**BỘ KHOA HỌC VÀ CÔNG NGHỆ**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

-----------------------------



**BÁO CÁO   
ĐỒ ÁN CUỐI KỲ LẬP TRÌNH WEB**

**Môn học :** Lập trình Web

**Khoa :** Công Nghệ Thông Tin 2

**Giảng viên hướng dẫn:** Nguyễn Văn Hữu Hoàng

**Thực hiện bởi nhóm sinh viên, bao gồm:**

|  |  |  |
| --- | --- | --- |
| **TÊN** | **MSSV** | **LỚP** |
| 1. **Lâm Nhật Tiên** | **N23DCCN198** | **D23CQCN03-N** |
| 1. **Lê Viết Xuân** | **N23DCCN069** | **D23CQCN01-N** |

**TP.HCM, ngày 16/06/2025**

# MỤC LỤC

[MỤC LỤC 2](#_Toc201701585)

[Chương 1. GIỚI THIỆU ĐỀ TÀI 3](#_Toc201701586)

[Chương 2 : Phân tích hệ thống 6](#_Toc201701587)

[Chương 3 : Thiết kế hệ thống 44](#_Toc201701588)

[3.1. Thiết kế cơ sở dữ liệu : 44](#_Toc201701589)

[3.2. Thiết kế kiến trúc hệ thống : 47](#_Toc201701590)

[3.3. Các sơ đồ khác 51](#_Toc201701591)

[Chương 4 : Cài đặt – triển khai 52](#_Toc201701592)

[4.1. Môi trường phát triển : 52](#_Toc201701593)

[4.2. Cài đặt Frontend : 53](#_Toc201701594)

[4.3. Cài đặt Backend : 58](#_Toc201701595)

[4.4. Kết nối và xử lý dữ liệu : 58](#_Toc201701596)

[4.4.1. Kết nối cơ sở dữ liệu 59](#_Toc201701597)

[4.5. Đóng gói và triển khai : 59](#_Toc201701598)

[4.5.1. Mô hình triển khai 59](#_Toc201701599)

[4.5.2. Đóng gói hệ thống 60](#_Toc201701600)

[4.5.3. Cấu hình Reverse Proxy bằng Nginx 60](#_Toc201701601)

[4.5.4. Tự động triển khai bằng GitHub Actions (CI/CD) 63](#_Toc201701602)

[Chương 5 : Kết quả đạt được 67](#_Toc201701603)

[5.1. Các phần đạt được: 67](#_Toc201701604)

[5.2. Khó khăn, hạn chế 86](#_Toc201701605)

[5.3. Hướng mở rộng trong tương lai 86](#_Toc201701606)

[Chương 6: Kết luận và Hướng phát triển 87](#_Toc201701607)

[6.1. Kết luận 87](#_Toc201701608)

[6.2. Hướng phát triển 87](#_Toc201701609)

# Chương 1. GIỚI THIỆU ĐỀ TÀI

* 1. **Lý do chọn đề tài :**

Trong bối cảnh thị trường xe cũ ngày càng phát triển, người dùng có xu hướng tìm kiếm các nền tảng trực tuyến để **đăng bán và mua xe đã qua sử dụng** một cách nhanh chóng và tiện lợi. Tuy nhiên, hiện nay nhiều trang web chưa tối ưu trải nghiệm người dùng, thiếu sự tương tác trực tiếp giữa người mua và người bán, hoặc không thân thiện với người dùng phổ thông.

Nhằm giải quyết vấn đề đó, đề tài **“Xây dựng website FakeAuto – nền tảng đăng bán xe trực tuyến”** được lựa chọn với mục tiêu tạo ra một website thân thiện, dễ sử dụng, cho phép người dùng **đăng tin bán xe, quản lý tin đăng, trò chuyện trực tiếp**, đồng thời hỗ trợ người mua dễ dàng **tìm kiếm và lọc xe theo nhu cầu**.

Đề tài này vừa mang tính thực tiễn cao, vừa giúp sinh viên rèn luyện kỹ năng xây dựng một ứng dụng web hiện đại, tích hợp cả frontend, backend và giao tiếp realtime bằng WebSocket.

* 1. **Mục tiêu đề tài :**
* Xây dựng website **FakeAuto** với các chức năng chính:
  + Đăng ký, đăng nhập tài khoản người dùng
  + Đăng tin bán xe (tiêu đề, mô tả, ảnh, giá,...)
  + Tìm kiếm, lọc, phân trang danh sách xe
  + Giao tiếp trực tiếp giữa người mua và người bán qua hệ thống nhắn tin
  + Quản lý tin đăng (chỉnh sửa, xoá, cập nhật trạng thái)
* Tạo giao diện thân thiện, phản hồi nhanh, tương thích mobile
* Xây dựng trang quản trị (Admin) để theo dõi và quản lý nội dung
  1. **Phạm vi thực hiện :**

