

BÁO CÁO ĐỒ ÁN

Môn học: Bảo mật web và ứng dụng

Tên chủ đề: Bảo mật web dùng NodeJs

GVHD: Nguyễn Công Danh

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT213.P12.ANTT

STT	Họ và tên	MSSV	Email
1	Trịnh Thị Bích Thảo	22521376	22521376@gm.uit.edu.vn
2	Huỳnh Trung Thuận	22521444	22521444@gm.uit.edu.vn
3	Lê Hiệp Thuận	22521446	22521446@gm.uit.edu.vn
4	Phạm Trung Thành	22521360	22521360@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:¹

STT	Công việc	Kết quả tự đánh giá
1	Phát triển ứng dụng	100%
2	Thiết kế các nguyên tắc bảo mật	100%
3	Kiểm thử bảo mật	100%
4	Deploy và giám sát	100%

Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

¹ Ghi nội dung công việc

Nội dung

A.	TỔNG QUAN	7
1.	Giới thiệu ứng dụng.....	7
a)	Giới thiệu về Nodejs	7
b)	Ứng dụng quản lý và đặt phòng khách sạn - LastingTrip	7
2.	Mô hình xây dựng ứng dụng	8
a)	Giới thiệu về mô hình MVC	8
b)	Phát triển Lastingtrip theo kiến trúc MVC.....	10
c)	Thiết kế bảo mật trên các thành phần của mô hình	11
3.	Các chức năng của ứng dụng.	11
a)	Đăng ký, đăng nhập, quên mật khẩu	11
b)	Search.....	19
c)	Filter	20
d)	Chatbot	21
e)	Tìm kiếm địa điểm từ ảnh	21
f)	Đặt phòng	23
g)	Thanh toán.....	24
h)	Bình luận.....	30
i)	Quản lí khách sạn và các thông tin khác của hệ thống.....	32
j)	Quản lí tài khoản người dùng.....	33
B.	CHÍNH SÁCH BẢO MẬT	36
4.	Hệ thống xác thực và phân quyền người dùng.....	36
a)	Các vai trò trong ứng dụng.....	36
b)	Access Token và Refresh Token.....	37
c)	CSRF Token.....	42
5.	Bảo mật cơ sở dữ liệu	44
d)	ORM (Object-Relational Mapping), Sequelize.....	44
e)	Mã hóa mật khẩu.....	45
f)	Passport	46
g)	Kiểm tra và làm sạch đầu vào	46
6.	Ứng dụng các thư viện hỗ trợ tích hợp an toàn.....	50

a)	Express-rate-limit.....	50
b)	CORS	51
c)	JWT (Json web token), bcryptjs và Crypto	51
d)	Crusf.....	53
e)	dotenv.....	53
f)	Multer-storage-cloudinary.....	54
g)	sqlString	55
7.	Thiết kế helmet CSP (Content Security Policy)	55
C.	Kết quả và demo	58
8.	Giao diện tổng quan của ứng dụng.....	58
9.	Deploy ứng dụng	63
10.	Kết quả kiểm thử bảo mật	64
a)	Zap	64
b)	Synk	67
c)	K6.....	68
d)	Acunetix	69
11.	Demo	70

MỤC LỤC HÌNH ẢNH

Hình 1.	Sơ đồ hoạt động của MVC	10
Hình 2.	Giao diện trang đăng kí.....	12
Hình 3.	Mã OTP được gửi về mail của người dùng	13
Hình 4.	Tiến hành xác thực OTP	14
Hình 5.	Tiến hành đăng nhập.....	15
Hình 6.	Sau khi đăng nhập sẽ yêu cầu xác thực OTP	16
Hình 7.	OTP được gửi về mail người dùng	16
Hình 8.	Nhập thông tin tài khoản google.....	17
Hình 9.	Xác nhận thông tin google	17
Hình 10.	Chức năng quên mật khẩu	18
Hình 11.	Thông tin được trả về mail	18
Hình 12.	Tiến hành đổi mật khẩu	19
Hình 13.	Giao diện của thanh Search	20

Hình 14.	Kết quả trả về khi tìm kiếm	20
Hình 15.	Filter theo nhu cầu của người	21
Hình 16.	Chức năng ChatBot	21
Hình 17.	Chức năng tìm kiếm địa điểm qua hình ảnh.....	22
Hình 18.	Các phòng khách sạn phù hợp với địa điểm.....	23
Hình 19.	Chức năng đặt phòng để chuyển tới trang thanh toán	23
Hình 20.	Trang thanh toán trước điền coupon.....	24
Hình 21.	Trang thanh toán khi áp dụng coupon	25
Hình 22.	Trang phương thức thanh toán để lựa chọn ngân hàng	26
Hình 23.	Trang thanh toán của vnpay.....	27
Hình 24.	Chính sách và điều khoản của vnpay.....	28
Hình 25.	Trang xác thực OTP	29
Hình 26.	Trang thanh toán thành công	29
Hình 27.	Thư xác nhận đã đăng ký phòng thành công	30
Hình 28.	Chức năng đánh giá và comment	31
Hình 29.	Đánh giá được hiển thị tới các người dùng khác	31
Hình 30.	Trang quản lý khách sạn của admin	32
Hình 31.	Trang quản lý các tiện nghi của khách sạn.....	32
Hình 32.	Trang quản lý đơn đặt hàng	33
Hình 33.	Quản lí danh sách người dùng	33
Hình 34.	Chỉnh sửa thông tin người dùng	34
Hình 35.	Trang biểu đồ thống kê	34
Hình 36.	Thêm mã coupon	35
Hình 37.	Quản lý mã giảm giá	35
Hình 38.	Middleware phân quyền Admin	36
Hình 39.	Middle kiểm tra user thường	37
Hình 40.	Lưu access token vào cookies	37
Hình 41.	Tiến hành tạo accessToken	38
Hình 42.	API lấy thông tin người dùng hiện tại	38
Hình 43.	RefreshToken	39
Hình 44.	Tạo và lưu cặp Token	40
Hình 45.	Middleware kiểm tra Token	41
Hình 46.	Thêm các middleware vào trước các Routes	41
Hình 47.	Thêm CSRF Token vào Header	42

Hình 48.	Middleware kiểm tra CSRF.....	42
Hình 49.	Thêm Middleware vào các Router	43
Hình 50.	Thêm Token vào front end	44
Hình 51.	Gửi request kèm Token	44
Hình 52.	Sequelize	45
Hình 53.	Băm mật khẩu.....	46
Hình 54.	Passport	46
Hình 55.	Validator.....	47
Hình 56.	Sanitize	48
Hình 57.	Code sanitizeLoginInputs.....	49
Hình 58.	Khai báo biến sử dụng validator.....	49
Hình 59.	Dùng sanitize cho các trường dữ liệu	50
Hình 60.	Thư viện rate-limit.....	50
Hình 61.	Code thực hiện rate-limit.....	51
Hình 62.	Import vào trong các phương thức	51
Hình 63.	Code sử dụng thư viện cors	51
Hình 64.	Code thực hiện cors	51
Hình 65.	Tạo accessToken có thời hạn trong 40 phút.....	52
Hình 66.	Băm mật khẩu.....	52
Hình 67.	Dùng thư viện crypto.....	52
Hình 68.	Import thư viện crypto.....	53
Hình 69.	Sử dụng thư viện Cruf.....	53
Hình 70.	Sử dụng dotenv.....	53
Hình 71.	File .env	54
Hình 72.	Thông báo lỗi	54
Hình 73.	Sử dụng package multer-storage-cloudinary.....	55
Hình 74.	Sử dụng thư viện sqlstring.....	55
Hình 75.	Code thực hiện.....	55
Hình 76.	Code thực hiện helmet.....	56
Hình 77.	Code thực hiện helmet (tiếp theo)	57
Hình 78.	Giao diện trang chủ	58
Hình 79.	Giao diện khách sạn	58
Hình 80.	Giao diện trang user	59
Hình 81.	Trang about us	59

Hình 82.	Giao diện trang owner	60
Hình 83.	Trang quản lý khách sạn.....	60
Hình 84.	Trang quản lý phòng.....	61
Hình 85.	Trang giao diện của admin	61
Hình 86.	Hệ thống PNG Stack	62
Hình 87.	Server giám sát hệ thống của Ứng dụng.....	63
Hình 88.	AWS	63
Hình 89.	Ứng dụng quét lỗ hổng bảo mật Zap trang ForgotPass	64
Hình 90.	Ứng dụng quét lỗ hổng bảo mật Zap cho api goi chatbot.....	64
Hình 91.	Zap.....	65
Hình 92.	Kết quả quét Zap	65
Hình 93.	Kết quả quét Zap	66
Hình 94.	Zap.....	66
Hình 95.	Zap.....	67
Hình 96.	Kết quả quét Snyk	67
Hình 97.	Lỗ hổng	68
Hình 98.	68
Hình 99.	Kết quả thực hiện quét trên k6	69
Hình 100.	Kết quả thực hiện quét trên Acunetix.....	69
Hình 101.	Kết quả thực hiện quét trên Acunetix.....	70

BÁO CÁO CHI TIẾT

Các bước thực hiện/ Phương pháp thực hiện/Nội dung tìm hiểu (Ảnh chụp màn hình, có giải thích)

A. TỔNG QUAN

1. Giới thiệu ứng dụng

a) Giới thiệu về Nodejs

- Node.js là một nền tảng phát triển ứng dụng mã nguồn mở, được xây dựng trên JavaScript V8 engine của Google Chrome, cho phép các nhà phát triển tạo các ứng dụng web và dịch vụ máy chủ mạnh mẽ, hiệu quả. Node.js đã mở rộng khả năng của JavaScript từ việc chỉ phát triển front-end trong trình duyệt để bao gồm cả phát triển back-end. Điều này có nghĩa là các lập trình viên có thể sử dụng cùng một ngôn ngữ lập trình, JavaScript, để phát triển toàn bộ ứng dụng, từ front-end đến back-end, qua đó tạo điều kiện cho việc học tập và phát triển ứng dụng nhanh chóng và hiệu quả hơn.
- Các lí do nên phát triển ứng dụng với Node.js:
 - Hiệu suất cao: sử dụng V8 engine của Chrome, xử lý đồng thời được nhiều kết nối, thực thi mã nguồn nhanh chóng.
 - Hệ sinh thái rộng lớn: hệ sinh thái NPM khổng lồ (với hơn 1.3 triệu thư viện) cùng nhiều framework đa dạng, cộng đồng sử dụng cũng vô cùng đông đảo tạo môi trường cho các nhà phát triển.
 - Khả năng mở rộng: dễ dàng mở rộng và phát triển dự án.

b) Ứng dụng quản lý và đặt phòng khách sạn - LastingTrip

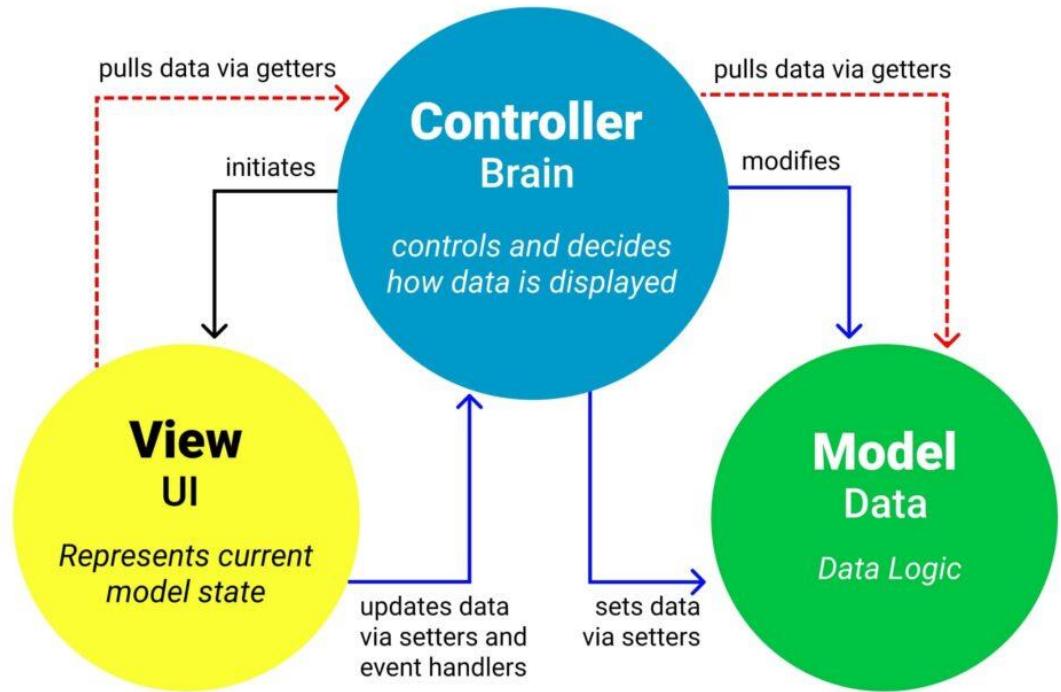
- Thực trạng: Trong guồng quay hối hả và áp lực của cuộc sống hiện đại, khao khát khám phá thế giới và tìm kiếm những trải nghiệm mới mẻ đã trở thành một phần tất yếu của con người. Du lịch không chỉ là liều thuốc giải tỏa căng thẳng, mà còn là cánh cửa mở ra những cơ hội kết nối với những nền văn hóa đa dạng, những cảnh quan ngoạn mục và những con người đầy cảm hứng. Nhận thức được nhu cầu ngày càng tăng này, các doanh nghiệp trên toàn thế giới đang không ngừng tận dụng công nghệ để cách mạng hóa ngành du lịch. Các ứng dụng tiên tiến cho việc đặt phòng khách sạn và lên kế hoạch hành trình đã nổi lên như một xu hướng tất yếu trong những năm gần đây. Lấy cảm hứng từ xu hướng toàn cầu này và mong muốn đóng góp vào sự phát triển của ngành, nhóm 10 chúng em đã phát triển LastingTrip, một ứng dụng quản lý và đặt phòng khách sạn hiện đại, được thiết kế để nâng cao trải nghiệm du lịch và giúp du khách tạo nên những kỷ niệm đáng nhớ.
- LastingTrip hướng đến việc mang lại trải nghiệm người dùng xuất sắc bằng cách cung cấp thông tin chi tiết, đánh giá thực tế và hình ảnh chất lượng cao về các khách sạn. Không chỉ giúp khách hàng tìm được nơi lưu trú phù hợp với nhu cầu và ngân sách, LastingTrip còn hỗ trợ họ lên kế hoạch chuyến đi một cách dễ dàng và nhanh chóng. Ứng dụng cung cấp các chức năng tìm kiếm khách sạn theo địa điểm, giá cả, đánh giá, tiện ích, cùng các gói ưu đãi và khuyến mãi đặc biệt. Hơn thế nữa, LastingTrip cam kết bảo mật thông tin người dùng bằng việc áp dụng các công nghệ mã hóa tiên tiến và tuân thủ nghiêm ngặt các quy định về bảo vệ dữ liệu. Mọi giao dịch đều được xử lý an toàn qua cổng thanh toán bảo mật, đảm bảo khách hàng hoàn toàn yên tâm khi sử dụng dịch vụ. Với LastingTrip, khách hàng không chỉ tìm được chốn dừng chân hoàn hảo mà còn được bảo vệ tối đa thông tin cá nhân và tài chính, để mỗi chuyến đi luôn là một trải nghiệm trọn vẹn và an tâm.

2. Mô hình xây dựng ứng dụng

a) Giới thiệu về mô hình MVC

- MVC viết tắt của cụm từ “Model-View-Controller”. Đây là mô hình thiết kế được sử dụng trong kỹ thuật phần mềm. MVC là một mẫu kiến trúc phần mềm để tạo lập giao diện người dùng trên máy tính. MVC chia thành ba phần được kết nối với nhau và mỗi thành phần đều có một nhiệm vụ riêng của nó và độc lập với các thành phần khác.
- MVC được sử dụng rộng rãi trong phát triển web, sự khác biệt được tùy chỉnh liên quan đến sự có mặt của server-client.
- Hoạt động của MVC:
 - View: về cơ bản, View đại diện cho cách dữ liệu được trình bày trong ứng dụng(UI). View được tạo ra dựa trên dữ liệu được thu thập từ model. Bằng cách yêu cầu thông tin từ model, sau đó sẽ trả kết quả đến người dùng.
 - Controller: là thành phần xử lý tương tác của người dùng. Dữ liệu đầu vào của người dùng được controller phân tích và xử lí, controller sẽ gửi thông tin đến model để xử lí và sau đó trả về kết quả view.
 - Model: là nơi lưu trữ dữ liệu và logic. Nó thao tác dữ liệu và gửi lại cơ sở dữ liệu, hoặc được sử dụng để hiển thị thông tin tương tự.

MVC Architecture Pattern



Hình 1. Sơ đồ hoạt động của MVC

b) Phát triển Lastingtrip theo kiến trúc MVC

- Dựa trên kiến trúc MVC, Lastingtrip đã được phát triển theo hệ thống Model-View-Controller.
 - **Model:** Được thiết kế để kết nối trực tiếp với cơ sở dữ liệu MySQL, đóng vai trò lưu trữ và xử lý các thông tin quan trọng. Model đảm bảo: Tổ chức dữ liệu theo mối quan hệ rõ ràng giữa các bảng. Cung cấp các phương thức truy vấn và cập nhật để hỗ trợ các API hoạt động hiệu quả. Tích hợp các logic xử lý như tự động xóa dữ liệu liên quan hoặc tính toán các giá trị động. Mỗi model tương ứng với một bảng trong cơ sở dữ liệu và được quản lý bởi Sequelize ORM, giúp đơn giản hóa các tác vụ truy vấn và xử lý dữ liệu.

