêu cầu phi chức năng (Non-functional Requirements)

2.2.1. Hiệu năng (Performance)

1. Thời gian phản hồi: API response time < 200ms cho các request thông thường
2. Số lượng người dùng đồng thời: Hỗ trợ tối thiểu 100+ concurrent users
3. Tối ưu database: Sử dụng indexing cho các trường tìm kiếm thường xuyên
4. Caching: Implement caching cho dữ liệu ít thay đổi
5. Lazy loading: Áp dụng cho hình ảnh và danh sách dài

2.2.2. Bảo mật (Security)

- Xác thực và phân quyền: JWT token với expiration time
- Mã hóa mật khẩu: Sử dụng bcrypt với salt rounds phù hợp
- CORS policy: Cấu hình chặt chẽ cho production
- Input validation: Kiểm tra và sanitize tất cả input từ người dùng
- SQL injection prevention: Sử dụng ORM và parameterized queries
- HTTPS: Bắt buộc sử dụng SSL/TLS cho production

2.2.3. Khả năng sử dụng (Usability)

- Giao diện người dùng: Thiết kế responsive, hỗ trợ mobile/tablet/desktop
- Ngôn ngữ: Giao diện tiếng Việt thân thiện
- Navigation: Menu và flow rõ ràng, dễ hiểu
- Error handling: Thông báo lỗi rõ ràng, hướng dẫn khắc phục
- Loading states: Hiển thị trạng thái loading cho mọi async operation

2.2.4. Độ tin cậy (Reliability)

- Uptime: Target 99.9% availability

- Backup: Sao lưu database định kỳ
- Error recovery: Xử lý graceful degradation khi có lỗi
- Transaction integrity: Đảm bảo tính toàn vẹn của giao dịch booking/payment

2.2.5. Khả năng mở rộng (Scalability)

- Kiến trúc microservices: Frontend/Backend tách biệt
- Horizontal scaling: Có thể scale các container độc lập
- Database connection pooling: Quản lý connection hiệu quả
- Stateless design: Backend không lưu state, dễ dàng scale

2.2.6. Khả năng bảo trì (Maintainability)

- Code structure: Tổ chức code theo modules/layers rõ ràng
- Documentation: API documentation tự động với Swagger
- Coding standards: Tuân thủ PEP 8 (Python) và ESLint (JavaScript)
- Version control: Sử dụng Git với branching strategy
- Environment configuration: Quản lý config qua environment variables

2.2.7. Tính tương thích (Compatibility)

- Browser support: Chrome, Firefox, Safari, Edge phiên bản mới nhất
- API versioning: Hỗ trợ versioning cho backward compatibility
- Database: MySQL 8.0+
- Container runtime: Docker 20.10+

3. THIẾT KẾ HỆ THỐNG

3.1. Kiến trúc tổng thể

Hệ thống đặt phòng khách sạn được thiết kế theo kiến trúc microservices hiện đại, với sự tách biệt rõ ràng giữa các thành phần. Kiến trúc này mang lại nhiều lợi ích về khả năng mở rộng, bảo trì và phát triển độc lập các module.

3.1.1. Tổng quan kiến trúc

Hệ thống được chia thành 3 tầng chính:

1. Tầng Presentation (Frontend)

- Công nghệ: React với TypeScript
- Chức năng: Giao diện người dùng, xử lý tương tác
- Giao tiếp với Backend qua RESTful API
- Deployment: Container riêng biệt với Nginx

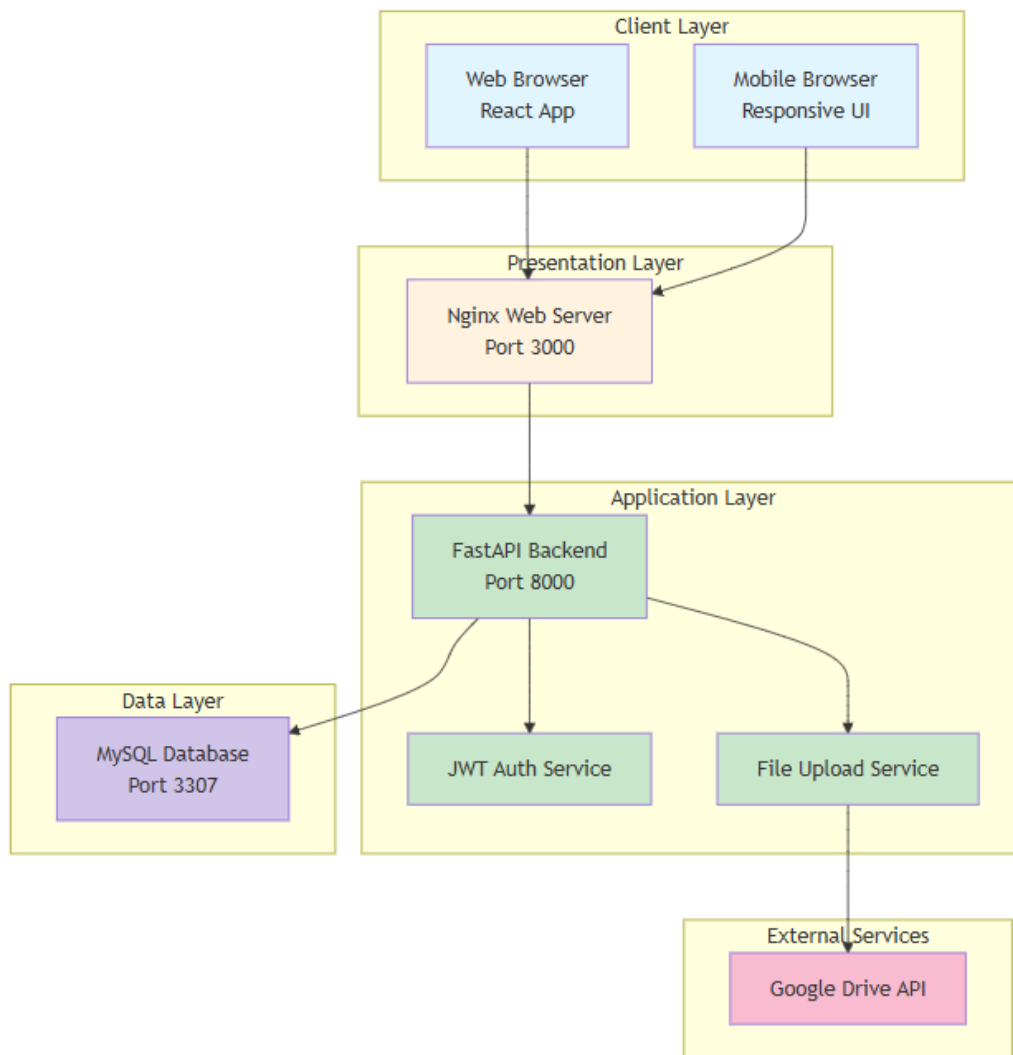
2. Tầng Business Logic (Backend)

- Công nghệ: FastAPI (Python)
- Chức năng: Xử lý business logic, authentication, authorization
- Cung cấp RESTful API endpoints
- Tích hợp với các service bên ngoài (Google Drive)

3. Tầng Data (Database)

- Công nghệ: MySQL 8.0
- Chức năng: Lưu trữ và quản lý dữ liệu
- Sử dụng ORM (SQLAlchemy) để tương tác

3.1.2. Sơ đồ kiến trúc hệ thống



Hình 1: Sơ đồ kiến trúc hệ thống

3.1.3. Luồng xử lý request

1. User Request Flow:

- Người dùng gửi request từ browser
- Request được Nginx (frontend container) tiếp nhận
- Nếu là static files (HTML, CSS, JS) → Nginx trả về trực tiếp
- Nếu là API call → Forward đến Backend

2. API Processing Flow:

- FastAPI nhận request tại endpoint tương ứng
- Middleware kiểm tra CORS và authentication

- JWT token được verify (nếu là protected route)
- Business logic được thực thi trong service layer
- Database query thông qua SQLAlchemy ORM
- Response được format và trả về client

3. File Upload Flow:

- Client upload file qua multipart/form-data
- Backend validate file (type, size)
- File được upload lên Google Drive
- URL được lưu trong database
- Public URL trả về cho client

3.1.4. Các thành phần chính

Frontend Components:

- Các trang: Trang chủ, Trang đăng nhập, Trang đăng ký, Trang hồ sơ, Bảng điều khiển quản trị, Trang phòng, Trang đặt phòng
- Thành phần: Thanh điều hướng, Chân trang, Đường dẫn được bảo vệ, Mô-đun quản trị
- Dịch vụ: Dịch vụ API với bộ chặn Axios
- Bối cảnh: AuthContext cho quản lý trạng thái
- Kiểu: Giao diện TypeScript cho an toàn kiểu

Backend Modules:

- Routers: người dùng, khách sạn, phòng, đặt phòng, thanh toán
- Services: Lớp logic nghiệp vụ cho mỗi mô-đun
- Models: Mô hình ORM SQLAlchemy
- Schemas: Mô hình Pydantic cho xác thực
- Auth: Quản lý mã thông báo JWT
- Utils: Hàm trợ giúp (tích hợp Google Drive)

Database Schema:

- users: Thông tin người dùng và authentication
- hotels: Thông tin khách sạn
- rooms: Phòng của từng khách sạn
- bookings: Thông tin đặt phòng
- payments: Giao dịch thanh toán

3.1.5. Deployment Architecture

Hệ thống sử dụng Docker Compose để orchestrate các services:

Container Configuration:

- **mysql:** MySQL 8.0 database container
- **backend:** Python FastAPI application
- **frontend:** React app served by Nginx

Network Configuration:

- Tất cả containers trong cùng một Docker network
- Giao tiếp nội bộ thông qua tên container
- Truy cập bên ngoài thông qua cổng mở

Volume Management:

- **mysql_data:** Lưu trữ liên tục cho cơ sở dữ liệu
- **media_files:** Lưu trữ cho các tệp đã tải lên
- **Config files:** Được gắn kết từ hệ thống máy chủ

3.1.6. Ưu điểm của kiến trúc

1. Tách biệt Frontend/Backend:

- Phát triển độc lập
- Technology agnostic
- Dễ dàng scale riêng từng phần

2. RESTful API Design:

- Chuẩn hóa communication
- Dễ dàng tích hợp với mobile app
- Self-documenting với Swagger

3. Container-based Deployment:

- Tính nhất quán của môi trường
- Triển khai và khôi phục dễ dàng
- Cô lập tài nguyên

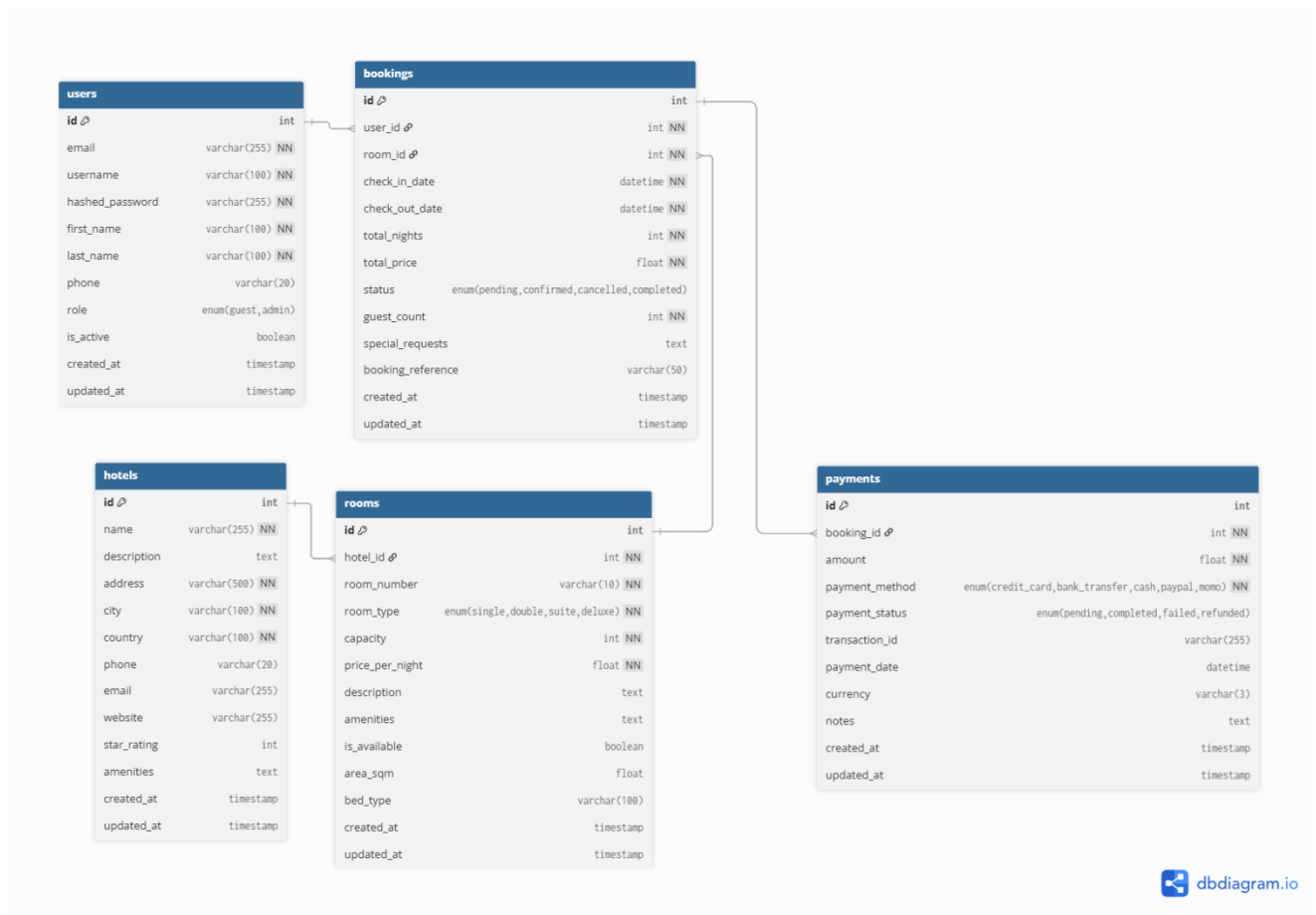
4. Microservices Benefits:

- Độ gắn kết cao, độ liên kết thấp
- Khả năng mở rộng độc lập
- Đa dạng công nghệ có thể

3.2. Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu của hệ thống được thiết kế theo mô hình quan hệ (Relational Database) với 5 bảng chính. Thiết kế này đảm bảo tính toàn vẹn dữ liệu, hiệu quả truy vấn và khả năng mở rộng.

3.2.1. Sơ đồ ERD (Entity Relationship Diagram)



Hình 2: Mô hình ERD

3.2.2. Mô tả chi tiết các bảng

1. Bảng **users** (Người dùng)

Bảng này lưu trữ thông tin của tất cả người dùng hệ thống, bao gồm khách hàng và quản trị viên.

Tên trường	Kiểu dữ liệu	Mô tả
id	INT, PK, AUTO_INCREMENT	Khóa chính
email	VARCHAR(255), UNIQUE, NOT NULL	Email đăng nhập
username	VARCHAR(100), UNIQUE, NOT NULL	Tên đăng nhập
hashed_password	VARCHAR(255), NOT NULL	Mật khẩu đã mã hóa
first name	VARCHAR(100), NOT NULL	Tên

Tên trường	Kiểu dữ liệu	Mô tả
last_name	VARCHAR(100), NOT NULL	Họ
phone	VARCHAR(20)	Số điện thoại
role	ENUM('guest', 'admin')	Vai trò người dùng
is_active	BOOLEAN	Trạng thái hoạt động
created_at	TIMESTAMP	Thời gian tạo
updated_at	TIMESTAMP	Thời gian cập nhật

Bảng 1: Bảng User

2. Bảng hotels (Khách sạn)

Lưu trữ thông tin của các khách sạn trong hệ thống.

Tên trường	Kiểu dữ liệu	Mô tả
id	INT, PK, AUTO_INCREMENT	Khóa chính
name	VARCHAR(255), NOT NULL	Tên khách sạn
description	TEXT	Mô tả chi tiết
address	VARCHAR(500), NOT NULL	Địa chỉ
city	VARCHAR(100), NOT NULL	Thành phố
country	VARCHAR(100), NOT NULL	Quốc gia
phone	VARCHAR(20)	Số điện thoại
email	VARCHAR(255)	Email liên hệ
website	VARCHAR(255)	Website
star_rating	INT	Đánh giá sao (1-5)
amenities	TEXT	Tiện nghi (dạng JSON)
created_at	TIMESTAMP	Thời gian tạo
updated_at	TIMESTAMP	Thời gian cập nhật

Bảng 2: Bảng user

3. Bảng rooms (Phòng)

Lưu trữ thông tin các phòng của từng khách sạn.

Tên trường	Kiểu dữ liệu	Mô tả
id	INT, PK, AUTO_INCREMENT	Khóa chính
hotel_id	INT, FK, NOT NULL	Khóa ngoại tham chiếu hotels
room_number	VARCHAR(10), NOT NULL	Số phòng
room_type	ENUM	Loại phòng
capacity	INT, NOT NULL	Sức chứa
price_per_night	FLOAT, NOT NULL	Giá mỗi đêm
description	TEXT	Mô tả chi tiết
amenities	TEXT	Tiện nghi phòng
is_available	BOOLEAN	Trạng thái sẵn sàng
area_sqm	FLOAT	Diện tích (m ²)
bed_type	VARCHAR(100)	Loại giường
created_at	TIMESTAMP	Thời gian tạo
updated_at	TIMESTAMP	Thời gian cập nhật

Bảng 3: Bảng rooms

4. Bảng bookings (Đặt phòng)

Lưu trữ thông tin các đơn đặt phòng.

Tên trường	Kiểu dữ liệu	Mô tả
id	INT, PK, AUTO_INCREMENT	Khóa chính
user_id	INT, FK, NOT NULL	Khóa ngoại tham chiếu users
room_id	INT, FK, NOT NULL	Khóa ngoại tham chiếu rooms
check_in_date	DATETIME, NOT NULL	Ngày nhận phòng
check_out_date	DATETIME, NOT NULL	Ngày trả phòng
total_nights	INT, NOT NULL	Tổng số đêm
total_price	FLOAT, NOT NULL	Tổng tiền
status	ENUM	Trạng thái booking
guest_count	INT, NOT NULL	Số lượng khách

Tên trường	Kiểu dữ liệu	Mô tả
special_requests	TEXT	Yêu cầu đặc biệt
booking_reference	VARCHAR(50), UNIQUE	Mã booking
created_at	TIMESTAMP	Thời gian tạo
updated_at	TIMESTAMP	Thời gian cập nhật

Bảng 4: Bảng bookings

5. Bảng payments (Thanh toán)

Lưu trữ thông tin các giao dịch thanh toán.