|  |  |
| --- | --- |
| **Vai trò** | **Phạm vi** |
| Người dùng | * Đăng nhập/đăng ký qua tài khoản web, google * Xem, tìm kiếm, lọc xe theo (hãng, giá, năm, số km đã đi, địa điểm, nhiên liệu, số chỗ ngồi, …) * Nhắn tin với người bán trực tiếp qua hệ thống web * Xem thông tin người mua (số sao đánh giá, số lượt bán xe, thông tin kết nối – facebook, zalo, email, linkedin) * Xem lịch sử mua xe * Đánh giá đơn mua xe * Chăm sóc khách hàng |
| Người bán | * Tạo đơn yêu cầu đăng bán xe * Quản lý tin xin đăng bán xe, lọc theo trạng thái, tên, sắp xếp theo ngày tạo đơn * Quản lý xe bán xe đăng bán, thay đổi trạng thái ẩn <-> bán <-> đặt cọc, lọc theo trạng thái, giá, tìm theo tên * Xem thống kê lượt xem, tương tác, doanh thu |
| Nhân viên | * Chụp ảnh xe * Kiểm tra các tiêu chí xe xem đã đúng với mô tả xe chưa (tên, năm mua, km đã đi, số ghế ngồi, nhiên liệu) * Từ các tiêu chí có thể từ chối/ chấp nhận yêu cầu đăng bán xe (từ chối thì ghi mô tả lý do từ chối) |
| Quản trị viên | * Quản lý nhân viên đến kiểm tra xe |

* 1. **Công cụ sử dụng :**

Database :

MongoDB : Cơ sở dữ liệu NoSQL giúp dễ mở rộng và phù hợp với ứng dụng hiện đại, hiệu suất cao

React Vite (Frontend) :

|  |  |
| --- | --- |
| **Công cụ** | **Mô tả** |
| @cloudinary/react | Thư viện Cloudinary dùng trong React để xử lý ảnh, upload và tối ưu. |
| @cloudinary/url-gen | Tạo URL động cho ảnh/video từ Cloudinary |
| @react-google-maps/api | Thư viện dùng để tích hợp Google Maps vào React dễ dàng. |
| @reduxjs/toolkit | |  | | --- | |  |  |  | | --- | | Bộ công cụ hiện đại giúp viết Redux đơn giản hơn. | |
| @tailwindcss/vite | Plugin tích hợp Tailwind CSS với Vite. |
| axios | Thư viện gọi HTTP request (REST API). |
| bootstrap | Framework CSS phổ biến giúp thiết kế UI nhanh chóng. |
| chart.js | Thư viện vẽ biểu đồ bằng JavaScript. |
| lucide-react | Bộ icon hiện đại, nhẹ dành cho React. |
| react | Thư viện chính để xây dựng giao diện người dùng. |
| react-chartjs-2 | |  | | --- | |  |  |  | | --- | | Bao bọc Chart.js để dùng trong React. | |
| react-dom | Tích hợp React vào DOM trình duyệt. |
| react-icons | Thư viện biểu tượng gồm nhiều bộ icon như FontAwesome, Material... |
| react-loading-skeleton | Hiển thị hiệu ứng loading kiểu "xương" (skeleton screen). |
| react-redux | Kết nối React với Redux. |
| react-router-dom | Thư viện định tuyến (routing) cho ứng dụng React trên web. |
| react-toastify | Hiển thị thông báo (toast) đơn giản và đẹp trong React. |
| sass | Tiền xử lý CSS giúp viết CSS hiệu quả hơn. |
| swiper | Tạo các slider/carousel mượt mà. |

ExpressJS (Backend) :

|  |  |
| --- | --- |
| axios | Gửi HTTP request từ server (tương tự frontend). |
| bcrypt | Hash mật khẩu người dùng một cách an toàn. |
| cloudinary | Tích hợp Cloudinary để lưu trữ và xử lý ảnh/video. |
| cookie-parser | Middleware để đọc cookies từ request. |
| dotenv | Đọc biến môi trường từ file .env. |
| express | Framework backend nhẹ và phổ biến nhất cho Node.js. |
| fontkit | Xử lý và phân tích font (thường dùng trong tạo PDF, xuất văn bản...). |
| express-session | Quản lý phiên (session) trên server (cookie-based login). |
| jsonwebtoken | Tạo và xác thực JWT (JSON Web Token). |
| md5 | Mã hóa chuỗi bằng thuật toán MD5 (ít dùng cho bảo mật). |
| mongodb | Thư viện chính để kết nối MongoDB gốc (không dùng ORM). |
| mongoose | ORM giúp thao tác với MongoDB dễ dàng bằng mô hình schema. |
| mongoose-slug-updater | Plugin tạo và cập nhật "slug" tự động cho tài liệu MongoDB. |
| node-cron | Tạo các tác vụ chạy định kỳ (scheduler) như cron job. |
| node-telegram-bot-api | Tích hợp với bot Telegram dễ dàng từ Node.js. |
| nodemailer | Gửi email từ server (SMTP). |
| nodemon | Tự động reload server khi thay đổi file trong quá trình dev. |
| passport | Middleware xác thực người dùng. |
| passport-google-oauth20 | Cung cấp xác thực Google OAuth 2.0 cho Passport. |
| pdf-lib | |  | | --- | |  |  |  | | --- | | Tạo, chỉnh sửa file PDF trực tiếp bằng JavaScript. | |
| socket.io | Giao tiếp thời gian thực giữa client-server qua WebSocket. |
| bufferutil | |  | | --- | |  |  |  | | --- | | Tăng hiệu suất encode/decode dữ liệu WebSocket. | |
| utf-8-validate | Giúp kiểm tra dữ liệu WebSocket có hợp lệ UTF-8 không, tối ưu kết nối. |
| streamifier | Chuyển buffer hoặc chuỗi thành stream (dùng trong upload ảnh, video, PDF...). |
| validator | Kiểm tra, xác thực chuỗi (email, URL, v.v.) an toàn và nhanh chóng. |