- **View:** Hiển thị thông tin và xử lý tương tác người dùng, được tổ chức trong thư mục views với hai thư mục con: “User và Admin” để phân biệt giao diện cho người dùng thường và quản trị viên. View sử dụng Handlebars để nhúng dữ liệu từ Controller vào các mẫu HTML, sau đó kết xuất và gửi đến trình duyệt. Thư mục public chứa các file tĩnh như CSS, JavaScript, và hình ảnh, hỗ trợ trình bày và chức năng giao diện.
- **Controller:** Controller trong kiến trúc MVC của Lastingtrip là cầu nối giữa Model và View, chịu trách nhiệm xử lý logic ứng dụng. Khi nhận yêu cầu từ người dùng, Controller gọi dữ liệu từ Model, xử lý thông tin và trả kết quả về View để hiển thị. Các file Controller được tổ chức trong thư mục controllers, mỗi file đảm nhận một chức năng cụ thể như quản lý người dùng, khách sạn, phòng, mã giảm giá, đánh giá, thanh toán, và các tài nguyên liên quan. Nhờ sự phân chia rõ ràng, Controller giúp hệ thống dễ dàng mở rộng và duy trì.

c) Thiết kế bảo mật trên các thành phần của mô hình

- Để đảm bảo an toàn và bảo mật cho kiến trúc hệ thống, cần phải xây dựng các cơ chế và nguyên tắc bảo mật cho từng thành phần trong hệ thống.
- Chúng em đã xây dựng các nguyên tắc bảo mật cho từng khối như: database, view, api, và controller. Chi tiết về các nguyên tắc này sẽ được trình bày ở các phần tiếp theo.
- Xây dựng các middleware với mục đích phân quyền và thiết lập các chính sách bảo mật cho việc truy cập các thành phần trong hệ thống

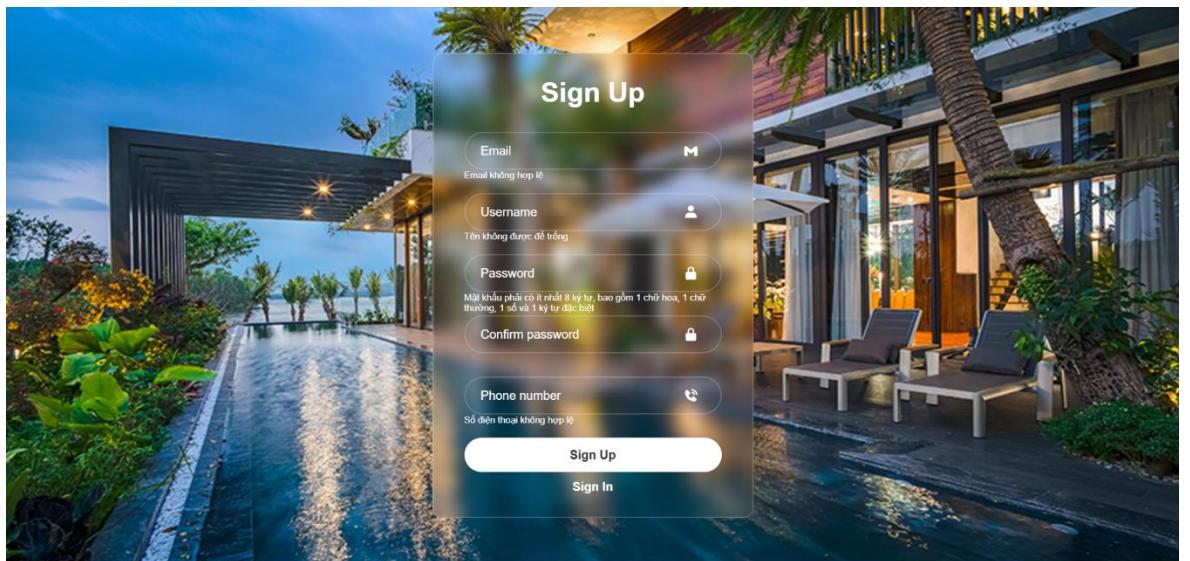
3. Các chức năng của ứng dụng.

d) Đăng ký, đăng nhập, quên mật khẩu

- Đăng ký:

- Người dùng nhập thông tin vào để đăng ký như email, username, password, số điện thoại

- Thông tin phải đúng và hợp lệ
- Sau khi đăng ký, hệ thống sẽ gửi OTP xác nhận người dùng đó là người dùng thật và chuyển sang trang đăng nhập.



Hình 2. Giao diện trang đăng kí



Xác Nhận Đăng Ký Tài Khoản

Hộp thư đến x



gklathumon@gmail.com

Mã Xác Nhận Đăng Ký Mã OTP của bạn là: 289685 Mã này sẽ hết hạn sau 10 phút



gklathumon@gmail.com

đến tôi ▾

Mã Xác Nhận Đăng Ký

Mã OTP của bạn là: **116180**

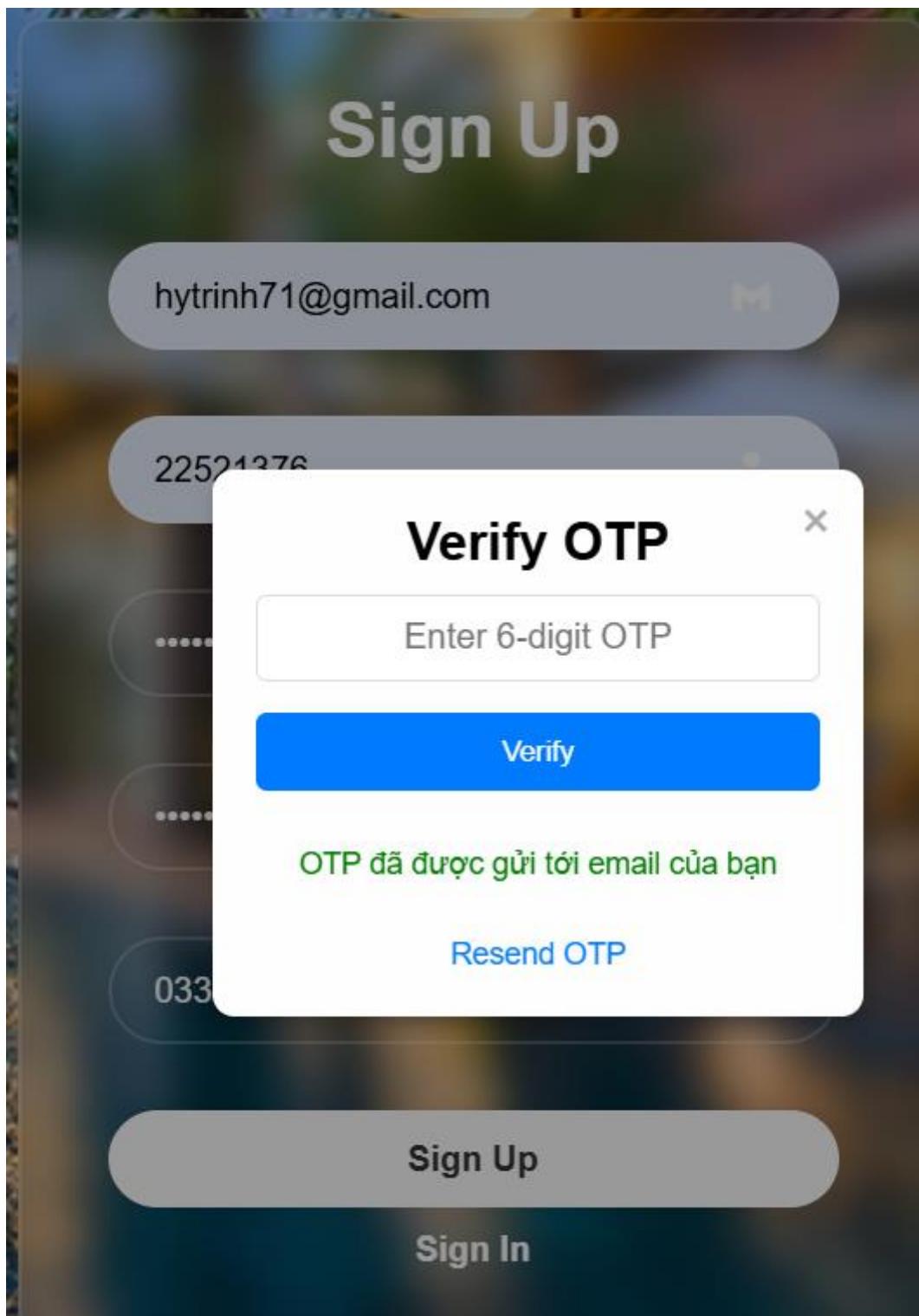
...

← Trả lời

→ Chuyển tiếp



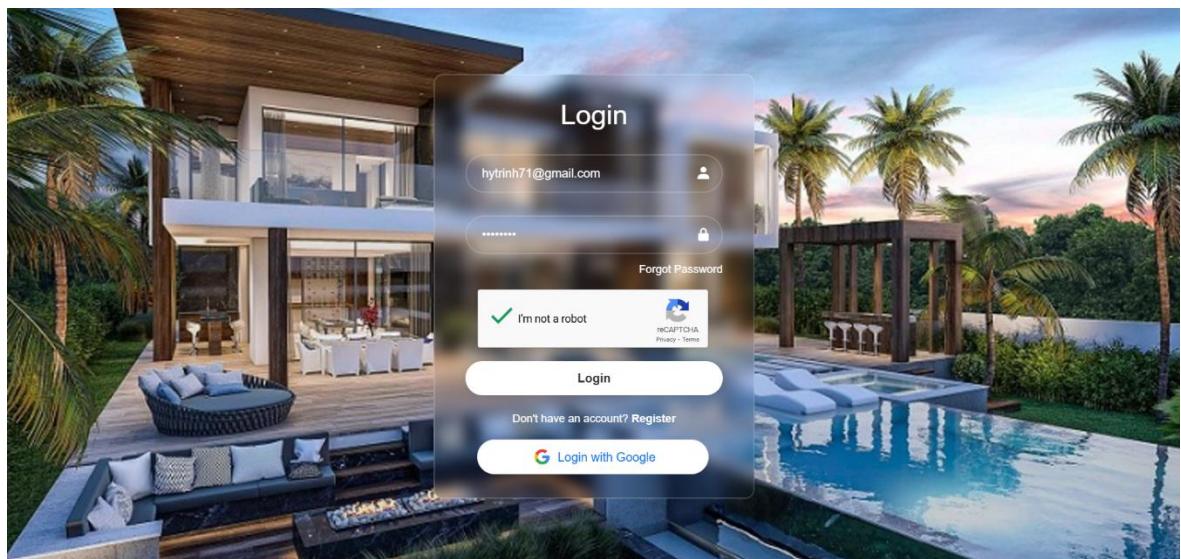
Hình 3. Mã OTP được gửi về mail của người dùng



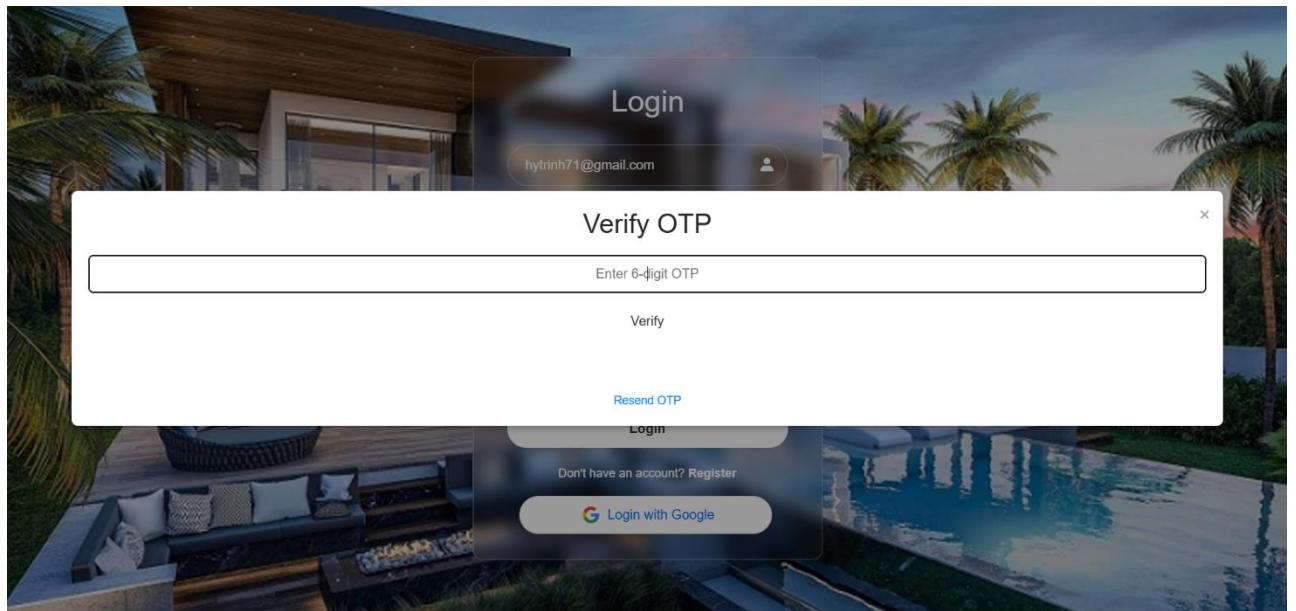
Hình 4. Tiến hành xác thực OTP

- **Đăng nhập:** Có 2 phương thức để đăng nhập, email hoặc sử dụng tài khoản google

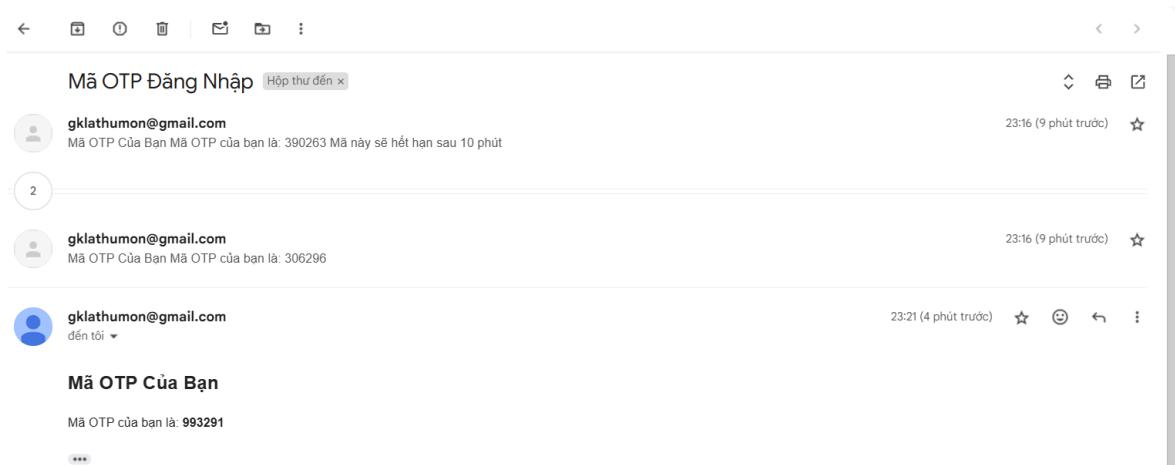
- Người dùng đăng nhập bằng tài khoản đã đăng ký và tick vào CAPTCHA để xác nhận không phải là robot.
- Sau khi bấm login, sẽ hiện ra một popup để người dùng nhập mã OTP gửi về email để xác nhận người đăng nhập chính chủ, có thể yêu cầu gửi lại mã OTP nếu hết hạn. Nếu người dùng nhập sai quá 5 lần mã OTP hiện hành sẽ bị vô hiệu hóa và yêu cầu người dùng thực hiện lại quá trình điền thông tin đăng nhập để nhận được 1 phiên OTP mới.



Hình 5. Tiết hành đăng nhập

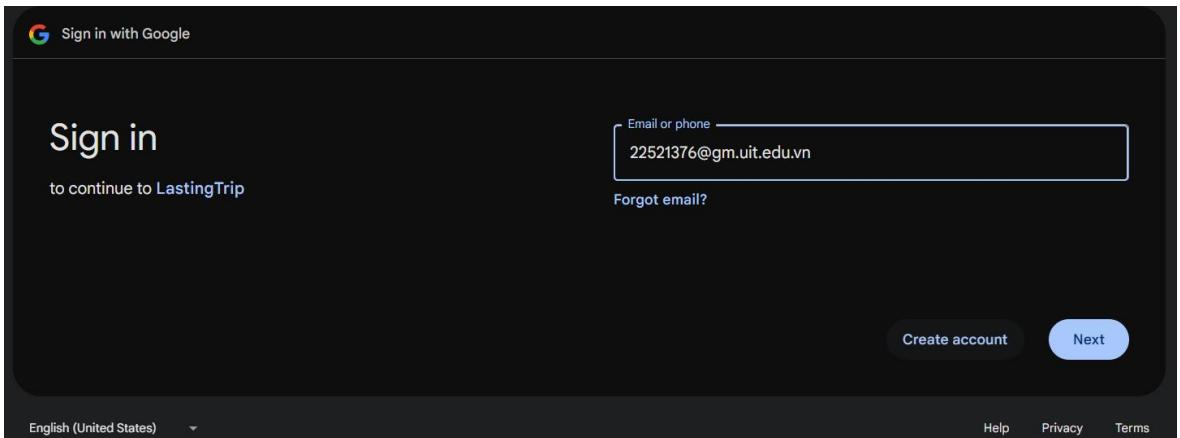


Hình 6. Sau khi đăng nhập sẽ yêu cầu xác thực OTP

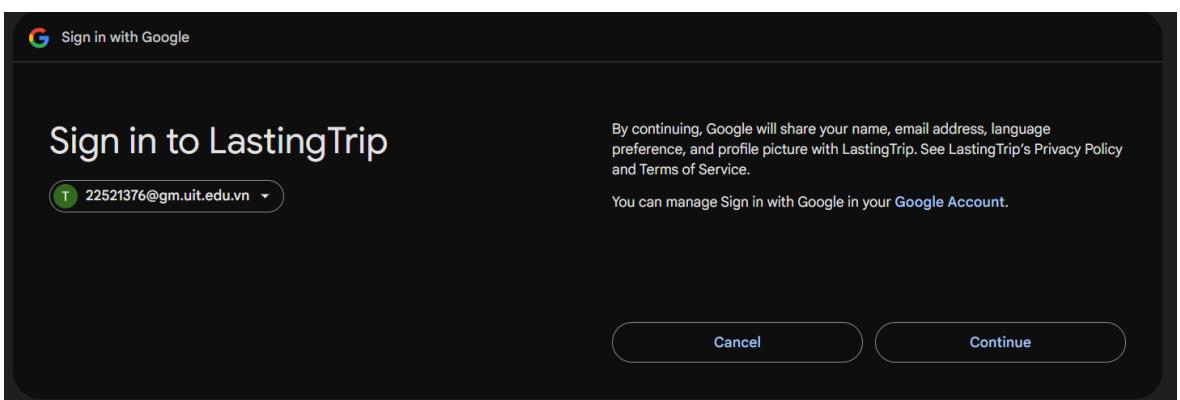


Hình 7. OTP được gửi về mail người dùng

- Khi người dùng đăng nhập bằng tài khoản Google, thì phải nhập đúng thông tin để đăng nhập.



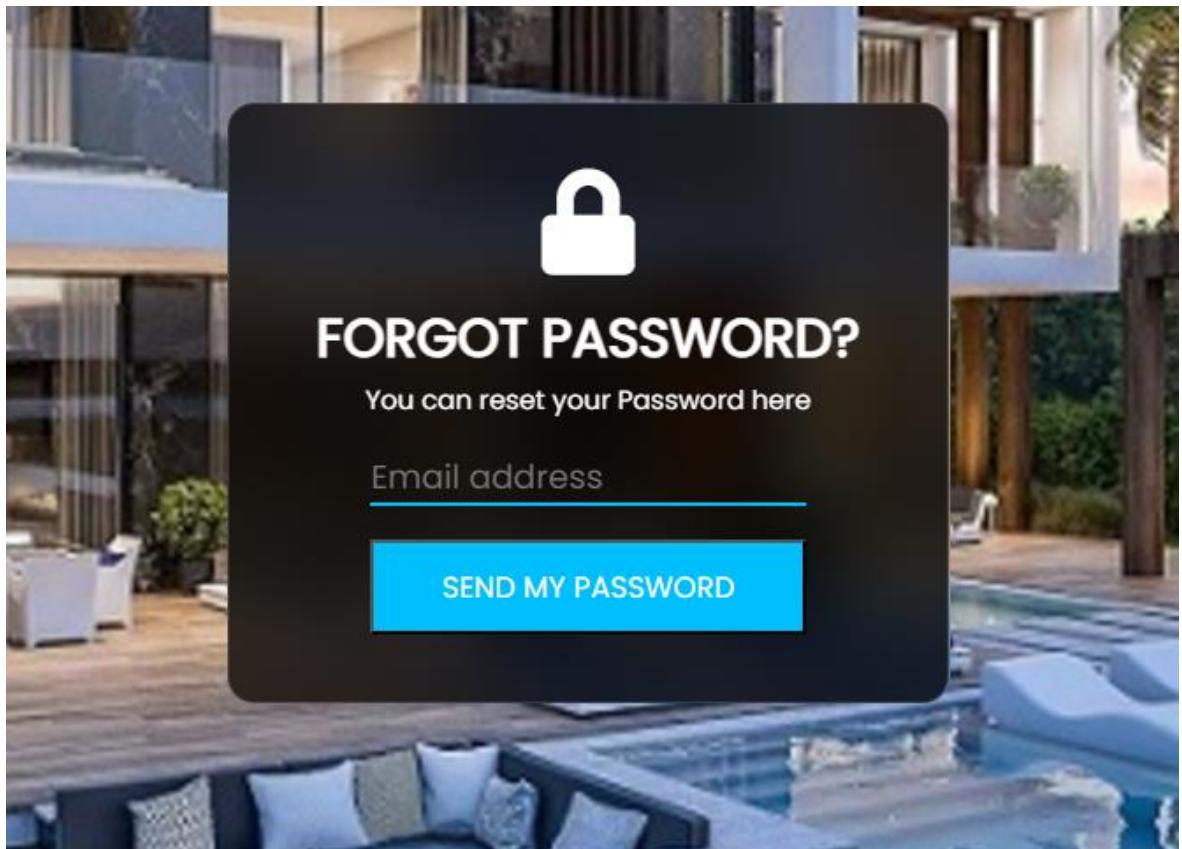
Hình 8. Nhập thông tin tài khoản google.