Tên trường	Kiểu dữ liệu	Mô tả
id	INT, PK, AUTO_INCREMENT	Khóa chính
booking_id	INT, FK, NOT NULL	Khóa ngoại tham chiếu bookings
amount	FLOAT, NOT NULL	Số tiền
payment_method	ENUM	Phương thức thanh toán
payment_status	ENUM	Trạng thái thanh toán
transaction_id	VARCHAR(255), UNIQUE	Mã giao dịch
payment_date	DATETIME	Ngày thanh toán
currency	VARCHAR(3)	Loại tiền tệ
notes	TEXT	Ghi chú
created_at	TIMESTAMP	Thời gian tạo
updated_at	TIMESTAMP	Thời gian cập nhật

Bảng 5: Bảng payments

3.2.3. Quan hệ giữa các bảng

1. Hotel - Room (1:N)

- Một khách sạn có nhiều phòng
- Khi xóa khách sạn, tất cả phòng thuộc khách sạn đó cũng bị xóa (CASCADE DELETE)

2. User - Booking (1:N)

- Một người dùng có thể có nhiều booking
- Không cho phép xóa user nếu còn booking liên quan

3. Room - Booking (1:N)

- Một phòng có thể có nhiều booking (ở các thời điểm khác nhau)
- Kiểm tra overlap khi tạo booking mới

4. Booking - Payment (1:N)

- Một booking có thể có nhiều payment (deposit, final payment, refund)
- Khi xóa booking, các payment liên quan cũng bị xóa

3.2.4. Các ràng buộc và chỉ mục

Ràng buộc duy nhất (UNIQUE):

- users.email
- users.username
- bookings.booking_reference
- payments.transaction_id

Chỉ mục (INDEX):

- hotels.city (để tìm kiếm nhanh theo thành phố)
- rooms.hotel_id (foreign key index)
- bookings.user_id (foreign key index)
- bookings.room_id (foreign key index)
- bookings.check_in_date, bookings.check_out_date (để kiểm tra availability)

Check constraints:

- check_out_date > check_in_date

- price_per_night > 0
- capacity > 0
- star_rating BETWEEN 1 AND 5

3.3. Thiết kế API

Hệ thống sử dụng RESTful API với chuẩn thiết kế hiện đại, tuân thủ các nguyên tắc REST và cung cấp documentation tự động thông qua Swagger/OpenAPI.

3.3.1. Cấu trúc URL và versioning

Base URL Pattern:

```
http://[domain]/api/v1/[resource]
```

Ví dụ:

```
http://localhost:8000/api/v1/users
```

```
http://localhost:8000/api/v1/hotels
```

```
http://localhost:8000/api/v1/bookings
```

3.3.2. Các endpoint chính

1. Authentication & Users

Method	Endpoint	Mô tả	Auth Required
POST	/users/register	Đăng ký tài khoản mới	No
POST	/users/login	Đăng nhập	No
GET	/users/me	Lấy thông tin profile	Yes
PUT	/users/me	Cập nhật profile	Yes
POST	/users/change-password	Đổi mật khẩu	Yes
GET	/users	Danh sách users (Admin)	Admin

Method	Endpoint	Mô tả	Auth Required
DELETE	/users/{id}	Xóa user (Admin)	Admin

Bảng 6: Endpoint users

2. Hotels Management

Method	Endpoint	Mô tả	Auth Required
GET	/hotels	Danh sách khách sạn	No
GET	/hotels/{id}	Chi tiết khách sạn	No
POST	/hotels	Tạo khách sạn mới	Admin
PUT	/hotels/{id}	Cập nhật khách sạn	Admin
DELETE	/hotels/{id}	Xóa khách sạn	Admin
POST	/hotels/{id}/images	Upload ảnh khách sạn	Admin

Bảng 7: Endpoint hotels

3. Rooms Management

Method	Endpoint	Mô tả	Auth Required
GET	/rooms	Danh sách phòng	No
GET	/rooms/{id}	Chi tiết phòng	No
POST	/rooms	Tạo phòng mới	Admin
PUT	/rooms/{id}	Cập nhật phòng	Admin
DELETE	/rooms/{id}	Xóa phòng	Admin
GET	/rooms/{id}/availability	Kiểm tra phòng trống	No

Method	Endpoint	Mô tả	Auth Required
POST	/rooms/{id}/images	Upload ảnh phòng	Admin

Bảng 8: Endpoint rooms

4. Bookings Management

Method	Endpoint	Mô tả	Auth Required
GET	/bookings	Danh sách booking	Yes
GET	/bookings/{id}	Chi tiết booking	Yes
POST	/bookings	Tạo booking mới	Yes
PUT	/bookings/{id}	Cập nhật booking	Yes
POST	/bookings/{id}/cancel	Hủy booking	Yes
POST	/bookings/{id}/confirm	Xác nhận booking	Admin
GET	/bookings/my-bookings	Booking của tôi	Yes

Bảng 9: Endpoint bookings

5. Payments Management

Method	Endpoint	Mô tả	Auth Required
GET	/payments	Danh sách thanh toán	Yes
GET	/payments/{id}	Chi tiết thanh toán	Yes
POST	/payments	Tạo thanh toán	Yes
POST	/payments/{id}/process	Xử lý thanh toán	Admin
POST	/payments/{id}/refund	Hoàn tiền	Admin

Bảng 10: Endpoint payments

3.3.3. Request/Response Format

Standard Response Format:

```
{
  "code": 200,
  "message": "Thành công",
  "data": {
    // Response data here
  }
}
```

Error Response Format:

```
{
  "code": 400,
  "message": "Lỗi validation",
  "errors": {
    "field_name": ["Error message"]
  }
}
```

Pagination Response:

```
{
  "code": 200,
  "message": "Thành công",
  "data": {
    "items": [...],
    "total": 100,
    "page": 1,
    "per_page": 20,
    "total_pages": 5
  }
}
```

3.3.4. Authentication & Authorization

JWT Token Structure:

Header: Algorithm và token type

Payload: User ID, username, role, expiration

Signature: Đảm bảo tính toàn vẹn

Authorization Header:

Authorization: Bearer <jwt_token>

Role-based Access:

Guest: Xem thông tin, đặt phòng cá nhân

Admin: Full access cho quản lý hệ thống

3.3.5. API Documentation

Hệ thống tự động generate API documentation thông qua FastAPI:

Swagger UI: /docs

ReDoc: /redoc

OpenAPI schema: /openapi.json

3.4. Thiết kế giao diện (UI/UX)

Giao diện được thiết kế theo phong cách hiện đại, tối giản và thân thiện với người dùng Việt Nam.

3.4.1. Design System

Color Palette:

- + Primary: #1976D2 (Blue)
- + Secondary: #FFA726 (Orange)
- + Success: #4CAF50 (Green)
- + Error: #F44336 (Red)
- + Background: #F5F5F5 (Light Gray)
- + Text: #212121 (Dark Gray)

Typography:

- + Font Family: Inter, system-ui
- + Heading: Bold, sizes từ 24px-48px
- + Body: Regular, 16px
- + Caption: Regular, 14px

Spacing System:

Base unit: 8px

Spacing scale: 8, 16, 24, 32, 48, 64px

3.4.2. Cấu trúc màn hình

1. Layout chung:

Header với navigation

Main content area

Footer với thông tin liên hệ

2. Responsive breakpoints:

Mobile: < 768px

Tablet: 768px - 1024px

Desktop: > 1024px

3.4.3. Danh sách màn hình chính

1. Trang chủ (HomePage)

- Hero section với search form
- Danh sách khách sạn nổi bật
- Danh sách phòng được đặt nhiều
- Call-to-action sections

2. Trang đăng nhập/dăng ký

- Form đăng nhập với validation
- Link chuyển sang đăng ký
- Forgot password option
- Social login buttons (future)

3. Trang danh sách khách sạn

- Thanh bên lọc (thành phố, đánh giá, giá)

- Chuyển đổi chế độ xem dạng lưới/danh sách
- Phân trang
- Chế độ xem nhanh

4. Trang chi tiết khách sạn

- Gallery với slideshow
- Thông tin chi tiết và tiện nghi
- Danh sách phòng available
- Map integration (future)

5. Trang đặt phòng

- Booking form với date picker
- Guest information
- Special requests
- Price summary
- Payment options

6. Trang quản lý booking

- Tabs cho các trạng thái booking
- Actions: View, Cancel, Pay
- Export functionality
- Search và filter

7. Admin Dashboard

- Thẻ thống kê
- Biểu đồ doanh thu, công suất phòng
- Bảng đặt phòng gần đây
- Menu thao tác nhanh

8. Trang quản lý (Admin)

- Bảng CRUD cho mỗi thực thể

- Biểu mẫu modal cho tạo/chỉnh sửa
- Hành động hàng loạt
- Nhập/Xuất dữ liệu

3.4.4. User Experience (UX) Guidelines

1. Navigation:

- Đường dẫn cho các trang sâu
- Xóa nút quay lại
- Tiêu đề cố định trên thiết bị di động
- Thanh tìm kiếm nổi bật

2. Forms:

- Xác thực nội tuyến
- Xóa thông báo lỗi
- Tự động lưu bản nháp
- Chỉ báo tiến trình

3. Feedback:

- Đang tải trạng thái cho mọi hành động
- Thông báo thành công/lỗi
- Hộp thoại xác nhận
- Trạng thái trống rỗng với hướng dẫn

4. Accessibility:

- Nhãn ARIA
- Điều hướng bằng bàn phím
- Chế độ tương phản cao
- Hỗ trợ trình đọc màn hình

3.4.5. Thiết kế ưu tiên thiết bị di động

Tối ưu hóa cho thiết bị di động:

Các nút cảm ứng (tối thiểu 44px)

Cử chỉ vuốt để xem thư viện

Điều hướng dưới cùng để xem hành động

Biểu mẫu được đơn giản hóa

Khả năng ngoại tuyến (tương lai của PWA)

4. TRIỂN KHAI VÀ CÔNG NGHỆ SỬ DỤNG

4.1. Danh sách công nghệ

Dự án sử dụng stack công nghệ hiện đại, được lựa chọn kỹ càng để đảm bảo hiệu năng, khả năng mở rộng và dễ bảo trì.