Hình 9. Xác nhận thông tin google

- **Quên mật khẩu:**

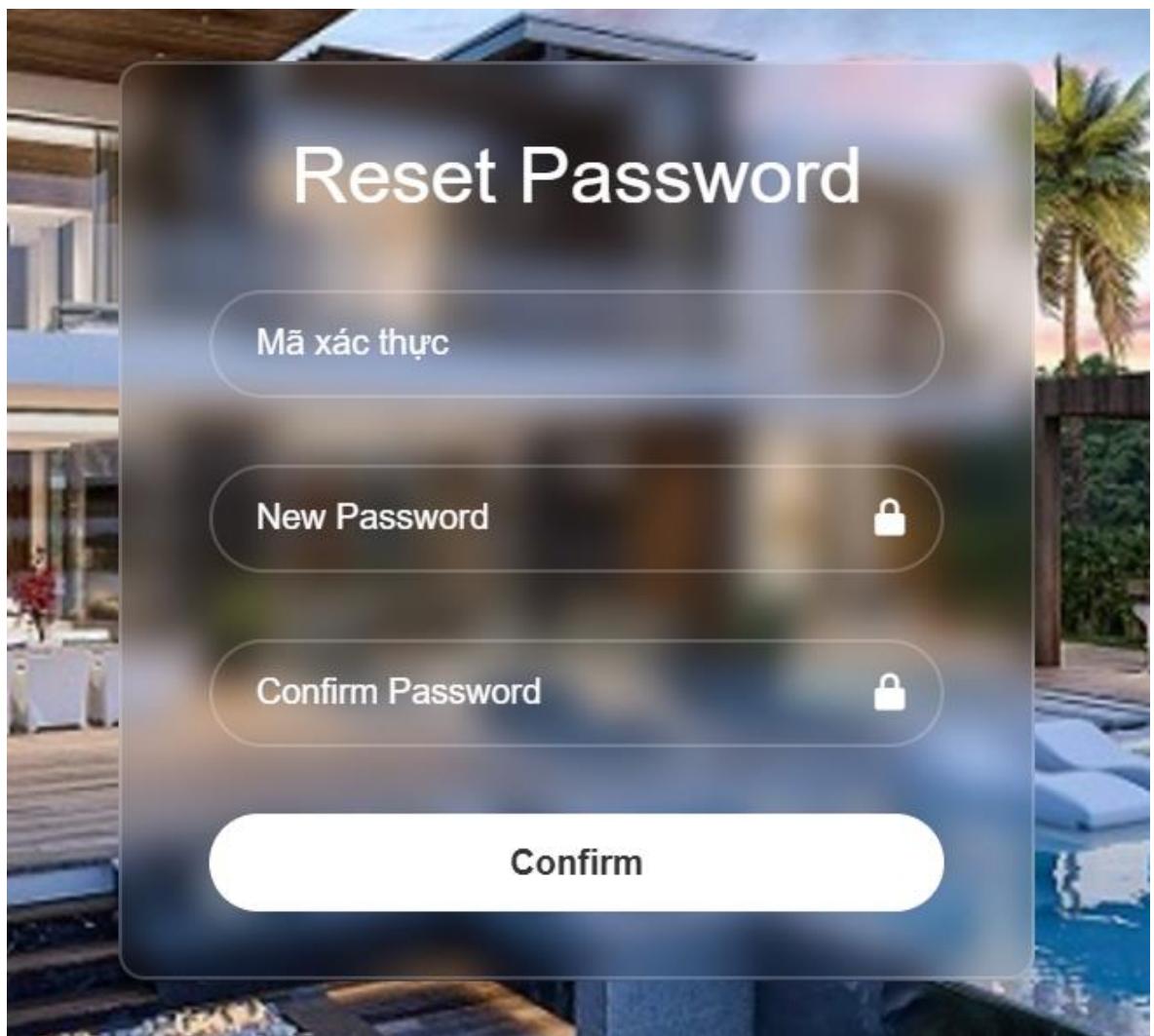
- Khi người dùng quên mật khẩu, có thể nhập lại email đã đăng ký để lấy lại mật khẩu
- Sau khi nhập email, một mã xác thực sẽ được gửi qua email vừa nhập, người dùng cần nhập mã xác thực đó để tạo lại mật khẩu mới



Hình 10. Chức năng quên mật khẩu



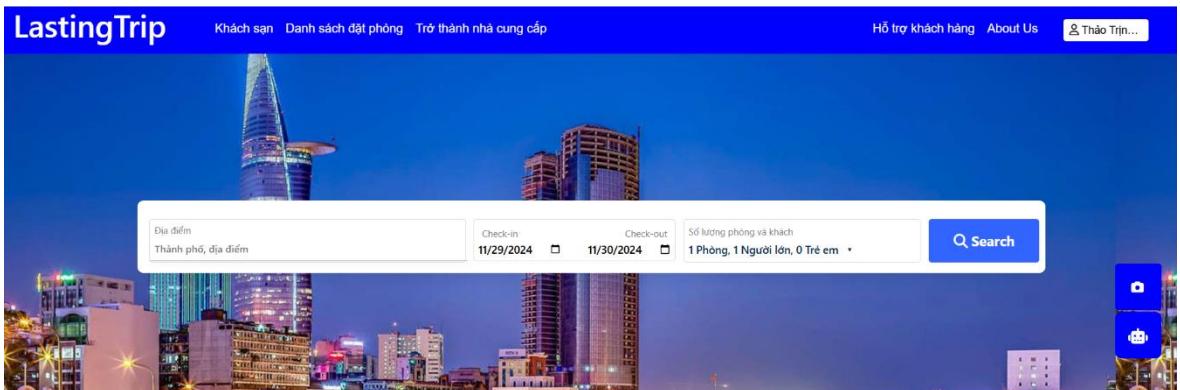
Hình 11. Thông tin được trả về mail



Hình 12. Tiến hành đổi mật khẩu

e) *Search*

- Nhập vào địa điểm, ngày check in, ngày check out, số lượng người để tìm khách sạn phù hợp



Hình 13. Giao diện của thanh Search

Hình 14. Kết quả trả về khi tìm kiếm

f) Filter

- Lọc các dữ liệu theo từng tiêu chí mà người dùng chọn

Hình 15. Filter theo nhu cầu của người

g) Chatbot

- Sử dụng các API có sẵn để tạo AI học dữ liệu của trang web, trợ giúp người dùng trong việc sử dụng web dễ hơn

Hình 16. Chức năng ChatBot

h) Tìm kiếm địa điểm từ ảnh

- Sử dụng TeachableMachine để train model đơn giản về việc phân loại ảnh để tạo thành chatbot hình ảnh.

- Chatbot hình ảnh có tích hợp logic Search, để tự động tìm kiếm khách sạn phù hợp với địa điểm mà người dùng muốn.



Hình 17. Chức năng tìm kiếm địa điểm qua hình ảnh

The screenshot shows a search interface for finding suitable rooms. The search parameters are set to check-in on 11/29/2024 and check-out on 11/30/2024, with one room and one adult selected. The results show two options:

- La Villa Da Lat** (★★★) - Standard Room: VND 138,396. Description: Although dark nearly to oil personal main. People understand event open training. Address: C21 KQH nghiên cứu du lịch Hùng Vương, Phường 11, Khu Chí Lăng, Đà Lạt, Lâm Đồng. [Xem bản đồ](#)
- California Hotel** (★★★) - Triple Room: VND 3,221,368. Description: Particular artist blood animal store available. Both specific quickly shoulder quickly rich dog. Keep seven their stuff enjoy ever in. Address: A28 Nguyễn Hữu Cánh, phường 8, Thành phố, Phường 8, Đà Lạt, Lâm Đồng. [Xem bản đồ](#)

Hình 18. Các phòng khách sạn phù hợp với địa điểm

i) Đặt phòng

- Sau khi người dùng tìm kiếm được phòng khách sạn với ngày ưng ý, người dùng bấm vào nút đặt phòng.
- Khi người dùng đã đăng nhập và được server xác thực sẽ chuyển đến thanh toán

Family Room

Lựa chọn của bạn	Số lượng người	Giá phòng hôm nay
Giá tốt nhất ngày! Bữa sáng: VND 1,500,427.4 Không hoàn tiền Xác nhận nhanh chóng Đặt tối đa 2 phòng	2 người	VND 7,502,137
Đặt phòng		

Twin Room

Lựa chọn của bạn	Số lượng người	Giá phòng hôm nay
Giá tốt nhất ngày! Bữa sáng: VND 520,755.6 Không hoàn tiền Xác nhận nhanh chóng Đặt tối đa 8 phòng	4 người	VND 2,603,778
Đặt phòng		

Hình 19. Chức năng đặt phòng để chuyển tới trang thanh toán

Sau khi ấn nút đặt phòng, trang thanh toán sẽ hiện ra.

j) Thanh toán

Trang thanh toán của bạn được thiết kế để giúp quá trình đặt dịch vụ trở nên nhanh chóng và tiện lợi. Tại đây, bạn chỉ cần điền đầy đủ thông tin cá nhân bao gồm họ tên, năm sinh, số căn cước công dân, địa chỉ, số điện thoại, và email. Ngoài ra, bạn có thể ghi chú các yêu cầu đặc biệt nếu cần.

Bên phải là phần tóm tắt đơn hàng, hiển thị thông tin khách sạn, ngày nhận và trả phòng, số lượng phòng đặt, cùng với tổng số tiền phải thanh toán. Nếu bạn có mã giảm giá, hãy nhập để tiết kiệm chi phí.

Thanh toán

Họ tên *	Thao
Năm sinh*	2004
Căn cước công dân *	09875764764
Địa chỉ	HCM
Số điện thoại*	0329771150
Email Address *	thanhptt133@gmail.com
Yêu cầu đặc biệt	k
Đơn hàng của bạn	
Khách sạn Dalat Palace Heritage 8,017,829 VND	
Từ: 2024-12-07	Đến: 2024-12-09
Số lượng phòng	1
Mã giảm giá	<input type="text"/>
Apply	
Tổng cộng 24,053,487 VND	
<input checked="" type="radio"/> Thanh toán qua ngân hàng	
Place Order	

Hình 20. Trang thanh toán trước điền coupon

Thanh toán

Họ tên *	Thao
Năm sinh*	2004
Căn cước công dân *	09875764764
Địa chỉ	HCM
Số điện thoại*	0329771150
Email Address *	thanhptt1337@gmail.com
Yêu cầu đặc biệt	k

Đơn hàng của bạn

Khách sạn Dalat Palace Heritage	8,017,829 VND
Từ: 2024-12-07	Đến: 2024-12-09
Số lượng phòng	1
Mã giảm giá	AOTHATDAY

Apply

Tổng cộng	21,648,138.30 VND
-----------	-------------------

Thanh toán qua ngân hàng

Place Order

Hình 21. Trang thanh toán khi áp dụng coupon

Khi nhấn "**Place Order**", hệ thống sẽ dẫn bạn đến trang "**Phương Thức Thanh Toán**", nơi bạn có thể chọn ngân hàng phù hợp để hoàn tất giao dịch. Danh sách các ngân hàng uy tín như Vietcombank, BIDV, Agribank, Techcombank, và nhiều ngân hàng khác đều có sẵn để bạn lựa chọn.

Ở bên phải giao diện, thông tin chi tiết về đơn đặt phòng của bạn sẽ được hiển thị đầy đủ, bao gồm:

- **Tên khách sạn:** Cụ thể với từng đặt phòng.
- **Loại phòng:** Hiển thị đúng hạng phòng bạn đã chọn.
- **Thời gian lưu trú:** Ghi rõ ngày nhận và trả phòng.
- **Tổng số tiền:** Số tiền chính xác cần thanh toán, bao gồm thuế và phí.
- **Thông tin khách hàng:** Họ tên, email, và các thông tin liên quan.

Sau khi chọn ngân hàng, bạn sẽ được chuyển đến cổng thanh toán của ngân hàng để hoàn tất giao dịch. Tất cả quá trình đều được mã hóa và bảo mật nhằm mang lại sự an toàn cho bạn.

Chúng tôi cam kết mang lại trải nghiệm đặt phòng dễ dàng, nhanh chóng và an toàn, đảm bảo chuyến đi của bạn được chuẩn bị một cách hoàn hảo nhất!

Phương Thức Thanh Toán

Danh sách ngân hàng

-  **Vietcombank**
Ngân hàng VietcomBank
-  **VietinBank**
Ngân hàng Vietinbank
-  **BIDV**
Ngân hàng BIDV
-  **Agribank**
Ngân hàng Agribank
-  **Sacombank**
Ngân hàng Sacombank
-  **TECHCOMBANK**
Ngân hàng Techcombank
-  **MB**
Ngân hàng MBBank
-  **ACB**
Ngân hàng ACB
-  **VPBank**
Ngân hàng VPBank
-  **DONGA Bank**
Ngân hàng Đông Á
-  **SHB**
Ngân hàng SHB
-  **EXIMBANK**
Ngân hàng EximBank - OMNI
-  **EXIMBANK**
Ngân hàng EximBank
-  **TPBank**
Ngân hàng TPBank
-  **NCB**
Ngân hàng NCB

Đơn hàng của bạn	
InterContinental Phu Quoc Long Beach Resort	Loại phòng Suite
Từ: 07/12/2024	Đến: 09/12/2024
Tổng cộng	34,628,232 VND
Thông tin khách hàng:	
Trung Thành thanhpptt1337@gmail.com	

[Place Order](#)

Hình 22. Trang phương thức thanh toán để lựa chọn ngân hàng

Trang thanh toán được thiết kế nhằm giúp bạn hoàn tất giao dịch một cách nhanh chóng, an toàn và thuận tiện. Tại đây, bạn sẽ được cung cấp đầy đủ thông tin về đơn hàng, bao gồm:

- **Số tiền thanh toán:** Tổng số tiền cần thanh toán, bao gồm giá trị sản phẩm/dịch vụ và bất kỳ khoản phí liên quan.
- **Thông tin đơn hàng:** Mã đơn hàng và các chi tiết liên quan để bạn dễ dàng kiểm tra, theo dõi.
- **Hỗ trợ khách hàng:** Thông tin liên hệ (hotline và email) sẵn sàng giải đáp mọi thắc mắc của bạn.

Thông tin do vnappy sandbox cung cấp để kiểm thử thanh toán

Số thẻ	9704198526191432198
Tên chủ thẻ	NGUYEN VAN A
Ngày phát hành	07/15
Mật khẩu OTP	123456

 CÔNG THANH TOÁN
VNPayQR

Giao dịch hết hạn sau **14 : 09**

Thông tin đơn hàng (Test)

Số tiền thanh toán
34.628.232VND

Giá trị đơn hàng
34.628.232VND

Phí giao dịch
0VND

Mã đơn hàng
17

Nhà cung cấp
<https://vnshop.vn/>

Thanh toán qua Ngân hàng NCB

Thẻ nội địa

Số thẻ
*****2198 

Tên chủ thẻ
NGUYEN VAN A

Ngày phát hành  07/15 

Mã khuyến mại Chọn hoặc nhập mã

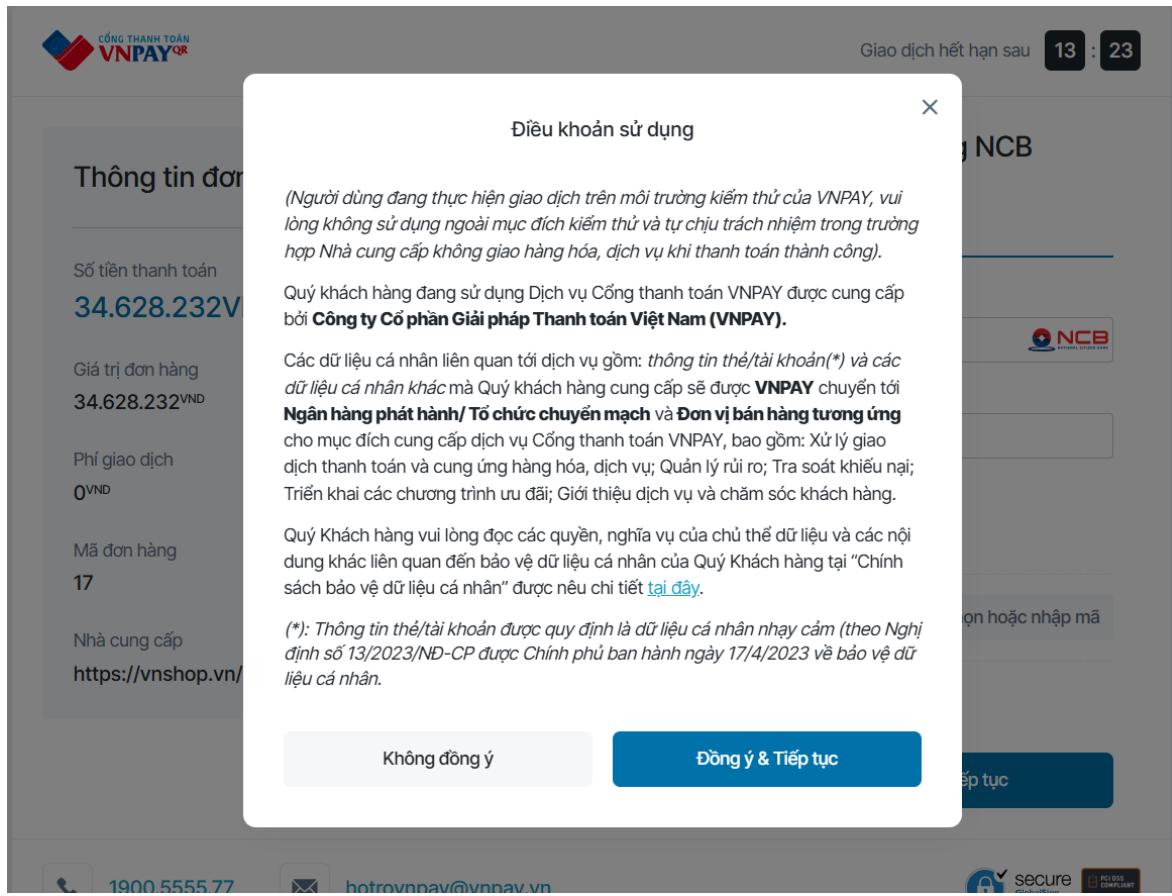
[Điều kiện sử dụng dịch vụ](#)

Hủy thanh toán **Tiếp tục**

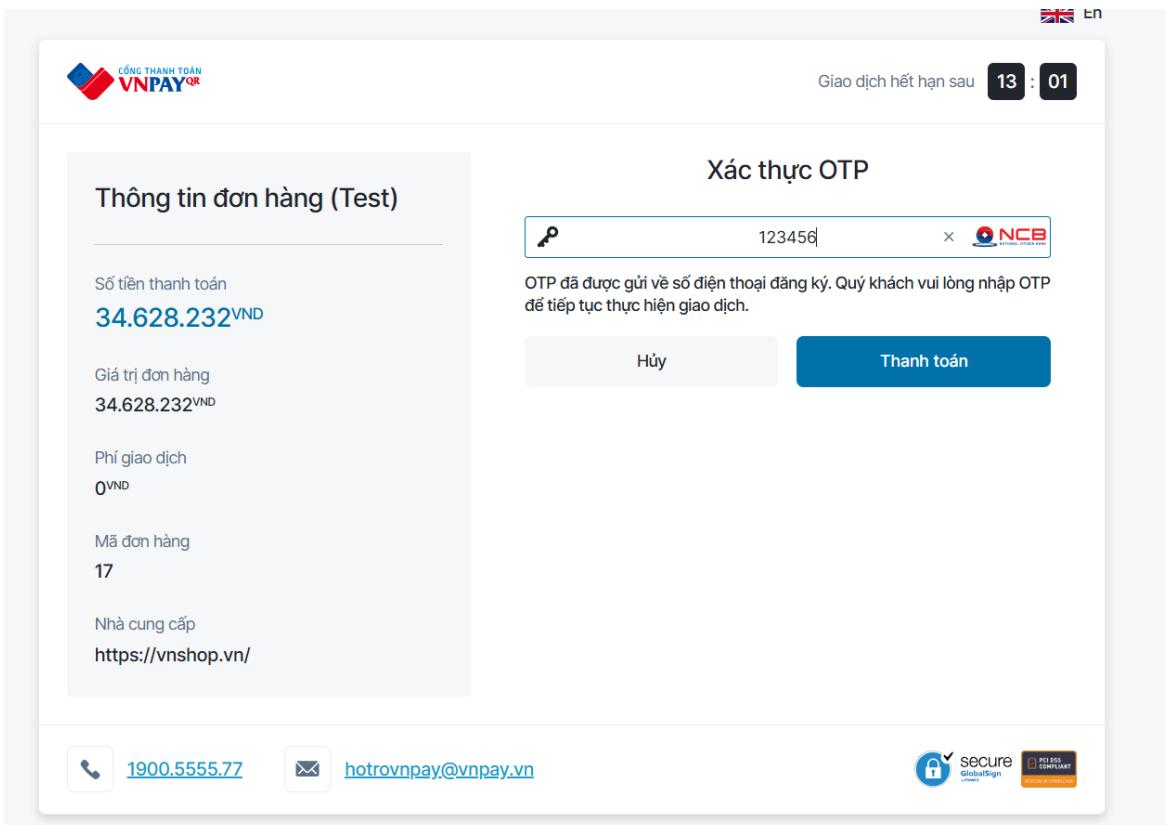
 [1900.5555.77](tel:1900.5555.77)  hotrovnpay@vnpay.vn

 secure
GlobalSign
PIPED COMPLIANT
EV SSL CERTIFICATE

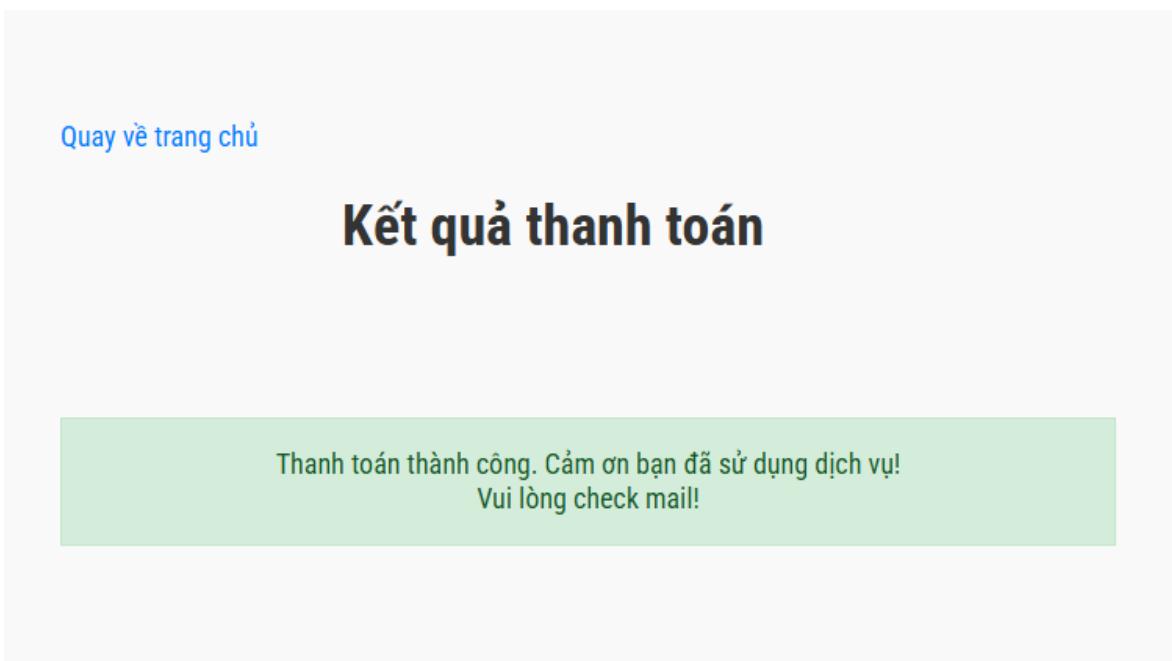
Hình 23. Trang thanh toán của vnpay



Hình 24. Chính sách và điều khoản của vnpay



Hình 25. Trang xác thực OTP



Hình 26. Trang thanh toán thành công

Sau khi thanh toán hoàn tất, người dùng sẽ nhận được mail xác nhận nếu đã thanh toán thành công



gklathumon@gmail.com
đến tôi ▾

Thanh toán thành công

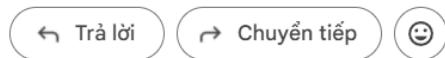
Chào thuan thuan,

Đơn hàng của bạn với mã số **14725796** đã được thanh toán thành công.

Chi tiết:

- **Ngày đặt:** Sat Dec 07 2024 07:00:00 GMT+0700 (Indochina Time)
- **Tổng tiền:** 88533405 VND
- **Khách sạn:** JW Marriott Hotel Hanoi
- **Phòng:** Grand Suite
- **Loại giường:** Three Single

Cảm ơn bạn đã sử dụng dịch vụ của chúng tôi!



Hình 27. Thư xác nhận đã đăng ký phòng thành công

k) Bình luận

- Người dùng có thể đánh giá sao và chia sẻ những trải nghiệm của mình để review cho những người dùng khác tham khảo.

Đánh giá số sao

Rất thích hợp cho người
thư giãn. Rất tuyệt vời.
Mười điểm

Nội dung

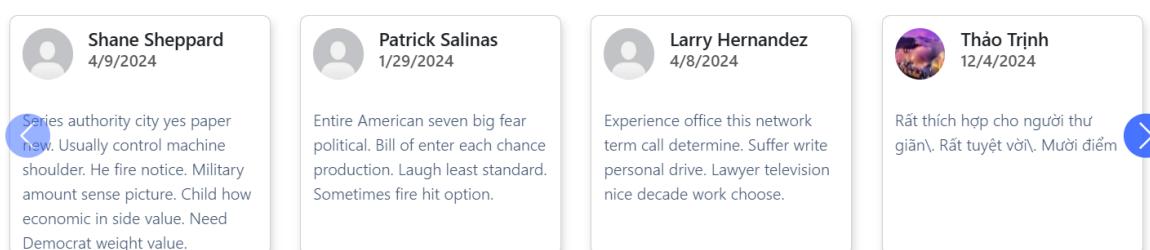
Hình ảnh

Choose File du-lich-Da-Lat-ivivu1.jpg



Gửi feedback

Hình 28. Các chức năng đánh giá và comment



Hình 29. Đánh giá được hiển thị tới các người dùng khác

l) Quản lý khách sạn và các thông tin khác của hệ thống

-Admin sẽ quản lý thông tin của các khách sạn đang tồn tại trên hệ thống. Người quản trị có thể chỉnh sửa các thông tin của khách sạn cũng như thông tin về chi tiết phòng và dịch vụ. Ngoài ra quản trị viên còn có thể quản lý thông tin về các đơn đặt hàng, mã giảm giá cũng như thông kê về thông tin của hệ thống.

ID	TÊN	SỐ SAO	ĐỊA CHỈ	LOẠI HÌNH	GIÁ	HÌNH THỨC THANH TOAN	Chú ý hữu	CHỨC NĂNG	HÌNH ANH
1	Holiday Inn & Suites Saigon Airport	5	TP.HCM	Hotel	149510	online	2	<button>Chỉnh sửa</button> <button>Xóa</button> <button>Phòng</button> <button>Dịch Vụ</button>	
2	LA VELA SAIGON HOTEL	5	TP.HCM	Hotel	5120972	online	3	<button>Chỉnh sửa</button> <button>Xóa</button> <button>Phòng</button> <button>Dịch Vụ</button>	
3	Sakura Hostel Saigon	4	TP.HCM	Hotel	3091487	online	4	<button>Chỉnh sửa</button> <button>Xóa</button> <button>Phòng</button> <button>Dịch Vụ</button>	
4	LANGUAGE EXCHANGE HOTEL 3	3	TP.HCM	Hotel	3885219	online	5	<button>Chỉnh sửa</button> <button>Xóa</button> <button>Phòng</button> <button>Dịch Vụ</button>	
5	Ehome Saigon	2	TP.HCM	homestay	146103	online	6	<button>Chỉnh sửa</button> <button>Xóa</button> <button>Phòng</button> <button>Dịch Vụ</button>	
6	SEA QUEEN HOTEL	3	Đà Nẵng	Hotel	2603778	online	7	<button>Chỉnh sửa</button> <button>Xóa</button> <button>Phòng</button> <button>Dịch Vụ</button>	

Hình 30. Trang quản lý khách sạn của admin

Admin có quyền quản lý các tiện ích của mỗi khách sạn như có thể thêm, xóa, sửa nhằm giúp admin quản lý trang chặt chẽ hơn

STT	TÊN	CHỨC NĂNG
1	Cafes	<button>Chỉnh sửa</button> <button>Xóa</button>
2	Bể bơi ngoài trời	<button>Chỉnh sửa</button> <button>Xóa</button>
3	Bar	<button>Chỉnh sửa</button> <button>Xóa</button>
4	Bãi đậu xe riêng	<button>Chỉnh sửa</button> <button>Xóa</button>
5	Spa	<button>Chỉnh sửa</button> <button>Xóa</button>
6	Wifi công cộng	<button>Chỉnh sửa</button> <button>Xóa</button>

Hình 31. Trang quản lý các tiện nghi của khách sạn

Bên cạnh đó admin, có thể quản lý các yêu cầu đặt khách sạn, ở đây admin có thể xóa các đơn đặt hàng nếu phát hiện các hành vi khả nghi

QUẢN LÝ ĐƠN ĐẶT HÀNG

STT	ID PHÒNG	HỌ TÊN	ID NGƯỜI DÙNG	TỔNG TIỀN	TRẠNG THÁI	NGÀY CHECK-IN	NGÀY CHECK-OUT	YÊU CẦU ĐẶC BIỆT	CHỨC NĂNG
1	1	Antonio Goodwin	1	10000	Đã thanh toán	1936-09-12	1936-09-12	Don	Xóa
3	5	Trịnh Thảo	34	11872146	Đã thanh toán	2024-12-10	2024-12-11	hihi	Xóa
4	5	Trịnh Thảo	34	11872146	Chưa thanh toán	2024-12-10	2024-12-11	k	Xóa
5	1	Trịnh Thảo	34	15036660	Chưa thanh toán	2024-12-10	2024-12-11	n	Xóa
6	1	Trịnh Thảo	34	15036660	Chưa thanh toán	2024-12-10	2024-12-11	n	Xóa
7	1	Trịnh Thảo	34	15036660	Đã thanh toán	2024-12-10	2024-12-11	k	Xóa
8	37	Thao	39	15004274	Đã thanh toán	2024-12-10	2024-12-11	k	Xóa
9	2	Thanh	40	3319470	Đã thanh toán	2024-12-10	2024-12-11	notion	Xóa
10	133	Thanh	41	6442736	Chưa thanh toán	2024-12-10	2024-12-11	no	Xóa
11	133	Thanh	41	6442736	Chưa thanh toán	2024-12-10	2024-12-11	no	Xóa

Hình 32. Trang quản lý đơn đặt hàng

m) Quản lý tài khoản người dùng.

- Admin có thể quản lý thông tin về các user tồn tại trên hệ thống. Người quản trị có quyền xóa, sửa thông tin của các user

ID	TÊN	EMAIL	SDT	LOẠI	CHỨC NĂNG
1	Cindy Kemp	dpaul@example.org	0333	owner	<button>Chỉnh sửa</button> <button>Xóa</button>
2	Alan Sosa	weberfelicia@example.net	(551)388-7441	owner	<button>Chỉnh sửa</button> <button>Xóa</button>
3	Patrick Salinas	teresa20@example.net	+1-930-416-9932x8701	owner	<button>Chỉnh sửa</button> <button>Xóa</button>
4	James Powell	cantujessica@example.com	001-895-874-1649x48143	owner	<button>Chỉnh sửa</button> <button>Xóa</button>
5	Todd Doyle	jacksonleslie@example.net	(436)366-5012x1223	owner	<button>Chỉnh sửa</button> <button>Xóa</button>
6	Jason Padilla	edward777@example.org	941-787-6142x27369	owner	<button>Chỉnh sửa</button> <button>Xóa</button>
7	Joshua Cox	timothybruce@example.net	+1-387-906-6384x98629	owner	<button>Chỉnh sửa</button> <button>Xóa</button>
8	Harold Gonzales	lmullins@example.com	(263)785-2213	owner	<button>Chỉnh sửa</button> <button>Xóa</button>
9	Daniel Williams	johncarter@example.org	621.422.6807x30549	owner	<button>Chỉnh sửa</button> <button>Xóa</button>
10	Shane Sheppard	brittanycastillo@example.com	245-891-8228x74866	owner	<button>Chỉnh sửa</button> <button>Xóa</button>
11	Justin Fleming	pgonzalez@example.com	001-208-466-6307x4503	owner	<button>Chỉnh sửa</button> <button>Xóa</button>
12	Jonathan Reyes	jeremysnyder@example.com	224.854.8513x2823	owner	<button>Chỉnh sửa</button> <button>Xóa</button>

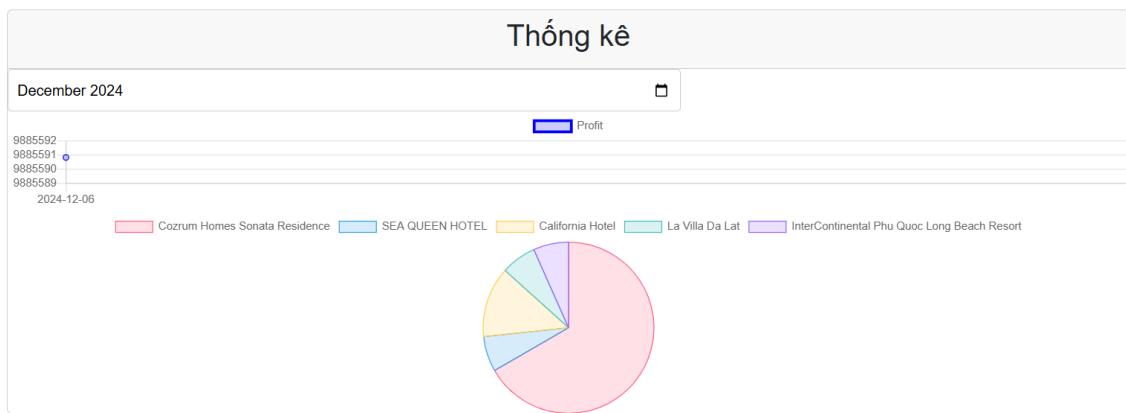
Hình 33. Quản lý danh sách người dùng

Admin có quyền chỉnh sửa thông tin người dùng để quản lý người dùng nếu có sai sót

The screenshot shows a web application interface for managing users. On the left, there's a sidebar with navigation links: NGƯỜI DÙNG, KHÁCH SẠN, ĐƠN ĐÁT HÀNG, THỐNG KÊ, MÃ GIẢM GIÁ, and ĐĂNG XUẤT. The main area is titled 'QUẢN LÝ NGƯỜI DÙNG' (User Management) and displays a table with columns: ID, TÊN (Name), EMAIL, SDT (Phone Number), LOẠI (Type), and CHỨC NĂNG (Functions). A modal window titled 'Cập nhật' (Update) is open over the table, containing fields for Name, Email, Phone Number, and User Type, along with a 'Cập nhật' (Update) button.

Hình 34. *Chỉnh sửa thông tin người dùng*

Trang thống kê giúp admin có thể biết được, lợi nhuận của trang web



Hình 35. *Trang biểu đồ thống kê*

Admin có thể thêm xóa sửa mã giảm giá để tiện cho việc cấp phát mã giảm giá cho người dùng

PHẦN TRĂM SỐ LƯỢNG NGÀY BẮT ĐẦU

10 2024-12-10

Thêm mã giảm giá

Code

Phần trăm

Số lượng

Ngày bắt đầu

mm/dd/yyyy

Ngày kết thúc

mm/dd/yyyy

Add

Hình 36. Thêm mã coupon

QUẢN LÝ MÃ GIẢM GIÁ						Thêm mã giảm giá
STT	CODE	PHẦN TRĂM	SỐ LƯỢNG	NGÀY BẮT ĐẦU	NGÀY KẾT THÚC	Chỉnh sửa/Xóa
1	AOTHATDAY	10	6	2024-12-10	2024-12-13	Xóa

Hình 37. Quản lý mã giảm giá

B. CHÍNH SÁCH BẢO MẬT

4. Hệ thống xác thực và phân quyền người dùng.

n) Các vai trò trong ứng dụng.