4.1.1. Công nghệ Backend

Framework chính:

- **FastAPI 0.104.1:** Framework Python hiện đại cho việc xây dựng API
 - + Tự động generate API documentation
 - + Type hints và validation với Pydantic
 - + Async/await support
 - + High performance

Database & ORM:

- **MySQL 8.0:** Hệ quản trị cơ sở dữ liệu quan hệ
 - + Reliability và data integrity
 - + Support cho complex queries
 - + Good performance với indexing
- **SQLAlchemy 2.0:** Python ORM
 - + Database abstraction layer
 - + Migration support

- + Relationship mapping

Xác thực và Bảo mật:

PyJWT: Triển khai JSON Web Token

Passlib[bcrypt]: Password hashing

python-jose: Triển khai JOSE cho JWT

cryptography: Công thức mã hóa

File Storage:

- + **Google Drive API:** Giải pháp lưu trữ đám mây

- google-api-python-client
- thư viện google-auth
- Tiềm năng lưu trữ không giới hạn

Other Backend Libraries:

- **uvicorn:** Máy chủ ASGI
- **python-multipart:** Phân tích dữ liệu biểu mẫu
- **python-dotenv:** Quản lý môi trường
- **email-validator:** Xác thực email

4.1.2. Công nghệ frontend

Core Framework:

React 18.2.0: Thư viện JavaScript cho UI

- Kiến trúc dựa trên thành phần
- Virtual DOM
- Hệ sinh thái lớn

TypeScript 4.9.5: JavaScript an toàn về kiểu

- Kiểu tĩnh
- Hỗ trợ IDE tốt hơn
- Ít lỗi thời gian chạy hơn

Routing & State:

React Router v6: Định tuyến phía máy khách

React Context API: Quản lý trạng thái

React Query: Quản lý trạng thái máy chủ

4.1.3. DevOps & Cơ sở hạ tầng

Containerization:

- Docker: Container hóa ứng dụng
- Docker Compose: Phối hợp đa container
- Nginx: Máy chủ web cho giao diện người dùng

Công cụ phát triển:

- Git: Kiểm soát phiên bản
- ESLint: Kiểm tra lỗi JavaScript
- Prettier: Định dạng mã

4.2. Quy trình CI/CD với GitHub Actions

Hiện tại dự án chưa implement CI/CD với GitHub Actions

4.2.1. Quản lý môi trường

Development:

- Phát triển cục bộ với Docker Compose
- Bật tính năng tải lại nóng
- Bật chế độ gỡ lỗi

Staging:

- Phản chiếu phiên bản production
- Kiểm thử với dữ liệu thực (ẩn danh)
- Kiểm thử hiệu năng

Production:

- Tối ưu hóa bản dựng
- Hỗ trợ SSL/TLS
- Giám sát và ghi nhật ký

4.2. Cấu hình Docker và quy trình triển khai

4.2.1. Docker Architecture

1. Service Containers:

MySQL Container:

- Image: mysql:8.0
- Port mapping: 3307:3306
- Volume: mysql_data
- Health check configured
- Auto-restart policy
-

Backend Container:

- Hình ảnh tùy chỉnh từ Python 3.11-slim
- Ánh xạ cổng: 8000:8000
- Gắn kết ổ đĩa cho mã và phương tiện
- Tiêm biến môi trường
- Kiểm tra tình trạng điểm cuối

Frontend Container:

- Xây dựng nhiều giai đoạn
- Nginx phục vụ các tệp tĩnh

- Cổng: 3000:80
- Tối ưu hóa cho môi trường production

4.2.2. Cấu hình Docker Compose

Services orchestration:

```
version: '3.8'
services:
  mysql:
    # Database configuration
    # Persistent volume
    # Network isolation

  backend:
    # Depends on mysql
    # Environment configuration
    # Code hot-reload

  frontend:
    # Depends on backend
    # Build args for API URL
    # Nginx configuration
```

4.2.3. Deployment Process

1. Development Deployment:

```
# Sao chép kho lưu trữ
git clone [repository]

# Thiết lập môi trường
cp .env.example .env

# Chỉnh sửa tệp .env

# Khởi động dịch vụ
docker-compose up --build

# Truy cập ứng dụng

# Giao diện người dùng: http://localhost:3000

# Giao diện người dùng: http://localhost:8000

# Tài liệu API: http://localhost:8000/docs
```

2. Triển khai sản xuất:

Các bước triển khai:

Kéo mã mới nhất

```
git pull origin main
```

Xây dựng ảnh production

```
docker-compose -f docker-compose.prod.yml build
```

Dừng các container cũ

```
docker-compose -f docker-compose.prod.yml down
```

Khởi động các container mới

```
docker-compose -f docker-compose.prod.yml up -d
```

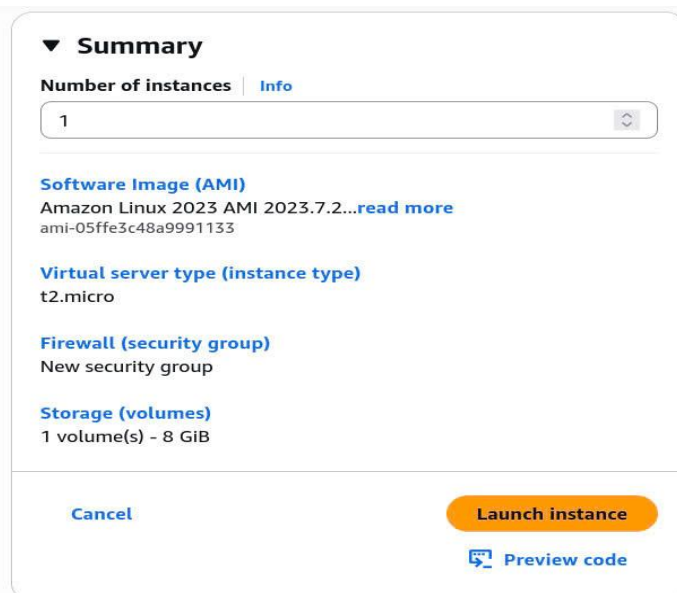
Kiểm tra nhật ký

```
docker-compose -f docker-compose.prod.yml logs -f
```

4.3 Build và Deploy ứng dụng

Cài đặt môi trường EC2 trên AWS

Tạo EC2 Instance:



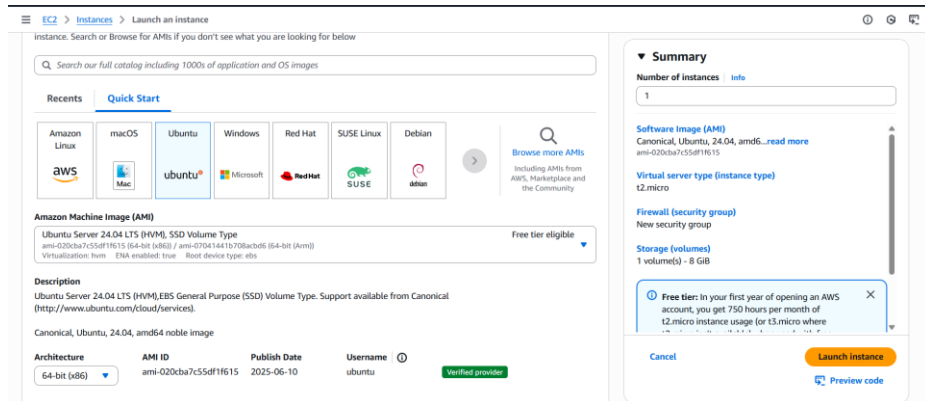
The screenshot shows the 'Summary' tab of the AWS Management Console for creating a new EC2 instance. The 'Number of instances' is set to 1. The 'Software Image (AMI)' is 'Amazon Linux 2023 AMI 2023.7.2...' with ID 'ami-05ffe3c48a9991133'. The 'Virtual server type (instance type)' is 't2.micro'. The 'Firewall (security group)' is 'New security group'. The 'Storage (volumes)' section shows '1 volume(s) - 8 GiB'. At the bottom, there are 'Cancel' and 'Launch instance' buttons, along with a 'Preview code' link.