- Quyền hạn của người dùng được phân loại tùy theo vai trò của người dùng đó thông qua token.
- Chúng em chia thành ba loại người dùng chính:
 - **Admin:** Quản trị viên, người quản trị hệ thống, có toàn bộ quyền của hệ thống. Quản trị viên có thể quản lý người dùng, quản lý khách sạn và các thành phần khác liên quan. Quản trị viên còn có thể xem được thống kê doanh thu đạt được trong tháng.

```
// Middleware để kiểm tra vai trò admin
Tabnine | Edit | Test | Explain | Document | Ask | Qodo Gen: Options | Test this function
function requireAdmin(req, res, next) {
  if (req.user.type !== 'admin') {
    return res.status(403).json({ message: 'Vui lòng đăng nhập tài khoản quản lý để thực hiện.' });
  }
  next();
}
```

Hình 38. Middleware phân quyền Admin

- **Owner:** Chủ khách sạn, có ít nhất một khách sạn được đăng ký trong hệ thống. Chủ khách sạn có thể quản lý khách sạn của mình.

```
Tabnine | Edit | Test | Explain | Document | Ask | Qodo Gen: Options | Test this function
function requireOwner(req, res, next) {
  if (req.user.type !== "owner") {
    return res
      .status(403)
      .json({ message: "Vui lòng đăng nhập tài khoản Owner để thực hiện." });
  }
  next();
}
```

Hình 28. Middleware phân quyền Owner

- **Client:** Khách hàng với mục đích sử dụng chính là đặt phòng khách sạn. Khi đăng nhập bằng Google, người dùng chỉ có thể là client.

```
// Middleware để kiểm tra vai trò admin
Tabnine | Edit | Test | Explain | Document | Ask | Qodo Gen: Options | Test this function
✓ function requireCustomer(req, res, next) {
✓   if (req.user.type !== "client") {
✓     return res
✓       .status(403)
✓       .json({
✓         message: 'Vui lòng đăng nhập tài khoản "Khách hàng" để thực hiện.',
✓       });
✓   }
✓   next();
}
```

Hình 39. Middle kiểm tra user thường

- Mỗi người dùng sẽ tùy vào loại người dùng để khi đăng nhập thành công sẽ được cấp đúng quyền hạn của mình.

o) Access Token và Refresh Token

- Access Token:

- Là một token được generate bằng các thông tin của người dùng bao gồm id và type hỗ trợ quá trình phân quyền được lưu trong Cookie sau khi người dùng đăng nhập thành công, được sử dụng để xác thực danh tính người dùng và phân quyền người dùng khi họ thực hiện các request đến server.

```
const isAuthen = bcrypt.compareSync(password, user.password);

if (isAuthen) {
  res.cookie("accessToken", accessToken, { httpOnly: true });
  console.log("refreshToken", refreshToken);
  res.status(200).send({
    message: "successful",
    type: user.type,
    id: user.id,
    token: refreshToken,
    refreshToken: refreshToken,
  });
}
```

Hình 40. Lưu accestoken vào cookies

- Chúng em dùng jwt và private key (ACCESS_TOKEN) để kí accessToken với những thành phần như ID người dùng (userId), tên người dùng (name), loại người dùng (type).
- Token này có thời gian sống trong 40 phút nhằm giảm nguy cơ bị lạm dụng nếu bị lộ, Khi hết hạn, token sẽ được làm mới thông qua Refresh Token.

```
const accessToken = jwt.sign(
  { userId: user.id, type: user.type },
  process.env.ACCESS_TOKEN,
  { expiresIn: "40m" }
);
```

Hình 41. Tiến hành tạo accessToken

- Token này được sử dụng trong các yêu cầu tới API để xác minh và xử lý yêu cầu

```
async function getCurrentUser() {
  try {
    const response = await fetch("/api/v1/users/getCurrentUser", {
      method: "GET",
      credentials: "include", // Trong credentials: include, nếu có cookie thì sẽ tự động gửi kèm cookie,
                            // trong đó có tồn tại Access Token
    });

    if (!response.ok) {
      const errorText = await response.text();
      throw new Error(`Failed to fetch current user: ${errorText}`);
    }

    const currentUser = await response.json();
    if (!currentUser) {
      throw new Error("Current user data is not available");
    }

    return currentUser;
  } catch (error) {
    console.error("Error fetching current user:", error.message);
    return null; // Return null to indicate an error occurred
  }
}
```

Hình 42. API lấy thông tin người dùng hiện tại

- Refresh Token:

- Refresh token cho phép người dùng duy trì phiên làm việc mà không cần phải nhập lại thông tin đăng nhập mỗi khi access token hết hạn. Điều này giúp cải thiện trải nghiệm người dùng.
- Bởi vì Access token thường có thời gian tồn tại ngắn trong khi refresh token có thể có thời gian tồn tại dài hơn. Nên giúp giảm thiểu nguy cơ bị lợi dụng nếu access token bị đánh cắp.
- Nếu Access Token hết hạn thì sẽ kiểm tra Refresh Token, để xem có hay không và kiểm tra tính hợp lệ của Refresh Token. Nếu Refresh Token hợp lệ thì một Access Token mới sẽ được tạo lại cho client.

```

if(refreshToken == null){
    return res.status(401).json({message: 'Phiên đăng nhập hết hạn.'});
}
else{
try{
    const refeshTokendecode = jwt.verify(refreshToken,process.env.REFRESH_TOKEN);

    const newAccessToken = await RefreshToken(refeshTokendecode.userId)
    console.log("token moi trong cookie",newAccessToken)

    res.cookie['accessToken', newAccessToken, {
        httpOnly: true,
        secure: process.env.NODE_ENV === 'production',
        sameSite: 'strict',
        maxAge: 15 * 60 * 1000 // 15 phút
    }];
    console.log("Đã refresh token")
    req.user = jwt.verify(newAccessToken, JWT_SECRET);
    next();
} catch (refreshError){
    return res.status(401).json({message: 'Phiên đăng nhập không hợp lệ, hãy đăng nhập lại'})
}
}

```

Hình 43. RefreshToken

- Luồng dữ liệu kết hợp Access Token và Refresh Token:
 - Bước 1: Người dùng đăng nhập thành công và nhận được Cookie có chứa các token
 - Bước 2: Người dùng gửi các yêu cầu sau khi đăng nhập phải đính kèm Cookie để Server xác minh, nếu thành công thì xử lý yêu cầu, thất bại thì bỏ qua yêu cầu.

- Bước 3: Khi Access Token hết hạn, Refresh Token còn hạn thì sẽ tạo lại Access Token cho người dùng.

```
const user = await User.findOne({ where: { email } });
const accessToken = jwt.sign(
  { userId: user.id, type: user.type },
  process.env.ACCESS_TOKEN,
  { expiresIn: "40m" }
);
const refreshToken = jwt.sign(
  { userId: user.id, email: user.email },
  process.env.REFRESH_TOKEN,
  { expiresIn: "7d" }
);
res.cookie("accessToken", accessToken, {
  httpOnly: true,
  secure: process.env.NODE_ENV === "production",
  sameSite: "strict",
  maxAge: 40 * 60 * 1000, // 40 phút
});
res.cookie("refreshToken", refreshToken, {
  httpOnly: true,
  secure: process.env.NODE_ENV === "production",
  sameSite: "strict",
  maxAge: 1440 * 60 * 1000,
});
```

Hình 44. Tạo và lưu trữ Token

```

async function authenticateToken(req, res, next) {
  const token = req.cookies.accessToken; // Ensure this line correctly reads the cookie
  const refreshToken = req.cookies.refreshToken;
  if (token) {
    jwt.verify(token, JWT_SECRET, (err, user) => {
      if (err) return res.status(403).json({ message: "Token không hợp lệ." });
      req.user = user;
      next();
    });
  } else {
    if (refreshToken == null) {
      return res.status(401).json({ message: "Phiên đăng nhập hết hạn." });
    } else {
      try {
        const refeshTokendecode = jwt.verify(
          refreshToken,
          process.env.REFRESH_TOKEN
        );
        const newAccessToken = await RefreshToken(refeshTokendecode.userId);
        res.cookie("accessToken", newAccessToken, [
          httpOnly: true,
          secure: process.env.NODE_ENV === "production",
          sameSite: "strict",
          maxAge: 40 * 60 * 1000,
        ]);
        req.user = jwt.verify(newAccessToken, JWT_SECRET);
        next();
      } catch (refreshError) {
        return res
          .status(401)
          .json({ message: "Phiên đăng nhập không hợp lệ, hãy đăng nhập lại" });
      }
    }
  }
}

```

Hình 45. Middleware kiểm tra Token

```

ReviewRouter.post(
  "/create",
  parseForm,
  csrfProtection,
  authenticateToken,
  uploadCloud.single("file"),
  createReview
);

```

Hình 46. Thêm các middleware vào trước các Routes

```

const response = await fetch("/api/v1/reviews/create", {
  method: "POST",
  credentials: "include",
  headers: {
    "CSRF-Token": token, // <-- is the csrf token as a header
  },
  body: formData,
});

```

Hình 47. Thêm CSRF Token vào Header

p) CSRF Token

- Là một token duy nhất được tạo ra và gắn với mỗi phiên người dùng. Nó được gửi kèm theo cùng với các Requests từ phía client đến server.
- Server kiểm tra token này để đảm bảo rằng yêu cầu đến từ nguồn hợp lệ (ứng dụng của chính nó) chứ không phải từ một trang web độc hại.
- Cấu hình CSRF Token:
 - **httpOnly**: Không thể lấy được Cookie bằng JavaScript, giúp đề phòng XSS
 - **secure**: Cookie chỉ được truyền qua kết nối HTTPS, tăng cường bảo mật.
 - **sameSite: 'strict'**: Cookie chỉ được gửi nếu yêu cầu bắt nguồn từ cùng một trang web, giảm nguy cơ CSRF.

```

var csrf = require('csurf');
// CSRF protection middleware with secure cookie configuration
var csrfProtection = csrf({
  cookie: {
    httpOnly: true, // Prevent access via JavaScript
    secure: true,   // Ensure cookies are sent only over HTTPS
    sameSite: 'strict', // Mitigate CSRF attacks
  },
});

```

Hình 48. Middleware kiểm tra CSRF

- Luồng dữ liệu của CSRF Token:

- Bước 1: Server tạo một CSRF token khi truy cập vào trang web.
- Bước 2: CSRF Token được lưu ở metadata của trang web, được sử dụng khi người dùng gửi Request tới server.
- Bước 3: Server kiểm tra CSRF Token trong yêu cầu và Access Token trong Cookie. Nếu phù hợp yêu cầu sẽ được xử lý, nếu không server sẽ từ chối yêu cầu.

```
Tabnine | Edit | Test | Explain | Document | Ask
app.get("/", csrfProtection, (req, res) => {
|   res.render("User/mainpage", { csrfToken: req.csrfToken() });
});

Tabnine | Edit | Test | Explain | Document | Ask
app.get("/chatbotimage", csrfProtection, (req, res) => {
|   res.render("user/chatbotImage", { csrfToken: req.csrfToken() });
});

Tabnine | Edit | Test | Explain | Document | Ask
app.get("/chatbot", csrfProtection, (req, res) => {
|   res.render("User/chatbot", { csrfToken: req.csrfToken() });
});

Tabnine | Edit | Test | Explain | Document | Ask
app.get("/hotelList", csrfProtection, (req, res) => {
|   res.render("User/hotelList", { csrfToken: req.csrfToken() });
});

Tabnine | Edit | Test | Explain | Document | Ask
app.get("/supplier", csrfProtection, (req, res) => {
|   res.render("User/supplier", { csrfToken: req.csrfToken() });
});

Tabnine | Edit | Test | Explain | Document | Ask
app.get("/register", blockLogin, csrfProtection, (req, res) => {
|   // Render trang đăng ký với CSRF token
|   res.render("User/register", { csrfToken: req.csrfToken() });
});
```

Hình 49. Thêm Middleware vào các Router

```

<html lang="en">

  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    | <meta name="csrf-token" content="{{csrfToken}}>
    <title>Register</title>
    <link
      | href="https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css"
      | rel="stylesheet"
    />
    <link rel="stylesheet" href="/css/register.css" />
    MADARAKZ, 2 months ago • authen(update) : update authentication
  </head>

```

Hình 50. Thêm Token vào front end

```

// Lấy CSRF token
const token = document.querySelector('meta[name="csrf-token"]').getAttribute('content');

// Gửi request đăng ký
$.ajax({
  url: "/api/v1/users/register",
  type: "POST",
  credentials: 'same-origin',
  headers: {
    'CSRF-TOKEN': token,
    'Content-Type': 'application/json'
  },

```

Hình 51. Gửi request kèm Token

5. Bảo mật cơ sở dữ liệu

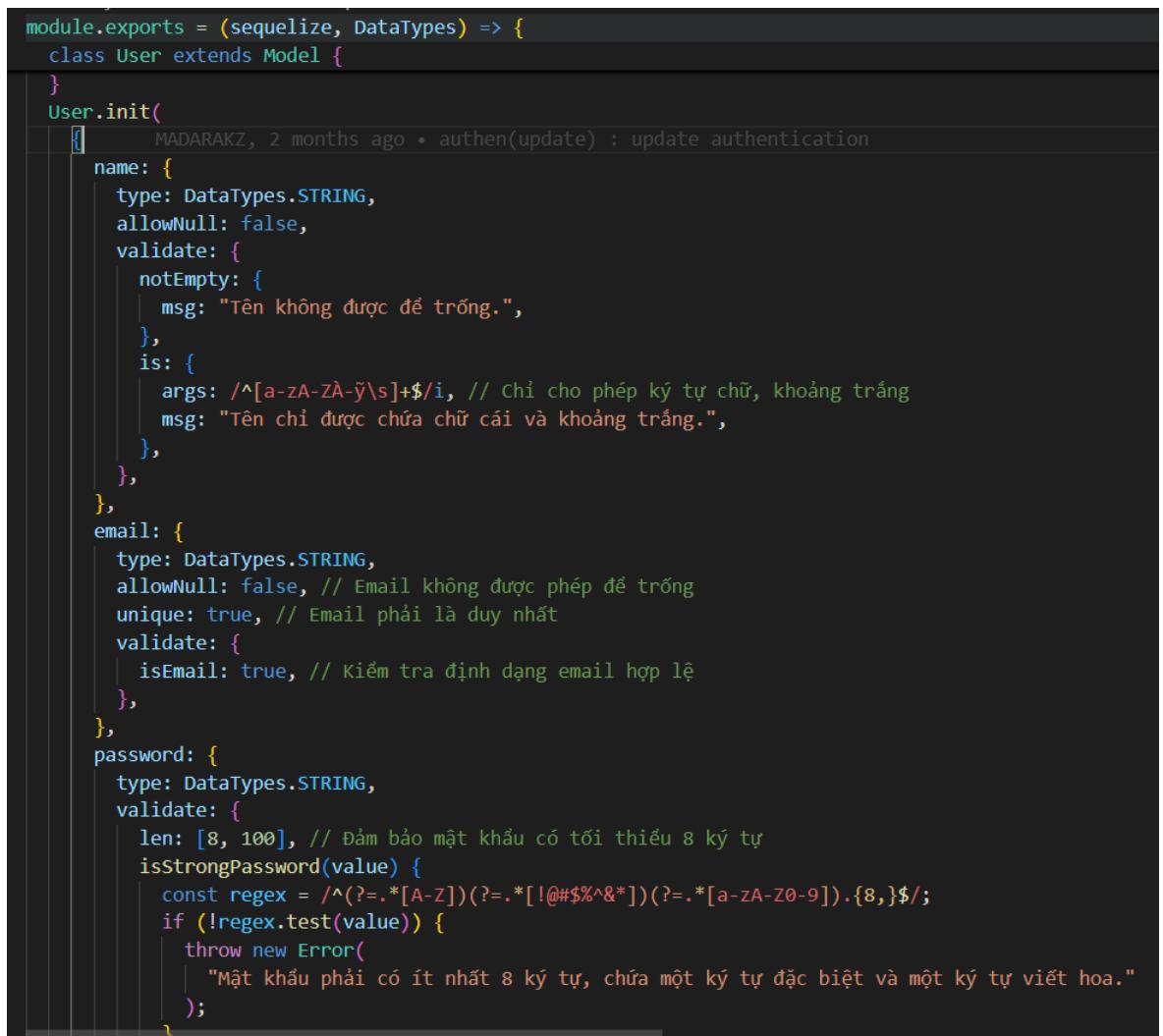
a) ORM (*Object-Relational Mapping*), Sequelize

- Là một kỹ thuật trong lập trình giúp lập bản đồ giữa các **đối tượng trong mã nguồn** (Objects) và các **bảng trong cơ sở dữ liệu quan hệ** (Relational Tables). Nó cung cấp một cách để thao tác với cơ sở dữ liệu mà không cần phải viết câu lệnh SQL trực tiếp.
- Sequelize là ORM phổ biến cho Nodejs hỗ trợ trên MySql. Có tác dụng tạo các quan hệ giữa các thành phần model dữ liệu. Dễ dàng tạo migration, hỗ trợ transaction và validate dữ liệu.

- Việc áp dụng Sequelize sẽ hạn chế được lỗ hổng Sql Injection bởi thuộc tính truy vấn của nó. Sequelize tham số hóa truy vấn đồng thời tách biệt dữ liệu và cấu trúc truy vấn.

```
const existingUser = await User.findOne({ where: { email: value } });
```

- Sequelize còn hỗ trợ xác thực đầu vào và tạo constrain trực tiếp trên model để giảm thiểu lỗi và tránh lưu dữ liệu không hợp lệ.



```
module.exports = (sequelize, DataTypes) => {
  class User extends Model {
  }
  User.init(
    {
      MADARAKZ, 2 months ago • authen(update) : update authentication
      name: {
        type: DataTypes.STRING,
        allowNull: false,
        validate: {
          notEmpty: {
            msg: "Tên không được để trống.",
          },
          is: {
            args: /^[a-zA-ZÀ-Ӄ\s]+$/i, // Chỉ cho phép ký tự chữ, khoảng trắng
            msg: "Tên chỉ được chứa chữ cái và khoảng trắng.",
          },
        },
      },
      email: {
        type: DataTypes.STRING,
        allowNull: false, // Email không được phép để trống
        unique: true, // Email phải là duy nhất
        validate: {
          isEmail: true, // Kiểm tra định dạng email hợp lệ
        },
      },
      password: {
        type: DataTypes.STRING,
        validate: {
          len: [8, 100], // Đảm bảo mật khẩu có tối thiểu 8 ký tự
          isStrongPassword(value) {
            const regex = /^(?=.*[A-Z])(?=.*[!@#$%^&*])(?=.*[a-zA-Z0-9]).{8,}$/;
            if (!regex.test(value)) {
              throw new Error(
                "Mật khẩu phải có ít nhất 8 ký tự, chứa một ký tự đặc biệt và một ký tự viết hoa."
              );
            }
          }
        }
      }
    }
  )
}
```

Hình 52. Sequelize

b) Mã hóa mật khẩu

- Trước khi được lưu vào cơ sở dữ liệu, mật khẩu sẽ được mã hóa với bộ mã khóa bí mật

```

// Hash password
const salt = await bcrypt.genSalt(10);
const hashPassword = await bcrypt.hash(password, salt);

// Create user
const newUser = await User.create({
  name,
  email,
  password: hashPassword,
  numberPhone,
  type,
});

```

Hình 53. Băm mật khẩu

c) Passport

- Áp dụng thư viện passport để xác thực và phân quyền khi sử dụng OAuth2 từ google. Passport còn có tác dụng mã hóa thông tin người dùng và lưu trữ id xác thực của Google API.

```

passport.js > ⌂ <function>
1 const express = require("express");
2 const passport = require("passport");
3 require("dotenv").config();
4 const GoogleStrategy = require("passport-google-oauth20").Strategy;
5 const { User } = require("./models");
6
7 // Simplified and consolidated passport configuration
8 passport.use(
9   new GoogleStrategy(
0     {
1       clientID: process.env.GOOGLE_CLIENT_ID,
2       clientSecret: process.env.GOOGLE_CLIENT_SECRET,
3       callbackURL: process.env.GOOGLE_CALLBACK_URL,
4       passReqToCallback: true
5     },
6     async (req, accessToken, refreshToken, profile, done) => {

```

Hình 54. Passport

d) Kiểm tra và làm sạch đầu vào

- Viết hàm middleware để lọc các dữ liệu đầu vào kết hợp sử dụng thư viện express-validator để kiểm tra tính hợp lệ của dữ liệu
- Middleware chứa các hàm được dùng để loại bỏ những thẻ HTML, mã hóa một số ký tự HTML để chặn XSS, thoát ký tự SQL để ngăn SQL Injection, thoát ký tự Regex đặc biệt để tránh lỗi khi sử dụng trong biểu thức chính quy.