Field	Value
Number of instances	1
Software Image (AMI)	Amazon Linux 2023 AMI 2023.7.2... ami-05ffe3c48a9991133
Virtual server type (instance type)	t2.micro
Firewall (security group)	New security group
Storage (volumes)	1 volume(s) - 8 GiB

Hình 1: Tạo EC2

Bước đầu tiên là tạo một EC2 instance trên AWS:

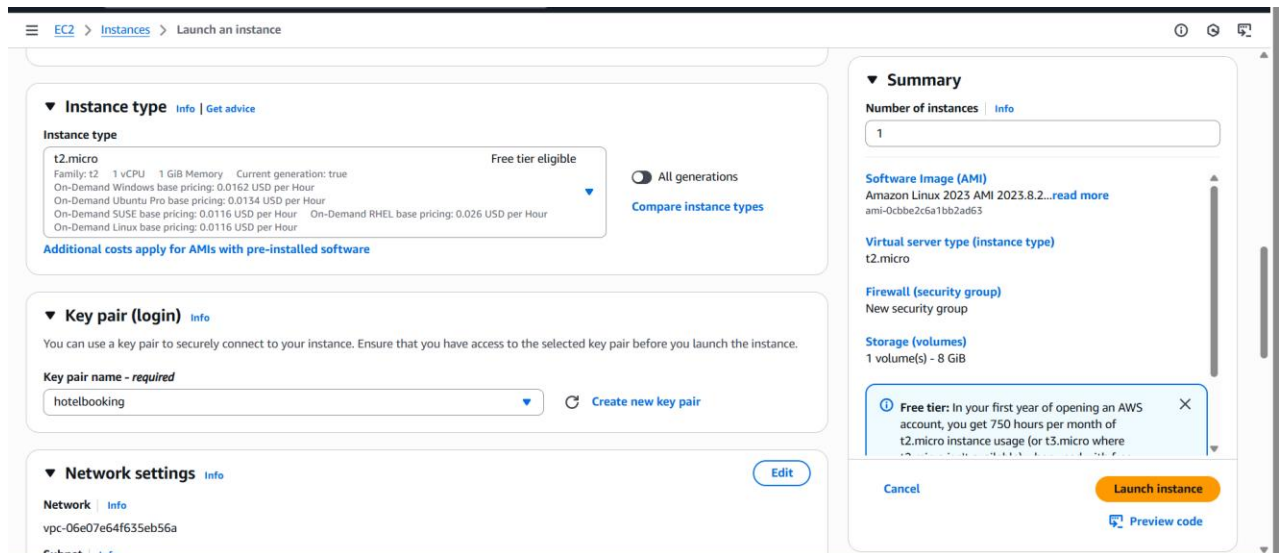
1. Đăng nhập AWS Console



Hình 2: Tạo một máy chủ Ubuntu ảo

2. Launch Instance:

- + AMI: Amazon Linux 2
- + Instance Type: t2.micro (Free tier eligible)
- + Key Pair: tạo key pair hotelbooking



Hình 3: Cài đặt Instance

- Security Group: Mở port 22 (SSH), 80 (HTTP), 8080 (Application)
- Storage: 8GB General Purpose SSD

3. Chi tiết phiên bản:

- Public IP: 3.92.10.169

- Instance ID: i-0123456789abcdef0
- Security Group: hotelbooking

```
sudo apt update && sudo apt upgrade -y  
curl -fsSL https://get.docker.com -o get-docker.sh  
sudo sh get-docker.sh
```

Thêm user ubuntu vào group docker để không cần sudo mỗi lần:

```
sudo usermod -aG docker ubuntu
```

Đăng xuất SSH và SSH lại để nhận quyền group docker, rồi chạy:

```
docker --version
```

Cài Docker Compose:

```
sudo apt install docker-compose -y
```

Kiểm tra:

```
docker-compose --version
```

Cấu hình tường lửa:

```
sudo ufw allow OpenSSH  
sudo ufw allow 80  
sudo ufw allow 443  
sudo ufw allow 3000  
sudo ufw allow 8000  
sudo ufw enable  
sudo ufw status
```

Triển khai dự án Hotel Booking trên AWS EC2:

```
sudo apt update && sudo apt upgrade -y
```

Cài Docker

```
curl -fsSL https://get.docker.com -o get-docker.sh  
sudo sh get-docker.sh
```

Thêm user 'ubuntu' vào docker group (để không cần sudo)

```
sudo usermod -aG docker ubuntu
```


Đăng xuất:

exit

chạy lại:

```
ssh -i "D:/tân/hotel-booking-key.pem" ubuntu@3.92.10.169
```

Clone dự án từ GitHub:

```
git clone https://github.com/ngoctan04/cnpm1.git
```

di chuyển vào thư mục:

```
cd cnpm1
```

Chạy ứng dụng:

```
docker compose up -d --build
```

frontend:

[http:// 3.92.10.169:3000/](http://3.92.10.169:3000/)

5. QUẢN LÝ DỰ ÁN

5.1. Cách sử dụng Jira để lập kế hoạch và theo dõi tiến độ

Trong dự án xây dựng hệ thống Quản lý đặt phòng khách sạn, nhóm sử dụng Jira Software – một công cụ quản lý dự án phổ biến của Atlassian – nhằm hỗ trợ quá trình phân công công việc, quản lý tiến độ, và theo dõi sự phối hợp giữa các thành viên trong nhóm. Jira là nền tảng phù hợp để áp dụng phương pháp phát triển Agile/Scrum, vốn đề cao sự linh hoạt và làm việc theo từng chu kỳ ngắn (Sprint).

a. Tạo dự án và cấu hình ban đầu

Nhóm tạo một dự án với tên CNPM_UDDPKS (viết tắt của Công nghệ phần mềm – Ứng dụng đặt phòng khách sạn). Dự án được thiết lập với mẫu Scrum Project, cho phép chia công việc theo từng Sprint kéo dài 1–2 tuần, phù hợp với tiến độ học phần.

Giao diện Jira bao gồm:

- Backlog: nơi lưu trữ tất cả các công việc chưa được đưa vào Sprint.
- Board (Scrum/Kanban): bảng theo dõi công việc theo trạng thái (To Do – In Progress – Done).
- Reports: hiển thị biểu đồ thống kê, báo cáo tiến độ.

b. Lập kế hoạch công việc

Dự án được chia thành các Epic tương ứng với từng nhóm chức năng chính:

- Thiết kế cơ sở dữ liệu (MySQL)
- Xây dựng backend bằng FastAPI
- Xây dựng giao diện frontend bằng React
- Chức năng đặt phòng, thống kê
- Tích hợp hệ thống và triển khai

Mỗi Epic được chia nhỏ thành nhiều Issue loại Task, Story, hoặc Bug. Các Issue này mô tả chi tiết từng công việc cụ thể: tạo bảng dữ liệu, viết API, tạo form giao diện, truy vấn thống kê doanh thu,...

Mỗi công việc đều được gán:

- Người thực hiện (Assignee) rõ ràng
- Mức độ ưu tiên (Priority): từ thấp đến cao
- Thời gian thực hiện trong từng Sprint
- Trạng thái: To Do (chưa làm), In Progress (đang làm), Done (đã xong)

c. Quản lý theo Sprint

Dự án được chia thành nhiều Sprint, mỗi Sprint kéo dài 1–2 tuần.

Ví dụ Sprint:

- Sprint 1 (10/07 – 17/07): Thiết lập hệ thống, tạo giao diện đăng nhập/đăng ký, thiết kế cơ sở dữ liệu và viết các API cơ bản.
- Sprint 2 (18/07 – 31/07): Tích hợp frontend và backend, xây dựng chức năng đặt phòng, viết truy vấn thống kê doanh thu, hiển thị biểu đồ.

Mỗi Sprint đều được lập kế hoạch bằng cách kéo các task từ Backlog vào Sprint tương ứng, sau đó nhóm tiến hành thực hiện và cập nhật tiến độ trên Jira.

d. Theo dõi tiến độ công việc

Jira cung cấp nhiều công cụ trực quan để theo dõi tiến độ:

- Board: hiển thị các cột trạng thái (To Do, In Progress, Done), giúp nhóm dễ dàng theo dõi công việc đang thực hiện và hoàn thành.
- Burndown Chart: biểu đồ thể hiện số lượng công việc còn lại trong Sprint theo thời gian, giúp đánh giá xem nhóm có đang đi đúng tiến độ hay không.
- Velocity Chart: cho thấy số lượng công việc nhóm có thể hoàn thành trong mỗi Sprint, từ đó điều chỉnh khối lượng công việc hợp lý hơn trong Sprint sau.

e. Đánh giá hiệu quả sử dụng Jira

Việc sử dụng Jira giúp nhóm:

- Lập kế hoạch bài bản, rõ ràng từ đầu
- Phân công công việc minh bạch, tránh chồng chéo
- Theo dõi tiến độ dễ dàng thông qua board và biểu đồ
- Dễ dàng điều chỉnh khi có thành viên bị chậm tiến độ hoặc thay đổi yêu cầu
- Ghi lại lịch sử hoạt động, trao đổi rõ ràng trong mỗi task thông qua comment

Việc cập nhật công việc thường xuyên giúp cả nhóm chủ động trong kiểm soát tiến độ, không bị động hay trễ hạn, đồng thời hỗ trợ báo cáo cuối kỳ một cách minh bạch.

5.2. Phân công nhiệm vụ của từng thành viên trong nhóm

Dự án được triển khai bởi nhóm gồm 3 thành viên. Nhóm đã phân chia công việc theo năng lực và chuyên môn từng người để đảm bảo hiệu quả cao nhất trong quá trình phát triển.