- express-validator giúp kiểm tra dữ liệu đầu vào, đảm bảo các trường như email, mật khẩu, số điện thoại có định dạng đúng, đáp ứng các yêu cầu logic và bảo mật.

```
const removeHtmlTags = (input) => {
  if (typeof input !== "string") {
    return input; // Không xử lý nếu không phải chuỗi
  }
  return input.replace(/<\/?[^>]+(>|$)/g, ""); // Xóa thẻ HTML
};

// Thoát ký tự HTML để ngăn chặn XSS
Tabnine | Edit | Explain | Qodo Gen: Options | Test this function
const encodeHtmlEntities = (str) => {
  if (typeof str !== "string") {
    return str; // Không xử lý nếu không phải chuỗi
  }
  return str
    .replace(/&/g, "&amp;")
    .replace(/</g, "&lt;")
    .replace(/>/g, "&gt;")
    .replace(/\"/g, "&quot;")
    .replace(/\'/g, "&#039;");
};

// Thoát ký tự SQL để ngăn chặn SQL Injection
Tabnine | Edit | Explain | Qodo Gen: Options | Test this function
const escapeSqlInjection = (str) => {
  if (typeof str !== "string") {
    return str; // Không xử lý nếu không phải chuỗi
  }
  return sqlstring.escape(str).slice(1, -1); // Xóa dấu nháy đơn bao quanh
};

// Thoát ký tự đặc biệt trong Regex
Tabnine | Edit | Explain | Qodo Gen: Options | Test this function
const escapeRegex = (str) => {
  if (typeof str !== "string") {
    return str; // Không xử lý nếu không phải chuỗi
  }
  return str.replace(/\*+\^$\{()\|[\]\\\]/g, "\\\$&");
};
```

Hình 55. Validator

```

const sanitizeString = (str) => {
  if (typeof str !== "string") {
    return str; // Không xử lý nếu không phải chuỗi
  }
  let sanitized = removeHtmlTags(str); // Loại bỏ thẻ HTML
  sanitized = encodeHtmlEntities(sanitized); // Thoát ký tự HTML
  sanitized = escapeSqlInjection(sanitized); // Thoát ký tự SQL
  sanitized = escapeRegex(sanitized); // Thoát ký tự Regex
  return sanitized;
};

// Làm sạch và kiểm tra email
Tabnine | Edit | Explain | Qodo Gen: Options | Test this function
const validateAndSanitizeEmail = (email) => {
  if (typeof email !== "string") {
    throw new Error("Invalid input: Email must be a string.");
  }
  // Loại bỏ HTML tags
  email = removeHtmlTags(email);
  // Kiểm tra định dạng email hợp lệ
  const emailRegex = /^[^@\s]+@[^\s@]+\.\[^@\s]+$/;
  if (!emailRegex.test(email)) {
    throw new Error("Invalid email format.");
  }
  return email;
};

// Làm sạch các trường trong đối tượng
Tabnine | Edit | Explain | Qodo Gen: Options | Test this function
const sanitizeObject = (obj, fieldsToSanitize) => {
  if (typeof obj !== "object" || obj === null) {
    throw new Error("Input must be an object.");
  }
  You, 24 hours ago • note
  fieldsToSanitize.forEach((field) => {
    if (typeof obj[field] === "string") {
      obj[field] = sanitizeString(obj[field]);
    }
  });
};

```

Hình 56. Sanitize

```
// Middleware làm sạch dữ liệu đầu vào cho login
Tabnine | Edit | Explain | Qodo Gen: Options | Test this function
const sanitizeLoginInputs = (req, res, next) => {
  try {
    if (req.body.email) {
      req.body.email = validateAndSanitizeEmail(req.body.email);
    }
    next();
  } catch (error) {
    res.status(400).send({ error: error.message });
  }
};

// Export các hàm
module.exports = {
  removeHtmlTags,
  encodeHtmlEntities,
  escapeSqlInjection,
  escapeRegex,
  sanitizeString,
  validateAndSanitizeEmail,
  sanitizeObject,
  sanitizeLoginInputs,
};
```

Hình 57. Code sanitizeLoginInputs

```
const { validationResult } = require("express-validator"); // Import validationResult
const { body } = require("express-validator");
const { sanitizeObject } = require("../middlewares/validations/sanitize");
```

Hình 58. Khai báo biến sử dụng validator

```

const register = [
    // Làm sạch dữ liệu đầu vào
    (req, res, next) => {
        sanitizeObject(req.body, ["name", "numberPhone", "type"]);
        next();
    },
    // Validate các trường
    body("name").trim().notEmpty().withMessage("Vui lòng nhập tên"),

    body("email")
        .trim()
        .isEmail()
        .withMessage("Email không hợp lệ")
        .custom(async (value) => {
            const existingUser = await User.findOne({ where: { email: value } });
            if (existingUser) {
                throw new Error("Email đã tồn tại");
            }
        }),
    body("password")
        .isLength({ min: 8 })
        .withMessage("Mật khẩu phải có ít nhất 8 ký tự")
        .matches(/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[$!%*?&])[A-Za-z\d@$!%*?&]/)
        .withMessage(
            "Mật khẩu phải chứa ít nhất 1 chữ hoa, 1 chữ thường, 1 số và 1 ký tự đặc biệt"
        ),
    body("confirmPassword").custom((value, { req }) => {
        if (value !== req.body.password) {
            throw new Error("Mật khẩu và xác nhận mật khẩu không khớp");
        }
        return true;
}),
]

```

Hình 59. Dùng sanitize cho các trường dữ liệu

6. Úng dụng các thư viện hỗ trợ tích hợp an toàn.

e) Express-rate-limit

- Thư viện express-rate-limit hỗ trợ cơ chế để nhóm em xây dựng một middleware có chức năng hạn chế số requests của một IP gửi lên server. Việc cấu hình này sẽ giới hạn số lượng request trong một phiên từ một địa chỉ IP, nhằm cân bằng tải cho server cũng như hạn chế được các cuộc tấn công brutce force và DOS.

```
const ratelimit = require("express-rate-limit");
```

Hình 60. Thư viện rate-limit

```
    },
    const limiter = ratelimit({
      windowMs: 15 * 60 * 1000,
      max: 1000,
      message: "Too many API request from this IP",
    });
  
```

Hình 61. Code thực hiện rate-limit

```
✓ app.get("/signin", blockLogin, limiter, csrfProtection, (req, res) => {
  |   res.render("User/signin", { csrfToken: req.csrfToken() });
  | });
  
```

Hình 62. Import vào trong các phương thức

f) CORS

- Cấu hình Cors (Cross-Origin Resource Sharing) cho ứng dụng. Chỉ cho phép yêu cầu từ domain của hệ thống, ngăn chặn các domain khác truy cập API. Điều này đảm bảo an toàn và kiểm soát truy cập cho ứng dụng web, chặn các yêu cầu từ các nguồn không được phép.
- Thiết lập Credential cho phép gửi nhận cookies giữa front end và back end.

```
const cors = require("cors");
  
```

Hình 63. Code sử dụng thư viện cors

```
1 ✓ app.use(
2 ✓   cors({
3   |     origin: "http://localhost:3030", // Domain của frontend
4   |     credentials: true, // Đảm bảo gửi và nhận cookies
5   |   })
6   );
  
```

Hình 64. Code thực hiện cors

g) JWT (Json web token), bcryptjs và Crypto

- Sử dụng thư viện jwt để thực hiện mã hóa các tham số thông tin người dùng và lưu trữ an toàn trên cookies, mã hóa được thực hiện với 1 secret key được lưu trữ an toàn, thiết lập thời gian có hiệu lực của token.

```
const accessToken = jwt.sign(
  { userId: user.id, type: user.type },
  process.env.ACCESS_TOKEN,
  { expiresIn: "40m" }
);
```

Hình 65. Tạo accessToken có thời hạn trong 40 phút

- Bcrypt.js là một thư viện mã hóa mật khẩu được sử dụng rộng rãi trong Node.js để bảo vệ mật khẩu người dùng. Tiến hành mã hóa với một chuỗi kí tự ngẫu nhiên. Hỗ trợ các hàm genSaltSync(), hashSync(), compareSync() hỗ trợ tạo khóa, băm và so sánh mật khẩu được người dùng cung cấp với mật khẩu xác thực.

```
// Hash password
const salt = await bcrypt.genSalt(10);
const hashPassword = await bcrypt.hash(password, salt);

...
// B4: Kiểm tra mật khẩu
const isAuthen = await bcrypt.compare(password, user.password);
```

Hình 66. Băm mật khẩu

- Crypto là một module trong Node.js cung cấp các chức năng mã hóa bằng nhiều thuật toán. Chức năng trong đoạn code để tạo và xác minh chữ kí, ứng dụng HMAC bằng thuật toán SHA512 để tạo chữ kí bảo mật cho VNpay.

```
const hmac = crypto.createHmac("sha512", config.get("vnpay.hashSecret"));
const signed = hmac.update(new Buffer(signData, "utf-8")).digest("hex");
```

Hình 67. Dùng thư viện crypto

```
const crypto = require("crypto");
const moment = require("moment");
```

Hình 68. Import thư viện crypto

h) Crusf

- Là một thư viện trong express.js hỗ trợ các chức năng: tự động sinh ra 1 token duy nhất cho mỗi request, token này được sử dụng đến xác minh tính hợp lệ của request

```
grewares / autnen / > csrfprotection.js > ↗ <unknown>
1 var cookieParser = require('cookie-parser');
2 var csrf = require('csurf');
3 var bodyParser = require('body-parser');
4
5 // CSRF protection middleware with secure cookie configuration
6 var csrfProtection = csrf({
7   cookie: {
8     httpOnly: true, // Prevent access via JavaScript
9     secure: true, // Ensure cookies are sent only over HTTPS
10    sameSite: 'strict', // Mitigate CSRF attacks
11  },
12});
```

Hình 69. Sử dụng thư viện Crusf

i) dotenv

- dotenv là một thư viện quản lý môi trường trong Node.js. Nhằm lưu trữ các thông tin nhạy cảm như secret key cho các thành phần sử dụng mã hóa, password của các thành phần sử dụng các bên thứ ba như Google API, Google Captcha, Cloudinary... Thông tin đăng nhập và truy suất của server Database.

```
require("dotenv").config();
```

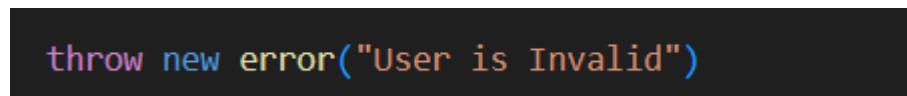
Hình 70. Sử dụng dotenv

```

1 PORT=3030
2 NODE_ENV=development
3 CLOUDINARY_NAME = de1a04hrb
4 CLOUDINARY_SECRET = vy2miqxsf1i6ilog0xf7co6P_2k
5 EMAIL_SERVICE=Gmail
6 EMAIL_USERNAME=225214A4@gnv.uvt.edu.vn
7 EMAIL_PASSWORD=fhdg itam dsjv hnpy
8 JWT_SECRET = firewallbase64
9
10 ACCESS_TOKEN=token_10_10_diem
11 ACCESS_TOKEN_EXPIRED=2h
12 REFRESH_TOKEN=token_10_10_diem
13 REFRESH_TOKEN_EXPIRED=3d
14
15
16 DB_HOST=us-cluster-east-01.kbs.cleardb.net
17 DB_USER=b222956f7ce284
18 DB_PASSWORD=3fbcae05
19 DB_NAME=heroku_0f4e26bb0bb88096
20 API_URL=https://lastingtrip-674584f294be.herokuapp.com
21 GOOGLE_CLIENT_ID="7513497144-1ogdmim3a0kryp9sttspcouvb7peb8r.apps.googleusercontent.com"
22 GOOGLE_CLIENT_SECRET="60CSPX-h3ktv1beZETKTAhsJbARq4fA"
23 GOOGLE_CALLBACK_URL="http://localhost:3030/api/v1/users/auth/google/callback"
24 SECRET_KEY="Q3lHn2Q6502CmrSug/kXQ8t0HmykbSH94SughU/SsFu="
25
26
27 RECAPTCHA_SITE_KEY=6Lfv34wqAAAAJ3YMjekI0MPatFxdg6R2NDNyxD
28 RECAPTCHA_SECRET_KEY=6Lfv34wqAAAAVi8xN3sFL3Gw_3q2sy5fHMD-NRS
29
30 EMAIL_USER=gklatthumong@gmail.com
31 EMAIL_PASS=ohep rnob rhss grny

```

Hình 71. File .env



Hình 72. Thông báo lỗi

j) *Multer-storage-cloudinary*

- Là một package npm hỗ trợ việc lưu trữ các file ảnh được upload lên Cloudinary thông qua middleware multer. Cụ thể nó giúp tích hợp dịch vụ lưu trữ và xử lý ảnh của Cloudinary với quy trình upload file của ứng dụng Node.js.
- File không lưu trữ trên server mà được tải trực tiếp lên Cloudinary giúp giảm nguy cơ mã độc ảnh hướng đến hệ thống server của bạn.

```

dlewares > upload > JS cloudinary.config.js > ...
1 const cloudinary = require('cloudinary').v2;
2 const { CloudinaryStorage } = require('multer-storage-cloudinary');
3 const multer = require('multer');
4 require('dotenv').config();
5 cloudinary.config({
6   cloud_name: process.env.CLOUDINARY_NAME,
7   api_key: process.env.CLOUDINARY_KEY,
8   api_secret: process.env.CLOUDINARY_SECRET
9 });

10 const storage = new CloudinaryStorage({
11   cloudinary,
12   allowedFormats: ['jpg', 'png'],
13   params :
14   {
15     folder : "LastingTrip"
16   }
17 });
18
19 const uploadCloud = multer({ storage });
20
21 module.exports = uploadCloud;
22

```

Hình 73. Sử dụng package multer-storage-cloudinary

k) sqlString

- sqlString là một thư viện Node.js được sử dụng để escape(thoát) và format các câu truy vấn SQL một cách an toàn. Nhằm hạn chế SqlInjection khi làm việc với dữ liệu động và không được kiểm soát hoàn toàn.

```

const sqlstring = ...require("sqlstring");

```

Hình 74. Sử dụng thư viện sqlstring

```

}
return sqlstring.escape(str).slice(1, -1); // Xóa dấu nháy đơn bao quanh

```

Hình 75. Code thực hiện

7. Thiết kế helmet CSP (Content Security Policy)

- Helmet content security policy được xem như là một middleware trong Express.js. Có chức năng thiết lập các chính sách riêng cho nguồn tài nguyên của ứng dụng. Chúng em đã xây dựng các bộ rule riêng biệt để ngăn chặn các script/resource độc hại từ các domain không tin cậy. Thiết lập các domain tin cậy từ các Third-party như Google, VNpay, Bootstrap.... và các CDN đã được kiểm duyệt.

- Việc xây dựng hệ thống chính sách này sẽ giúp ứng dụng ngăn chặn được các cuộc tấn công XSS, inline scripts, kiểm soát nguồn tài nguyên thoát, thậm chí là các cuộc tấn công Injection, CSRF. Bên cạnh đó bộ connectSrc cũng giúp kiểm soát các kết nối API, chặn các requests không mong muốn.

```

app.use(helmet());
  app.use(
    helmet.contentSecurityPolicy({
      directives: {
        defaultSrc: ["'self'",], // Allow resources only from the same origin (self)

        scriptSrc: [
          "'self'",
          "https://cdnjs.cloudflare.com",
          "https://cdn.jsdelivr.net",
          "https://apis.google.com",
          "https://ajax.googleapis.com",
          "https://code.jquery.com/",
          "https://sandbox.vnpayment.vn",
          "https://teachablemachine.withgoogle.com",
          "https://embed.pickaxeproject.com",
          "https://www.google.com",
          "https://www.gstatic.com/", // Allow scripts from Google APIs
        ],
        scriptSrcAttr: ["'self'", "https://www.bing.com"], // Allow inline event handlers
        stylesrc: [
          "'self'", // Allow styles from the same origin
          "https://cdnjs.cloudflare.com", // Font Awesome, Bootstrap from CDN
          "https://fonts.googleapis.com", // Google Fonts from CDN
          "https://fonts.gstatic.com", // Google Fonts static resources
          "https://cdn.jsdelivr.net", // Bootstrap styles
          "https://netdna.bootstrapcdn.com",
          "https://unpkg.com/", // Bootstrap fonts // Allow inline styles (use this cautiously; hashes or nonces are safer)
        ],
      }
    })
  )

```

Hình 76. Code thực hiện helmet

```

75   directives: {
76     styleSrc: [
77       "https://netdna.bootstrapcdn.com",
78       "https://unpkg.com", // Bootstrap fonts // Allow inline styles (use this cautiously; hashes or nonces are safer)
79     ],
80
81     imgSrc: [
82       "'self'", // Allow images from the same origin
83       "https://res.cloudinary.com",
84       "https://th.bing.com",
85       "https://www.bing.com",
86       "https://phongreviews.com",
87       "https://www.gstatic.com/",
88       "https://ak-d.tripcdn.com/",
89       "https://sandbox.vnpayment.vn", // Cloudinary for images
90       "data:", // Allow data URIs (used for inline images or icons)
91       "blob:",
92     ],
93
94     fontsSrc: [
95       "'self'", // Allow fonts from the same origin
96       "https://cdnjs.cloudflare.com", // Font Awesome
97       "https://fonts.googleapis.com", // Google Fonts stylesheets
98       "https://fonts.gstatic.com", // Google Fonts static resources
99       "https://netdna.bootstrapcdn.com",
100      "https://unpkg.com", // Bootstrap fonts
101    ],
102
103    connectSrc: [
104      "'self'", // Allow connections (e.g., API calls) from the same origin
105      "https://example.com",
106      "https://sandbox.vnpayment.vn",
107      "https://teachablemachine.withgoogle.com", // Replace with your specific API endpoint if needed
108    ],
109
110    objectSrc: ["'self'"], // Disallow all object, embed, or plugin-based resources for security
111  }

```

Hình 77. Code thực hiện helmet (tiếp theo)

C. Kết quả và demo

8. Giao diện tổng quan của ứng dụng

- Trang chủ

The screenshot shows the LastingTrip homepage. At the top, there's a search bar with fields for 'Check-in' (12/07/2024), 'Check-out' (12/09/2024), 'Số lượng phòng và khách' (1 Phòng, 1 Người lớn, 0 Trẻ em), and a 'Search' button. Below the search bar is a large banner image of a city skyline at night. Underneath the banner, there's a section titled 'Lựa chọn hàng đầu về khách sạn 5 sao sang trọng' (Top choices for 5-star luxury hotels) featuring four hotel cards:

- InterContinental Phu Quoc Long Beach Resort**: From VND 108,720. Includes a map and a photo of a lounge area.
- California Hotel**: From VND 3,221,368. Includes a photo of a poolside area.
- Premier Pearl Hotel Vung Tau**: From VND 191,723. Includes a photo of a room with two beds.
- Le Macaron Boutique Hotel**: From VND 121,076. Includes a photo of a room with a large bed.

Hình 78. Giao diện trang chủ

- Chi tiết thông tin về một khách sạn

The screenshot shows the detail page for the Cozum Homes Sonata Residence. At the top, it displays the hotel name, address (Đường Nguyễn Thị Thập, Phường Tân Phú, Quận 7, TP.HCM | Cách trung tâm thành phố 5.2 km), price (1,659,735 VND), and a 'Chọn phòng' (Select room) button. Below this is a summary section with a note about free Wi-Fi and a link to view more details. There are three large images of the hotel's exterior and interior.

Tiện nghi

- Wifi công cộng
- Bể bơi ngoài trời
- Khu vực tắm nắng
- Spa
- Bar
- Cafes

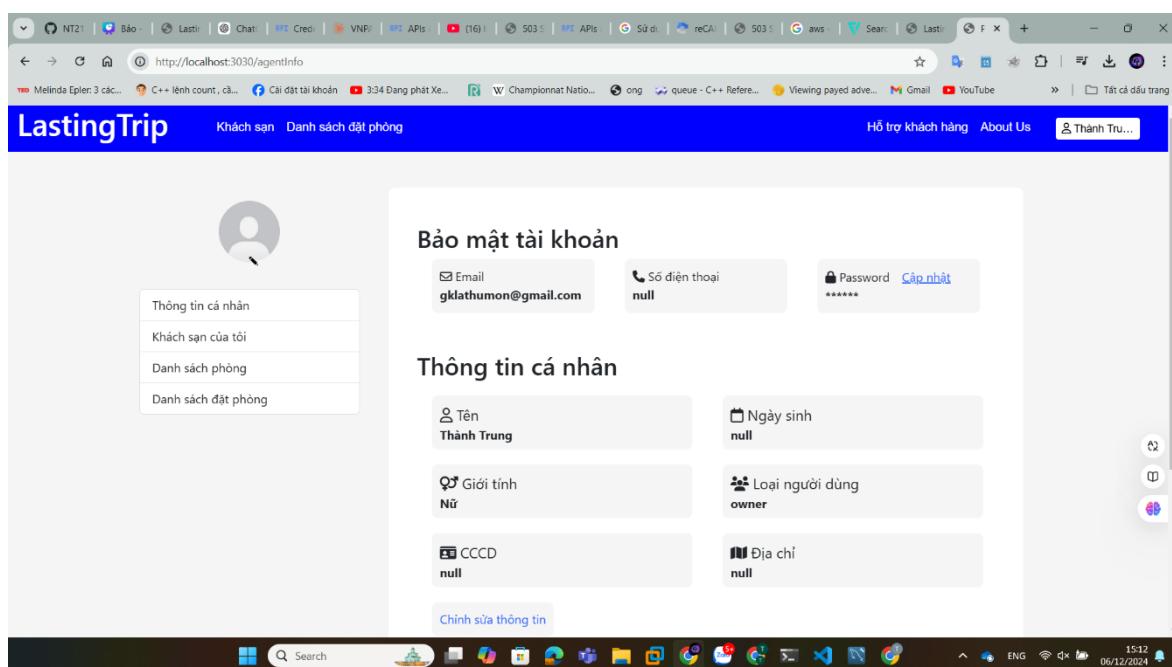
Chính sách

- Thời gian nhận/trả phòng
- Nội và giường phụ
- Thú cưng
- Chính sách trẻ em
- Bữa ăn sáng
- Xem thêm

At the bottom, there are tabs for 'Phòng' (Rooms), 'Đánh giá' (Reviews), 'Chính sách' (Policies), and 'Tiện nghi' (Facilities).

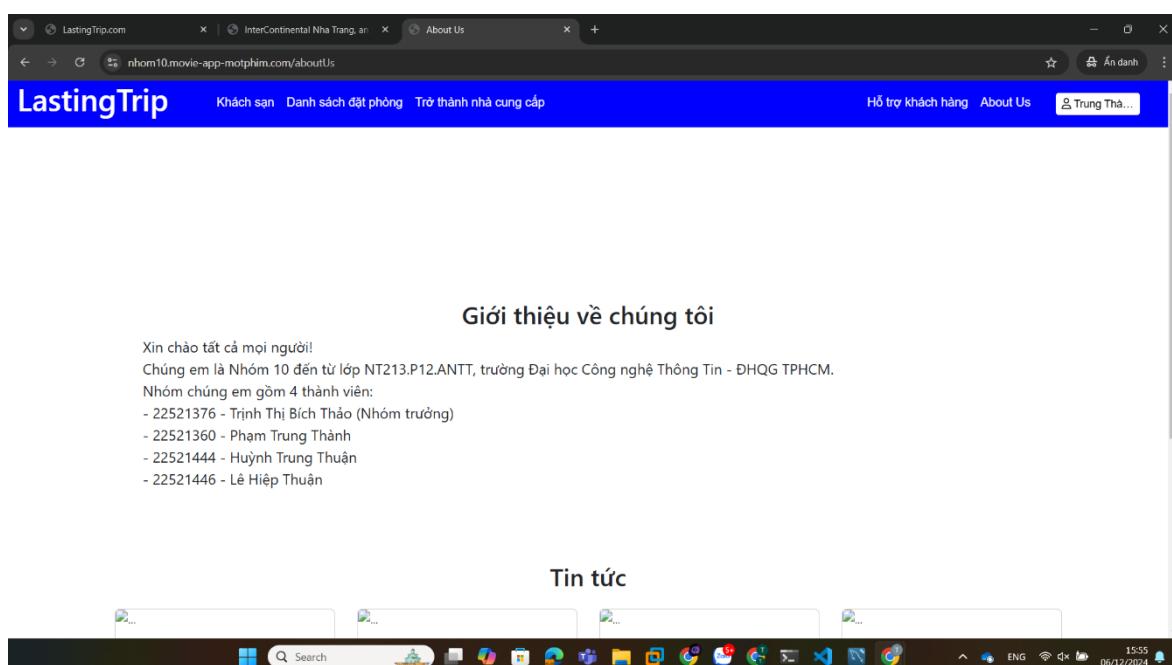
Hình 79. Giao diện khách sạn

- Thông tin người dùng

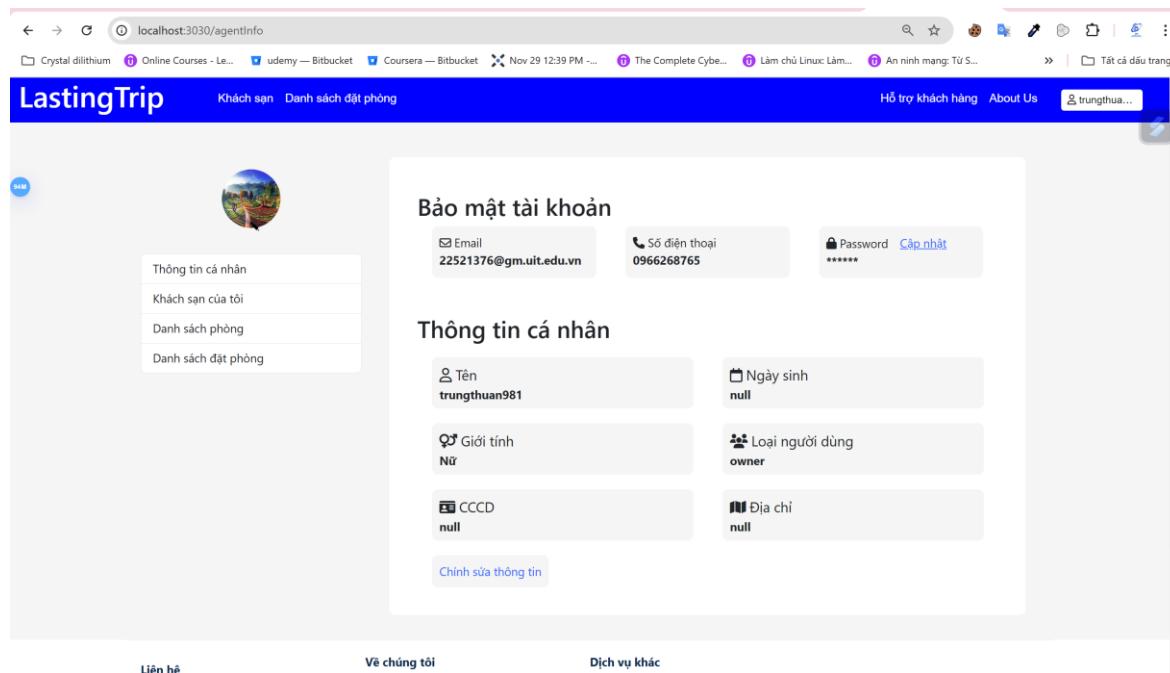


Hình 80. Giao diện trang user

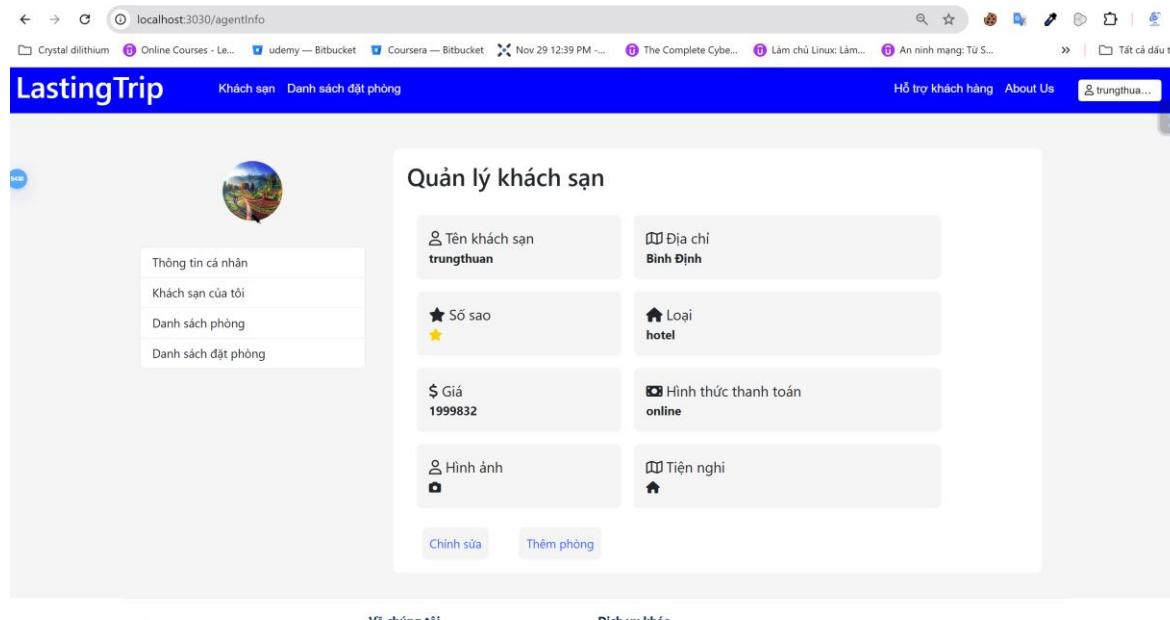
-About us



Hình 81. Trang about us



Hình 82. Giao diện trang owner



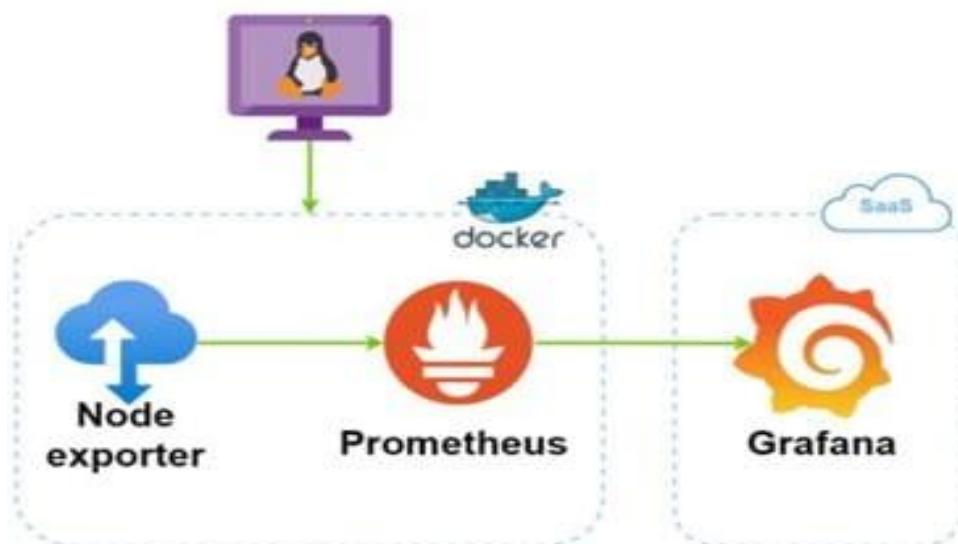
Hình 83. Trang quản lý khách sạn

Hình 84. Trang quản lý phòng

Hình 85. Trang giao diện của admin

Thiết lập hệ thống giám sát

- Xây dựng hệ thống giám sát với PNG Stack được cài đặt trên server AWS



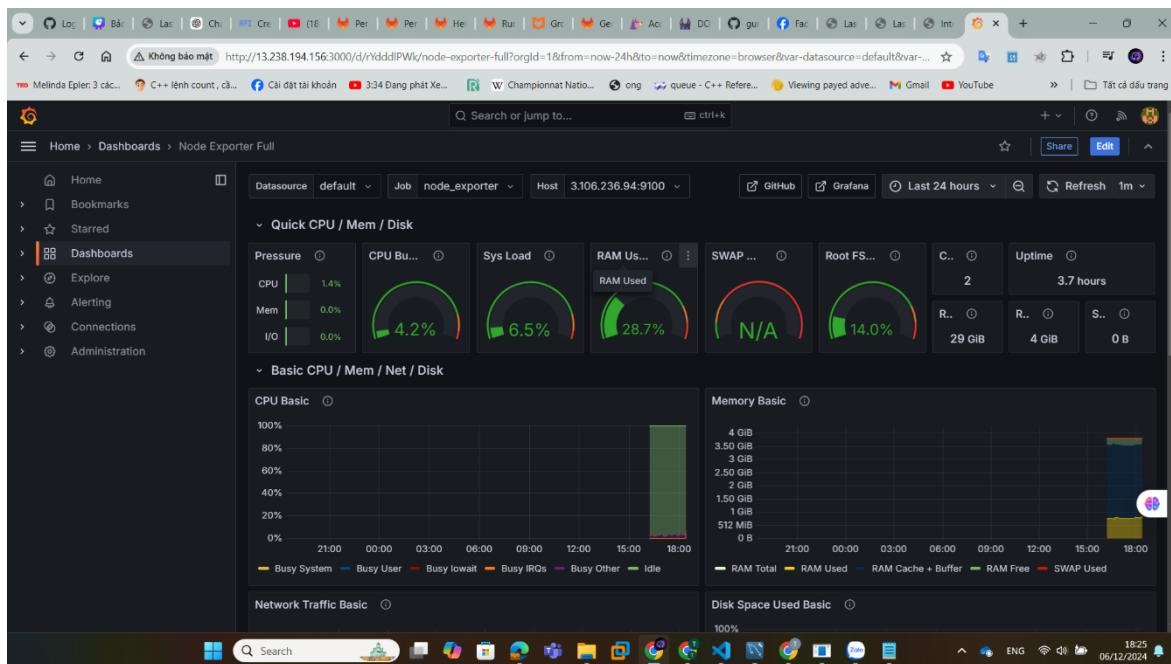
Hình 86. Hệ thống PNG Stack

Prometheus: Prometheus là một trong những công cụ giám sát phổ biến nhất hiện nay, được phát triển bởi Google và có một cộng đồng người dùng lớn. Nó được thiết kế để thu thập và lưu trữ dữ liệu giám sát từ các hệ thống và ứng dụng phân tán lớn. Prometheus sử dụng mô hình pull (kéo) để thu thập dữ liệu, cho phép nó linh hoạt và mở rộng tốt trên các hệ thống có quy mô khác nhau. Trong lĩnh vực giám sát có hai mô hình là pull và push, hiệu đơn giản liên quan đến luồng dữ liệu từ agent đến server và ngược lại.

Node Exporter: Đây là một trong những exporter phổ biến nhất trong cộng đồng Prometheus, chuyên thu thập các số liệu liên quan đến hệ điều hành Linux, bao gồm CPU, bộ nhớ, lưu trữ, và mạng. Còn rất nhiều Exporter khác, mình sẽ làm rõ hơn từng loại ở phần thực hành.

Grafana: Công cụ này hỗ trợ trực quan hóa dữ liệu mạnh mẽ, cho phép người dùng tạo các bảng dashboard tùy chỉnh để hiển thị dữ liệu một cách dễ dàng và hiệu quả. Grafana có thể phân tích được nhiều nguồn dữ liệu khác nhau không chỉ Prometheus.

- Ngoài server dùng để deploy ứng dụng, chúng em sẽ dùng thêm một server có trách nhiệm monitor. Hiển thị thông tin hệ thống của ứng dụng cũng như trạng thái của hệ thống.



Hình 87. Server giám sát hệ thống của Ứng dụng.

- Link server monitor: <http://13.238.194.156:3000>
- tài khoản admin/admin.

9. Deploy ứng dụng

- Nhóm chúng em Deploy ứng dụng trên Web service AWS, sử dụng nginx và mysql-server



Hình 88. AWS

Link deploy: <https://nhom10.movie-app-motphim.com>

10. Kết quả kiểm thử bảo mật

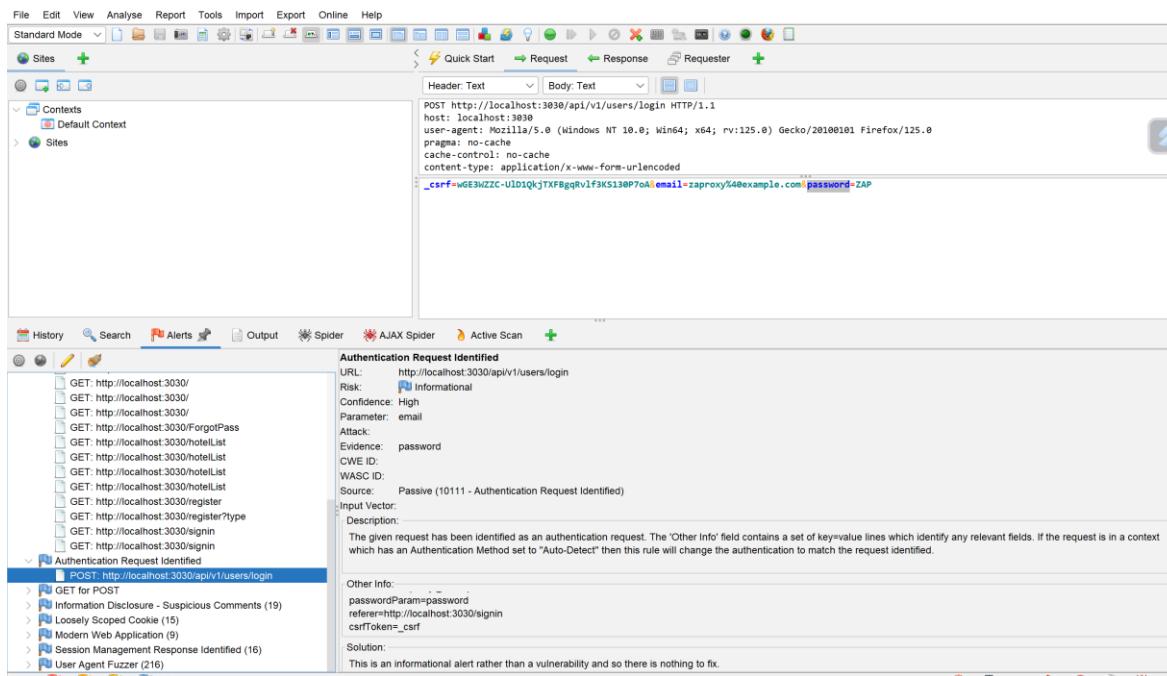
l) Zap

The screenshot shows the Zap proxy interface. At the top, a request and response pane displays a 200 OK status with various headers including Access-Control-Allow-Origin, Content-Security-Policy, and Vary. The body of the response contains a script tag with a source from ajax.googleapis.com. Below this, the 'Alerts' tab is selected, showing a single alert for 'Cross-Domain JavaScript Source File Inclusion'. The alert details show a GET request to /ForgotPass that includes a script from ajax.googleapis.com. It includes information like URL, Risk (Low), Confidence (Medium), Parameter (the script URL), Attack (script injection), Evidence (the script tag), CWE ID (829), WASC ID (15), and Source (Passive). A note states: 'The page includes one or more script files from a third-party domain.'