Họ và tên	Vai trò	Nhiệm vụ chính
Tân	Lập trình viên Backend (FastAPI)	- Thiết lập môi trường backend và cấu trúc dự án FastAPI - Xây dựng các API cho chức năng quản lý khách sạn, phòng, đặt phòng - Xử lý logic đăng nhập/đăng ký (JWT, Hashing) - Hỗ trợ tích hợp API với frontend
Phương	Lập trình viên Frontend (ReactJS)	- Thiết kế giao diện người dùng: đăng ký, đăng nhập, dashboard admin - Giao tiếp với API backend bằng React Axios - Hiển thị dữ liệu danh sách khách sạn, phòng, đặt phòng - Tạo biểu đồ thống kê (dùng Chart.js)
Nhi	Chuyên viên Cơ sở dữ liệu (MySQL & SQL)	- Thiết kế mô hình dữ liệu, tạo các bảng và ràng buộc - Viết các truy vấn SQL thống kê số lượng phòng, doanh thu theo tháng - Hỗ

		trợ backend trong việc truy xuất dữ liệu hiệu quả - Kiểm tra và tối ưu hóa truy vấn
--	--	---

Cả nhóm phối hợp thông qua Jira và thường xuyên họp nhóm để:

- Cập nhật tiến độ từng người
- Trao đổi và giải quyết vấn đề kỹ thuật
- Giao tiếp và thảo luận qua comment trên từng task

Việc phân công rõ ràng giúp mỗi thành viên chủ động công việc, đồng thời đảm bảo tính gắn kết trong quá trình làm việc nhóm.

6. KIỂM THỬ

6.1. Chiến lược kiểm thử và công cụ sử dụng

6.1.1. Các cấp độ kiểm thử

1. Unit Testing (Kiểm thử đơn vị)

6.1.2. Chiến lược kiểm thử

1. Phát triển theo hướng kiểm thử (TDD)

- Viết bài kiểm tra trước
- Triển khai mã để vượt qua bài kiểm tra
- Tái cấu trúc với độ tin cậy cao

2. Kiểm tra liên tục

- Kiểm tra tự động trong CI/CD
- Chạy kiểm tra trên mỗi lần xác nhận
- Chặn hợp nhất nếu kiểm tra không thành công

6.1.3. Công cụ kiểm thử sử dụng

1. Công cụ kiểm thử Backend:

- **pytest**: Khung kiểm thử Python
- **pytest-cov**: Độ bao phủ mã

- **pytest-asyncio**: Hỗ trợ kiểm thử bất đồng bộ

2. Công cụ kiểm thử giao diện người dùng:

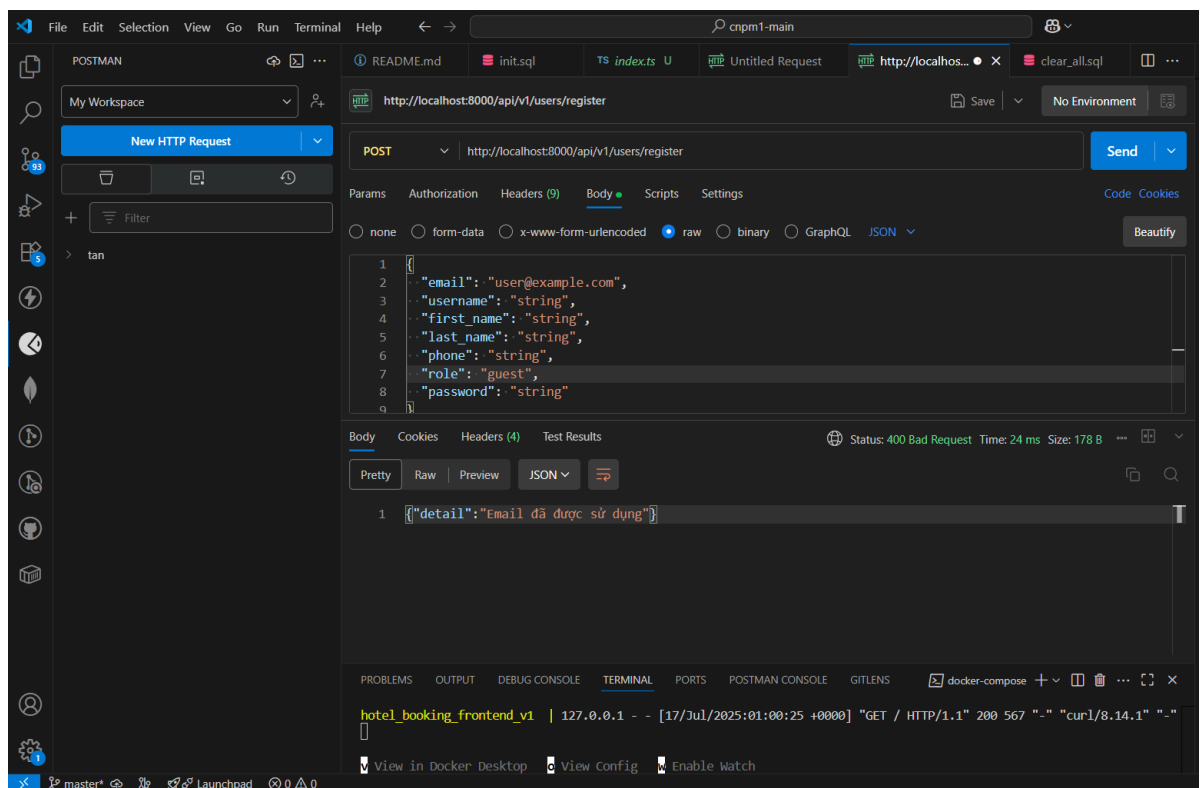
Jest: Khung kiểm thử JavaScript

3. API Testing Tools:

- **Postman**: Manual và automated API testing

Kiểm thử Postman đăng kí người dùng

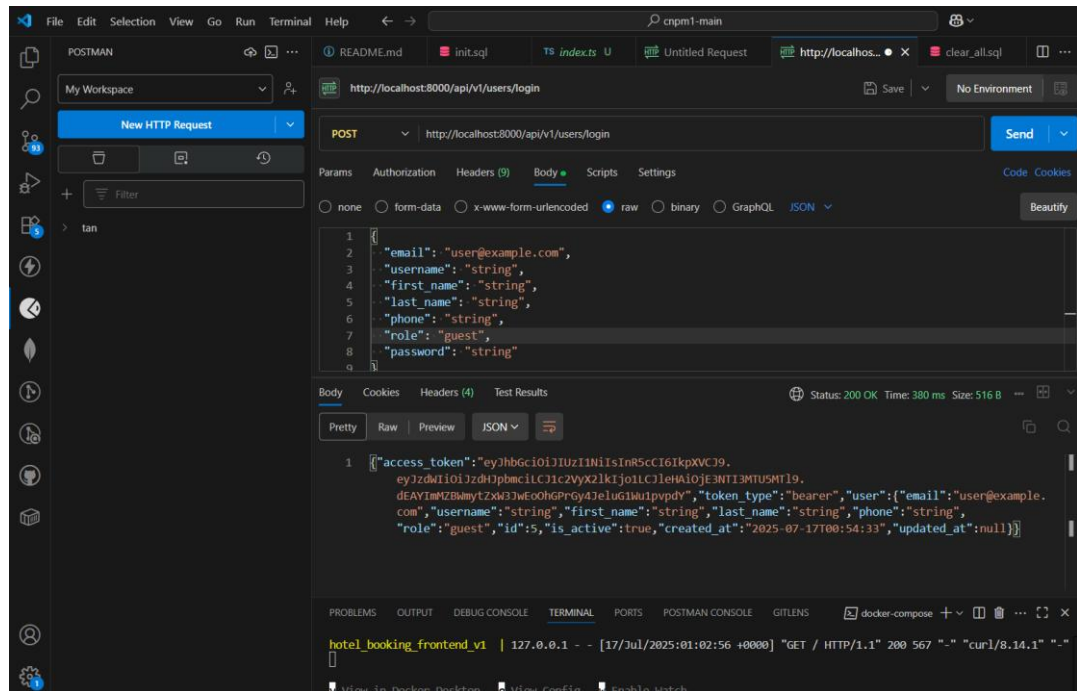
http://localhost:8000/api/v1/users/register



Hình 4: Kiểm thử Postman đăng kí người dùng

Kiểm thử Postman đăng nhập người dùng

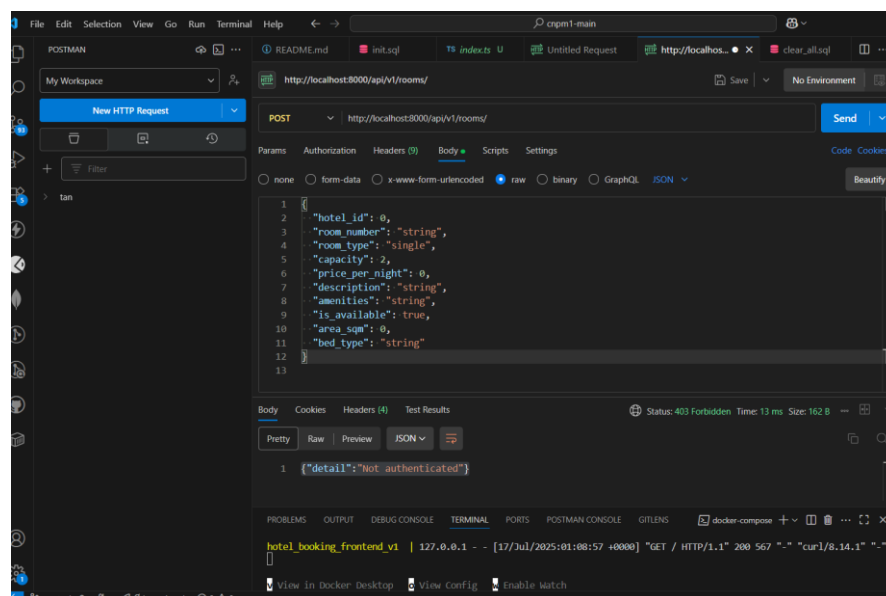
<http://localhost:8000/api/v1/users/login>



Hình 5: Kiểm thử Postman đăng nhập người dùng

Kiểm thử Postman tạo phòng

<http://localhost:8000/api/v1/rooms/>



Hình 6: Kiểm thử Postman tạo phòng

- **Swagger UI: Interactive API testing**

Người dùng:

👤 Người dùng			^
POST	/api/v1/users/register	Register User	▼
POST	/api/v1/users/login	Login User	▼
GET	/api/v1/users/me	Get Current User Profile	🔒 ▼
GET	/api/v1/users/	Get Users	🔒 ▼
GET	/api/v1/users/{user_id}	Get User	🔒 ▼
PUT	/api/v1/users/{user_id}	Update User	🔒 ▼
DELETE	/api/v1/users/{user_id}	Delete User	🔒 ▼
POST	/api/v1/users/{user_id}/change-password	Change Password	🔒 ▼
GET	/api/v1/users/stats/overview	Get User Stats	🔒 ▼

Hình 7: Swagger UI người dùng

Khách sạn:

🏨 Khách sạn			^
POST	/api/v1/hotels/	Create Hotel	🔒 ▼
GET	/api/v1/hotels/	Get Hotels	▼
GET	/api/v1/hotels/{hotel_id}	Get Hotel	▼
PUT	/api/v1/hotels/{hotel_id}	Update Hotel	🔒 ▼
DELETE	/api/v1/hotels/{hotel_id}	Delete Hotel	🔒 ▼
GET	/api/v1/hotels/{hotel_id}/rooms	Get Hotel Rooms	▼
GET	/api/v1/hotels/{hotel_id}/stats	Get Hotel Stats	🔒 ▼
GET	/api/v1/hotels/stats/overview	Get Hotels Overview	🔒 ▼
POST	/api/v1/hotels/search	Search Hotels	▼
POST	/api/v1/hotels/{hotel_id}/upload-images	Upload Hotel Images	🔒 ▼

Hình 8: Swagger UI Khách sạn

Phòng:

Phòng		
POST	/api/v1/rooms/	Create Room
GET	/api/v1/rooms/	Get Rooms
GET	/api/v1/rooms/{room_id}	Get Room
PUT	/api/v1/rooms/{room_id}	Update Room
DELETE	/api/v1/rooms/{room_id}	Delete Room
GET	/api/v1/rooms/{room_id}/availability	Check Room Availability
POST	/api/v1/rooms/{room_id}/maintenance	Set Room Maintenance
POST	/api/v1/rooms/search	Search Rooms
GET	/api/v1/rooms/stats/overview	Get Rooms Stats
POST	/api/v1/rooms/{room_id}/upload-images	Upload Room Images

Hình 9: Swagger UI của phòng

Đặt phòng:

Đặt phòng		
GET	/api/v1/bookings/	Get Bookings
POST	/api/v1/bookings/	Create Booking
GET	/api/v1/bookings/my-bookings/	Get My Bookings
GET	/api/v1/bookings/user/{user_id}	Get User Bookings
GET	/api/v1/bookings/{booking_id}/	Get Booking
PUT	/api/v1/bookings/{booking_id}	Update Booking
DELETE	/api/v1/bookings/{booking_id}	Delete Booking
POST	/api/v1/bookings/{booking_id}/cancel/	Cancel Booking
POST	/api/v1/bookings/{booking_id}/confirm	Confirm Booking
GET	/api/v1/bookings/stats/overview	Get Booking Stats
GET	/api/v1/bookings/upcoming/list	Get Upcoming Bookings
GET	/api/v1/bookings/current/guests	Get Current Guests

Hình 10: Swagger UI của đặt phòng

Thanh toán:

Thanh toán		
GET	/api/v1/payments/	Get Payments
GET	/api/v1/payments/my-payments	Get My Payments
GET	/api/v1/payments/user/{user_id}	Get User Payments
GET	/api/v1/payments/booking/{booking_id}	Get Booking Payments
GET	/api/v1/payments/{payment_id}	Get Payment
PUT	/api/v1/payments/{payment_id}	Update Payment
DELETE	/api/v1/payments/{payment_id}	Delete Payment
POST	/api/v1/payments/{payment_id}/process	Process Payment
POST	/api/v1/payments/{payment_id}/fail	Fail Payment
POST	/api/v1/payments/{payment_id}/cancel	Cancel Payment
GET	/api/v1/payments/stats/overview	Get Payment Stats
GET	/api/v1/payments/booking/{booking_id}/status	Get Booking Payment Status
GET	/api/v1/payments/recent/list	Get Recent Payments

Hình 11: Swagger UI của thanh toán

6.2. Kết quả kiểm thử API

UNIT TEST cho Frontend:

```
A worker process has failed to exit gracefully and has been force exited. This is likely caused by tests
leaking due to improper teardown. Try running with --detectOpenHandles to find leaks. Active time
rs can also cause this, ensure that .unref() was called on them.

Test Suites: 3 passed, 3 total
Tests:       3 passed, 3 total
Snapshots:   0 total
Time:        3.444 s
Ran all test suites.
PS D:\tan\cnpm1\frontend>
```

Hình 12: UNIT TEST cho Frontend

test cho các page:

- HomePage.test.tsx
- RegisterPage.test.tsx
- HotelDetailPage.test.tsx

Unit Test giúp phát hiện lỗi khi refactor.

Đảm bảo các chức năng cơ bản hoạt động ổn định.

UNIT TEST cho Backend:

Chạy test:

```
pytest tests/ --maxfail=1 --disable-warnings -v
```

```
tests/test_auth.py::test_login_fail PASSED [ 14%]
tests/test_auth.py::test_login_success PASSED [ 28%]
tests/test_bookings.py::test_health_check PASSED [ 42%]
tests/test_main.py::test_root PASSED [ 57%]
tests/test_main.py::test_health_check PASSED [ 71%]
tests/test_rooms.py::test_create_and_update_room PASSED [ 85%]
tests/test_users.py::test_register_login_get_update_user PASSED [100%]

===== 7 passed, 9 warnings in 1.14s =====
PS D:\tan\cnpm1\backend>
```

Hình 13: UNIT TEST cho Backend

API authentication hoạt động ổn định (đăng ký, đăng nhập).

API quản lý phòng hoạt động ổn định (tạo, cập nhật).

API health check ổn định (đảm bảo server chạy).

API quản lý người dùng hoạt động ổn định (CRUD user).

Không có lỗi khi chạy các chức năng chính backend.

7. ĐÁNH GIÁ VÀ KẾT LUẬN

7.1. Những khó khăn gặp phải trong quá trình thực hiện

Trong quá trình phát triển hệ thống đặt phòng khách sạn, nhóm đã gặp phải nhiều thách thức và khó khăn. Việc nhận diện và giải quyết những vấn đề này đã giúp nhóm học hỏi được nhiều kinh nghiệm quý báu.

7.1.1. Khó khăn về kỹ thuật

1. Tích hợp Google Drive API:

- Vấn đề: Xử lý authentication và authorization phức tạp
- Thách thức: Quản lý token refresh và quota limits
- Giải pháp: Implement caching mechanism và error handling robust

2. Xử lý concurrent bookings:

- Vấn đề: Race conditions khi nhiều người đặt cùng phòng
- Thách thức: Database locking và transaction isolation
- Giải pháp: Sử dụng database transactions và optimistic locking

3. JWT Token Management:

- Vấn đề: Token expiration và refresh flow
- Thách thức: Balance giữa security và user experience
- Giải pháp: Implement silent refresh với refresh token

4. Tối ưu hóa hiệu suất:

- Vấn đề: Slow queries với large dataset
- Thách thức: N+1 query problem trong ORM
- Giải pháp: Query optimization và proper indexing

7.1.2. Khó khăn về thiết kế

1. Thiết kế lược đồ cơ sở dữ liệu:

- Vấn đề: Cân nhắc giữa normalization và performance
- Thách thức: Handle complex relationships
- Giải pháp: Selective denormalization cho read-heavy operations

2. Tính nhất quán của thiết kế API:

- Vấn đề: Maintain RESTful principles
- Thách thức: Handle complex business logic
- Giải pháp: Clear API guidelines và documentation

3. UI/UX cho nhiều vai trò:

- Vấn đề: Different interfaces cho Admin và Guest
- Thách thức: Reusable components với role-based rendering
- Giải pháp: Component composition và conditional rendering

7.2. Bài học rút ra và đề xuất cải thiện trong tương lai

7.2.1. Bài học kinh nghiệm

1. Về Thiết kế Kiến trúc:

Kiến trúc microservices mang lại tính linh hoạt nhưng tăng độ phức tạp. Phân tách rõ ràng các mối quan tâm giúp duy trì và mở rộng dễ dàng. Thiết kế API-first giúp phát triển song song.