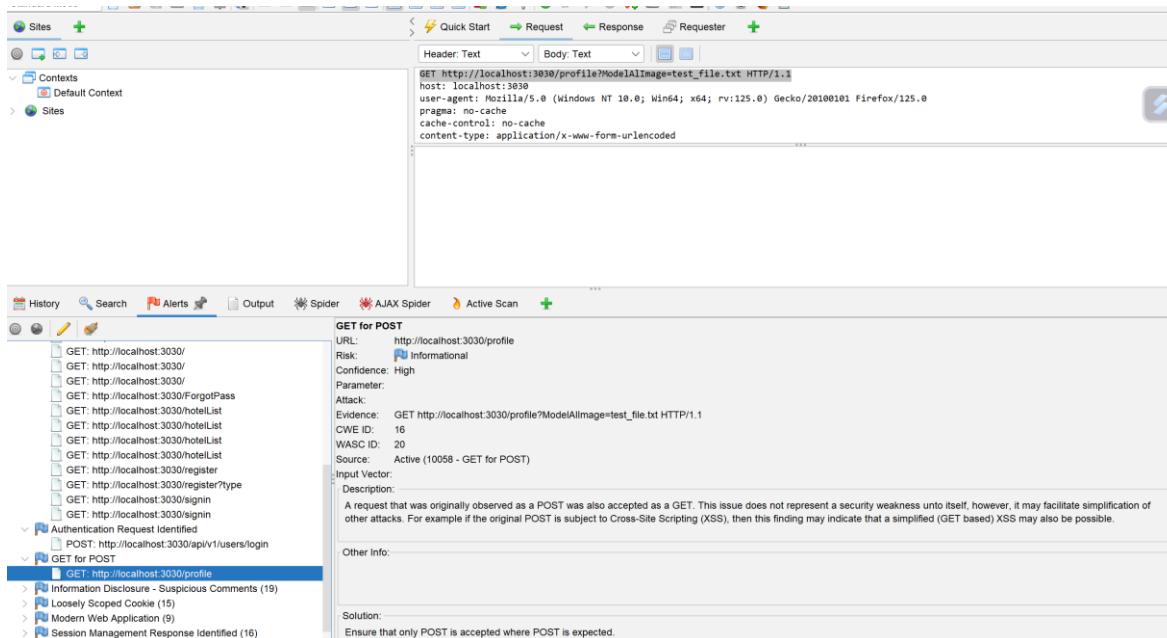
Hình 89. Ứng dụng quét lỗ hổng bảo mật Zap trang ForgotPass

This screenshot shows Zap's proxy interface with the 'Alerts' tab selected. It displays an alert for a 'CSP: Wildcard Directive' found at http://localhost:3030/api/v1/chatbotAI/findlocation. The alert details indicate a 404 Not Found status, a 'none' Content-Security-Policy, and a 'same-origin' Cross-Origin-Opener-Policy. The attack type is 'Content-Security-Policy', and the evidence shows a default-src 'none' directive. The alert includes CWE ID (693), WASC ID (15), and Source (Passive). A note explains that CSP is an added layer of security to detect XSS and data injection attacks. The 'Other Info' section notes that wildcard directives are often overly broad. The 'Solution' section provides a link to the CSP documentation.

Hình 90. Ứng dụng quét lỗ hổng bảo mật Zap cho api gọi chatbot



Hình 91. Zap



Hình 92. Kết quả quét Zap

The screenshot shows the Zap proxy interface. On the left, a sidebar lists various requests made to the host. In the main pane, a detailed alert is displayed:

- HTTP/1.1 200 OK**
- Vary: Origin**
- Access-Control-Allow-Origin: http://localhost:3030**
- Access-Control-Allow-Credentials: true**
- Content-Security-Policy: default-src 'self' script-src 'self' https://cdnjs.cloudflare.com https://cdn.jsdelivr.net https://apis.google.com https://ajax.googleapis.com https://code.jquery.com https://sandbox.vnpayment.vn https://teachablemachine.withgoogle.com https://embed.pickaxe-project.com https://www.google.com https://www.gstatic.com;script-src-attr 'self' https://www.bing.com/style-src 'self' https://cdnjs.cloudflare.com https://fonts.googleapis.com https://fonts.gstatic.com https://cdn.jsdelivr.net https://netdna.bootstrapcdncdn.com https://unkg.com/img-src 'self' https://res.cloudinary.com https://th.bing.com https://www.bing.com https://phongreviews.com https://www.gstatic.com https://ak-d.tripcdn.com/ https://sandbox.vnpayment.vn data: blob;font-src 'self'**
- //Get the button**
- let mybutton = document.getElementById("btn-back-to-top");**
- // When the user scrolls down 20px from the top of the document, show the button**
- window.onscroll = function () { scrollFunction();};**

Other Info:
The following pattern was used: !bUSER:b and was detected 2 times, the first in the element starting with: // When the user scrolls down 20px from the top of the document, show the button", see evidence field for the suspicious comment/snippet.

Solution:
Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.

Reference:

Alert Tags:

Key	Value
OWASP_2021_A01	https://owasp.org/Top10/A01_2021-Broken_Access_Control/
WSTG-v42-INFO-05	https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security/Test_04_Informational_Controls/05_Informational_Controls
OWASP_2017_A03	https://owasp.org/www-project-top10en/2017/A3_2017-Sensitive_Data_Exposure.html
CWE-200	https://cwe.mitre.org/data/definitions/200.html

Current Scans: 0/1/0/0/0/0/0/0/0/0/0/0/0/0/0/0/0/0

Hình 93. Kết quả quét Zap

The screenshot shows the Zap proxy interface with a detailed view of a security alert for a loosely scoped cookie vulnerability:

- Header Text**: HTTP/1.1 200 OK
- Access-Control-Allow-Origin: http://localhost:3030**
- Vary: Origin**
- Access-Control-Allow-Credentials: true**
- Content-Security-Policy: default-src 'self';script-src 'self' https://cdnjs.cloudflare.com https://cdn.jsdelivr.net https://apis.google.com https://ajax.googleapis.com https://code.jquery.com https://sandbox.vnpayment.vn https://teachablemachine.withgoogle.com https://embed.pickaxe-project.com https://www.google.com https://www.gstatic.com;script-src-attr 'self' https://www.bing.com/style-src 'self' https://cdnjs.cloudflare.com https://fonts.googleapis.com https://fonts.gstatic.com https://cdn.jsdelivr.net https://netdna.bootstrapcdncdn.com https://unkg.com/img-src 'self' https://res.cloudinary.com https://th.bing.com https://www.bing.com https://phongreviews.com https://www.gstatic.com https://ak-d.tripcdn.com/ https://sandbox.vnpayment.vn data: blob;font-src 'self'**
- <!DOCTYPE html>**
- <html lang="en">**
- <head>**
- <meta charset="UTF-8">**
- <meta name="csrf-token" content="HAPxMyM7-X0Kfn8SbGub_LSNPhLZFMhslREQ">**
- <meta name="viewport" content="width=device-width, initial-scale=1.0">**

Loosely Scoped Cookie

URL: http://localhost:3030

Risk: Informational

Confidence: Low

Parameter:

Attack:

Evidence:

- CWE ID: 565
- WASC ID: 15
- Source: Passive (90033 - Loosely Scoped Cookie)

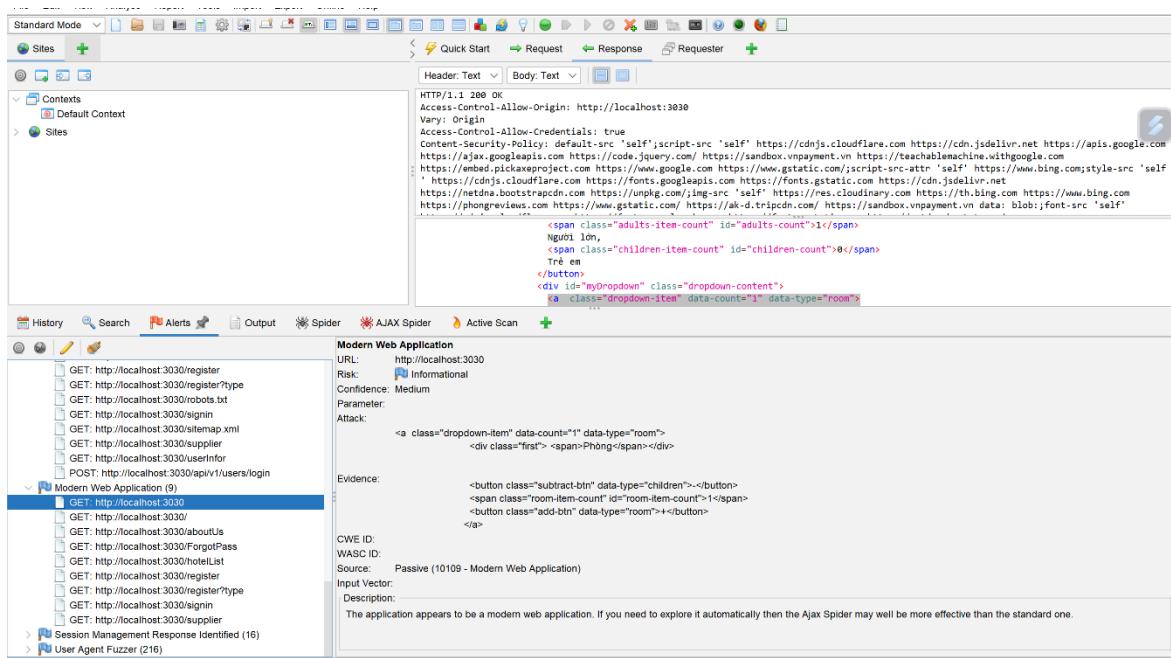
Input Vector:

Description: Cookies can be scoped by domain or path. This check is only concerned with domain scope. The domain scope applied to a cookie determines which domains can access it. For example, a cookie can be scoped strictly to a subdomain e.g. www.nottrusted.com, or loosely scoped to a parent domain e.g. nottrusted.com. In the latter case, any subdomain of nottrusted.com can access the cookie. Loosely scoped cookies are common in mega-applications like google.com and live.com. Cookies set from a subdomain like app.foo.bar are

Other Info: The origin domain used for comparison was: localhost _carf=JnnMAocgvFt0HZIDNbZft3

Solution: Always scope cookies to a FQDN (Fully Qualified Domain Name).

Hình 94. Zap

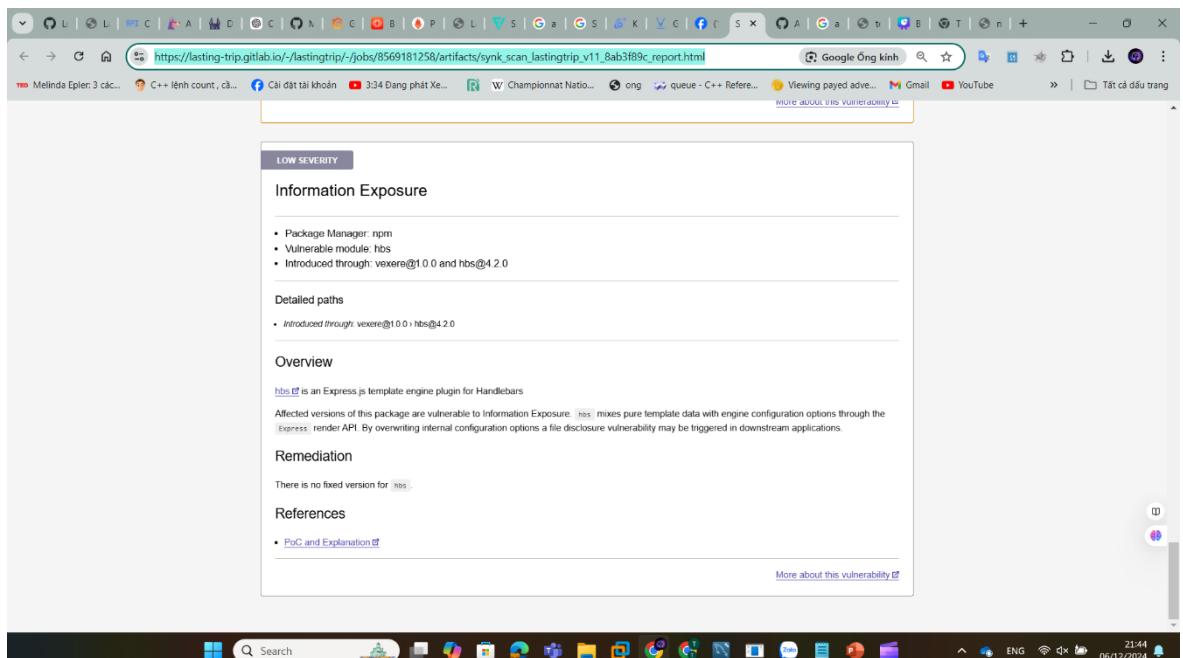


Hình 95. Zap

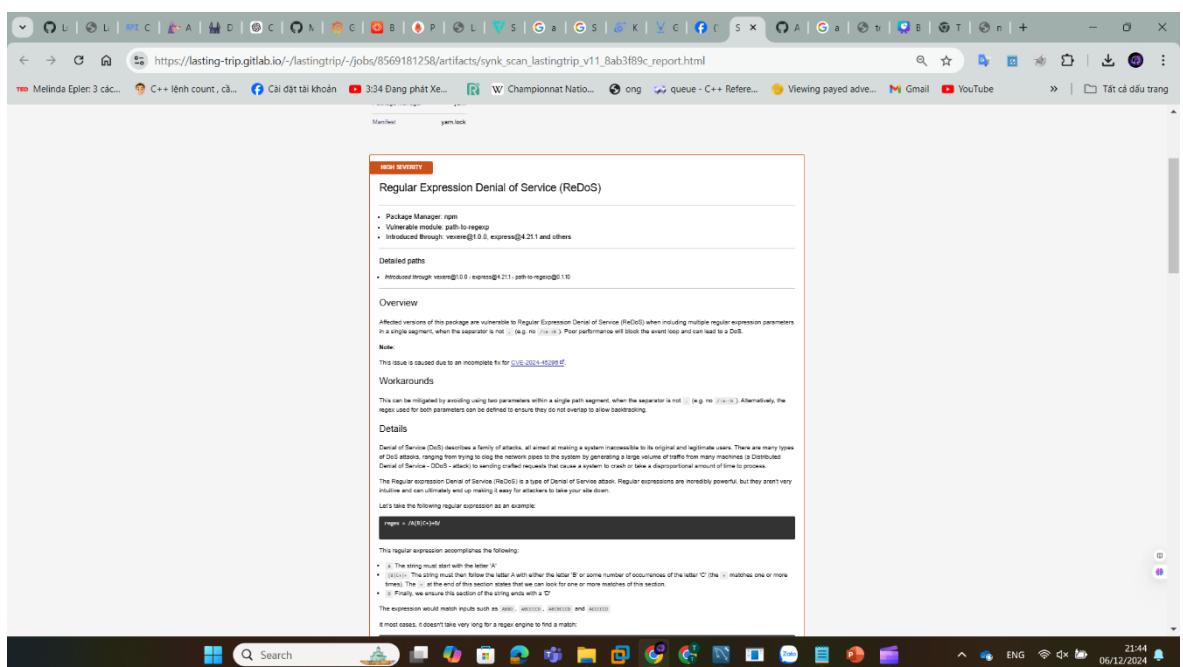
m) *Synk*

Project	vexore
Path	/app
Package Manager	yarn
Manifest	yarn.lock

Hình 96. Kết quả quét Snyk



Hình 97. Lỗi hổng



Hình 98.

Link: https://lasting-trip.gitlab.io/-/lastingtrip/-/jobs/8569181258/artifacts/synk_scan_lastingtrip_v11_8ab3f89c_report.html

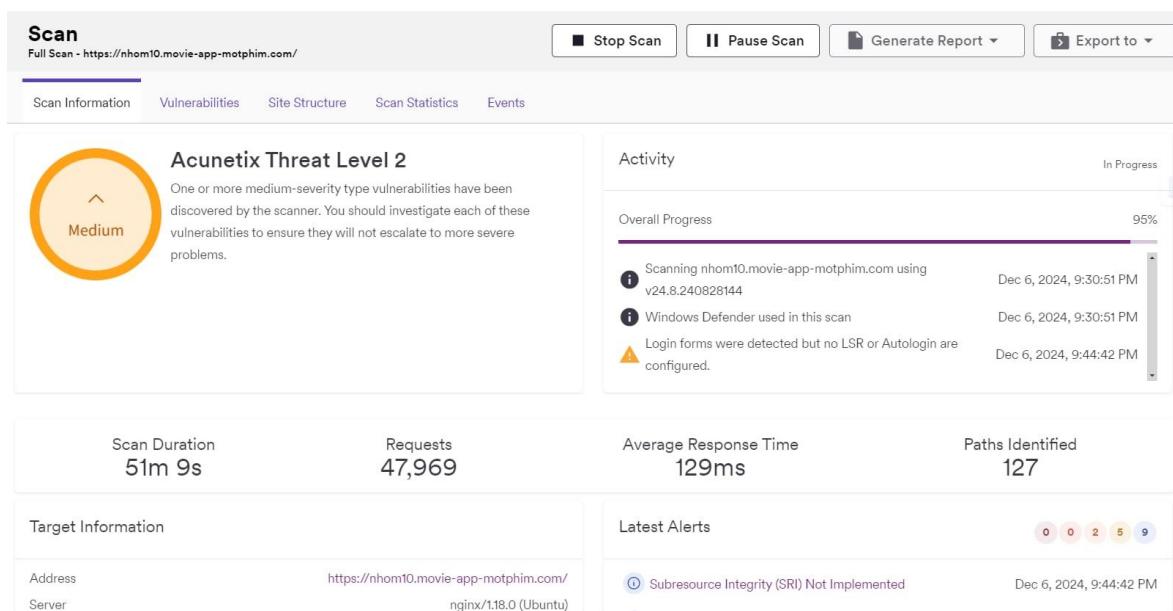
n) K6



Hình 99. Kết quả thực hiện quét trên k6

Link: https://lasting-trip.gitlab.io/-/lastingtrip/-/jobs/8569229005/artifacts/performace_test_lastingtrip_v12_8ab3f89c_report.html

o) Acunetix



Hình 100. Kết quả thực hiện quét trên Acunetix

Scan
Full Scan - <https://nhom10.movie-app-motphim.com/>

Stop Scan | Pause Scan | Generate Report | Export to

Scan Information | Vulnerabilities | Site Structure | Scan Statistics | Events

Filter X III >4

Severity	Vulnerability	URL	Parameter	Status	Confidence %
Medium	Node.js Running in Development Mode	https://nhom10.movie-app-motphim.com/		Open	95
Medium	URL redirection (Web Server)	https://nhom10.movie-app-motphim.com/		Open	95
Low	Clickjacking: CSP frame-ancestors missing	https://nhom10.movie-app-motphim.com/		Open	95
Low	Cookies Not Marked as Secure	https://nhom10.movie-app-motphim.com/		Open	100
Low	Insecure Frame (External)	https://nhom10.movie-app-motphim.com/chatbot		Open	100

Hình 101. Kết quả thực hiện quét trên Acunetix

11. Demo

- Video Demo chúng em : [Google Drive](#)

Phân công Hoàn thành nhiệm vụ

STT	Họ và tên	MSSV	Mức độ
1	Trịnh Thị Bích Thảo	22521376	25%
2	Huỳnh Trung Thuận	22521444	25%
3	Lê Hiệp Thuận	22521446	25%
4	Phạm Trung Thành	22521360	25%

HẾT