2. Về Công nghệ Stack:

- FastAPI là lựa chọn tuyệt vời để xây dựng API hiện đại
- React với TypeScript cải thiện chất lượng mã đáng kể

- Docker đơn giản hóa việc triển khai nhưng cần cấu hình phù hợp

7.2.2. Đề xuất cải thiện

1. Cải tiến tính năng:

Hệ thống Đặt phòng:

- Triển khai quy trình sửa đổi đặt phòng
- Thêm quản lý chính sách hủy
- Hỗ trợ đặt phòng theo nhóm
- Tích hợp với các nền tảng đặt phòng bên ngoài

Hệ thống Thanh toán:

- Tích hợp cổng thanh toán thực tế (Stripe, PayPal)
- Hỗ trợ nhiều loại tiền tệ
- Triển khai tự động hoàn tiền
- Thêm tính năng tạo hóa đơn

Trải nghiệm người dùng:

- Thêm hỗ trợ đa ngôn ngữ
- Triển khai công cụ đề xuất
- Thêm hệ thống đánh giá và xếp hạng
- Phát triển ứng dụng di động

Tính năng Quản trị:

- Bảng điều khiển phân tích nâng cao
- Dự báo doanh thu
- Thuật toán tối ưu hóa công suất phòng
- Quản lý chiến dịch tiếp thị

2. Cải tiến kỹ thuật:

Hiệu suất:

- Triển khai lớp bộ nhớ đệm Redis
- Thêm CDN cho tài nguyên tĩnh

- Bản sao đọc cơ sở dữ liệu
- Triển khai GraphQL cho truy vấn linh hoạt

Bảo mật:

- Xác thực hai yếu tố
- Đăng nhập mạng xã hội OAuth2
- Giới hạn tốc độ API cho mỗi người dùng
- Triển khai ghi nhật ký kiểm tra

Khả năng mở rộng:

- Điều phối Kubernetes
- Hàng đợi tin nhắn cho các hoạt động bất đồng bộ
- Các dịch vụ vi mô cho các môi quan tâm riêng biệt
- Triển khai kiến trúc hướng sự kiện

Giám sát:

- Tích hợp các công cụ APM (New Relic, DataDog)
- Thiết lập hệ thống cảnh báo
- Triển khai theo dõi phân tán
- Thêm tính năng theo dõi số liệu kinh doanh

7.2.3. Kết luận

Dự án Hệ thống đặt phòng khách sạn đã đạt được mục tiêu ban đầu về việc xây dựng một platform hoàn chỉnh cho quản lý khách sạn và đặt phòng trực tuyến. Với kiến trúc microservices, công nghệ hiện đại và quy trình phát triển chuyên nghiệp, hệ thống có nền tảng vững chắc cho việc mở rộng và phát triển trong tương lai.

Những bài học rút ra từ dự án không chỉ về mặt kỹ thuật mà còn về quản lý dự án, làm việc nhóm và tư duy giải quyết vấn đề. Đây là những kinh nghiệm quý báu cho các dự án phần mềm trong tương lai.

Với roadmap rõ ràng và các đề xuất cải thiện cụ thể, hệ thống có tiềm năng phát triển thành một giải pháp toàn diện cho ngành khách sạn, đáp ứng nhu cầu ngày càng cao của thị trường du lịch và hospitality.

8. PHỤ LỤC

8.1. Hướng dẫn cài đặt và chạy ứng dụng

8.1.1. Yêu cầu hệ thống

Hardware Requirements:

- CPU: 2 cores minimum (4 cores recommended)
- RAM: 4GB minimum (8GB recommended)
- Storage: 10GB free space
- Network: Stable internet connection

Software Requirements:

- Operating System: Windows 10/11, macOS 10.15+, Linux (Ubuntu 20.04+)
- Docker Desktop: Version 20.10 or higher
- Docker Compose: Version 1.29 or higher
- Git: Version 2.30 or higher
- Web Browser: Chrome, Firefox, Safari, Edge (latest versions)

8.1.2. Cài đặt môi trường

1. Cài đặt Docker Desktop

Windows:

1. Download Docker Desktop từ <https://www.docker.com/products/docker-desktop/>
2. Run installer và follow instructions
3. Restart computer sau khi cài đặt
4. Verify: Mở terminal và chạy `docker --version`

macOS:

1. Download Docker Desktop cho Mac

2. Double-click Docker.dmg và drag to Applications
3. Start Docker từ Applications
4. Verify trong terminal: `docker --version`

Linux (Ubuntu):

```
# Update package index
sudo apt-get update

# Install dependencies
sudo apt-get install ca-certificates curl gnupg lsb-release

# Add Docker GPG key
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
--dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

# Add Docker repository
echo "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs)
stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Install Docker Engine
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io docker-
compose-plugin

# Add user to docker group
sudo usermod -aG docker $USER

# Logout and login again
```

2. Cài đặt Git

Windows:

- Download từ <https://git-scm.com/download/win>

Chạy trình cài đặt với các tùy chọn mặc định

macOS:

- Install qua Homebrew: `brew install git`
- Hoặc download từ <https://git-scm.com/download/mac>

Linux:

```
sudo apt-get install git
```

8.1.3. Clone và setup project

1. Clone repository

```
# Clone từ GitHub
git clone https://github.com/[username]/hotel-booking-system.git

# Di chuyển vào thư mục project
cd hotel-booking-system
```

2. Cấu hình environment variables

Backend configuration:

```
# Copy file environment mẫu
cp backend/.env.example backend/.env

# Edit file .env với thông tin sau:
# DATABASE_HOST=mysql
# DATABASE_PORT=3306
# DATABASE_NAME=hotel_booking
# DATABASE_USER=hotel_user
# DATABASE_PASSWORD=hotel_password
# SECRET_KEY=your-secret-key-here-change-in-production
# ALGORITHM=HS256
# ACCESS_TOKEN_EXPIRE_MINUTES=30
```

Frontend configuration:

```
# Copy file environment mẫu
cp frontend/.env.example frontend/.env

# Edit file .env với thông tin sau:
# REACT_APP_API_URL=http://localhost:8000
```

3. Google Drive API Setup (Optional)

Nếu muốn sử dụng tính năng upload ảnh lên Google Drive:

Truy cập Google Cloud Console: <https://console.cloud.google.com/>

1. Tạo project mới hoặc chọn project có sẵn
2. Enable Google Drive API
3. Tạo credentials (OAuth 2.0 Client ID)
4. Download file `client_secret.json`
5. Copy file vào `backend/gdrive/client_secret.json`

8.1.4. Chạy ứng dụng với Docker Compose

1. Build và start tất cả services

```
# Build images lần đầu
docker-compose build

# Start all services
docker-compose up -d

# Hoặc build và start cùng lúc
docker-compose up -d --build
```

2. Kiểm tra trạng thái services

```
# Xem danh sách containers đang chạy
docker-compose ps

# Xem logs của tất cả services
docker-compose logs -f

# Xem logs của service cụ thể
docker-compose logs -f backend
docker-compose logs -f frontend
docker-compose logs -f mysql
```

3. Truy cập ứng dụng

Sau khi tất cả services khởi động thành công:

- **Frontend (React App):** <http://localhost:3000>
- **Backend API:** <http://localhost:8000>
- **API Documentation (Swagger):** <http://localhost:8000/docs>
- **API Documentation (ReDoc):** <http://localhost:8000/redoc>
- **MySQL Database:** localhost:3307 (user: hotel_user, password: hotel_password)

8.1.5. Sử dụng tài khoản demo

Hệ thống tự động tạo các tài khoản demo khi khởi động lần đầu:

Tài khoản Admin:

Username: `admin`

Password: `admin123`

Quyền: Quản lý toàn bộ hệ thống

Tài khoản Guest:

Username: `guest1` / Password: `guest123`

Username: `guest2` / Password: `guest123`

Username: `guest3` / Password: `guest123`

Quyền: Đặt phòng, xem thông tin cá nhân

8.1.6. Các lệnh Docker Compose hữu ích

Stop services:

```
# Stop tất cả services
docker-compose stop

# Stop service cụ thể
docker-compose stop backend
```

Restart services:

```
# Restart tất cả services
docker-compose restart

# Restart service cụ thể
docker-compose restart frontend
```

Remove services:

```
# Stop và remove containers
docker-compose down

# Remove với volumes (xóa cả database)
docker-compose down -v
```

Execute commands trong container:

```
# Access backend shell
docker-compose exec backend bash

# Access MySQL
docker-compose exec mysql mysql -u hotel_user -p

# Run migrations manually
docker-compose exec backend python -m alembic upgrade head
```

8.1.7. Troubleshooting

1. Port đã được sử dụng:

```
# Error: bind: address already in use

# Kiểm tra process đang dùng port
lsof -i :3000 # Frontend port
lsof -i :8000 # Backend port
lsof -i :3307 # MySQL port

# Kill process
kill -9 <PID>
```

2. Database connection error:

- Đảm bảo MySQL container đã start hoàn toàn
- Check logs: `docker-compose logs mysql`
- Verify credentials trong .env file

3. Frontend không connect được backend:

- Check CORS configuration
- Verify REACT_APP_API_URL trong frontend/.env
- Check backend logs cho errors

4. Docker issues:

```
# Reset Docker
docker system prune -a

# Rebuild images
docker-compose build --no-cache

# Check Docker daemon
sudo systemctl status docker # Linux
```

8.2. Liên kết GitHub repository và link demo

- **GitHub Repository:** [repository_url]
- **Demo:** <http://3.92.10.169:3000/>

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] M. B. Jones, J. Bradley, and N. Sakimura, "JSON Web Token (JWT)," Internet Engineering Task Force (IETF), RFC 7519, May 2015. [Online]. Available: <https://tools.ietf.org/html/rfc7519>