Deploy:

|  |  |
| --- | --- |
| **Tên** | **Mô tả** |
| Domain | Tên miền mua từ nhanhoa.com |
| VPS Linux | Ubuntu 22.04 mua từ VnData |
| Nodejs | Cài |
| Github | Ssh với vps |
| Github CI/CD | Giúp tự động triển khai lại code mới nhất từ github khi người khác push code lên |
| Nginx | Làm reverse proxy cho frontend/backend |
| Npm | Cài đặt thư viện |
| Pm2 | Quản lý tiến trình cho frontend + backend giúp không mất kết nối khi tắt ide |
| Discord webhook | Thông báo khi github action tự deploy code thành công/ thất bại |

# Chương 2 : Phân tích hệ thống

* 1. **Mục tiêu hệ thống :**

Hệ thống website “fakeauto.id.vn” được xây dựng nhằm mục tiêu tạo ra một nền tảng trung gian kết nối giữa người có nhu cầu mua và bán xe. Hệ thống hỗ trợ người dùng đăng tin, tìm kiếm, nhắn tin trực tiếp và quản lý thông tin xe. Qua đó giúp người dùng dễ dàng trao đổi, mua bán xe nhanh chóng và hiệu quả.

* 1. **Phân tích nghiệp vụ :**

Trong mô hình hệ thống, có 3 đối tượng chính :

* **Người bán** : Đăng ký, đăng nhập, đăng tin bán xe, quản lý tin bán xe, tạo đơn bán xe, quản lý đơn bán xe, xem chi tiết, chỉnh sửa
* **Người mua** : Tìm kiếm xe, xem chi tiết, nhắn tin người bán, xem thông tin người bán
* **Quản trị viên :** Quản lý, thống kê các đối tượng trong hệ thống như người dùng, xe, các đơn đặt, các khoản thanh toán,…
  1. **Phân tích dữ liệu :**

|  |  |  |
| --- | --- | --- |
| **Model** | **Các trường** | **Các quan hệ** |
| Car (tin đăng xe) | * Title : tiêu đề xe * Category\_id : chứa id của hãng xe * Location : tên địa điểm * Price : giá * Year : năm mua xe * Km : km đã đi * Seat\_capacity : số ghế ngồi * Comment : mô tả bởi người bán * Img\_src : Một mảng các ảnh * Slug : chứa slug của tin đăng * SellerId : chứa id của người bán * Status : tình trạng của xe (đang bán, đặt cọc, ẩn, đã bán) * Time\_sold : Thời gian bán xe nếu xe đã bán * createdAt : chứa thời gian tạo tin * updatedAt : chứa thời gian cập nhật tin | - Khi đã duyệt xong đơn của người bán, người bán nhập giá, mô tả của xe sẽ tạo nên tin đăng xe (car)  - Tìm kiếm xe đang bán đối với người dùng có thể lọc bởi giá, năm, km, nhiên liệu, số chỗ ngồi, tiêu đề  - Xem chi tiết xe theo slug xe, đang bán đối với người dùng, chi tiết đối với chủ sỡ hữu  - Tìm kiếm các xe đối với người đăng bán  - Chỉnh sửa tin bán xe bởi người bán (giá bán, mô tả, trạng thái) |
| Category | * Title : tiêu đề hãng xe * Parent\_id : chứa id của hãng xe cha * Description : chứa mô tả của hãng xe * Thumbnail : chứa ảnh đại diện của hãng xe * Position : chứa vị trí của hãng xe * Slug : chứa slug của hãng xe * Deleted : đã xóa mềm hay chưa | - Lấy danh sách các hãng xe  - Tạo thêm hãng xe |
| Chat | * SenderId : chứa id của người gửi * Text : chứa nội dung tin nhắn * Seen : đã seen hay chưa * Timestamp : chứa thời gian gửi tin | - Khi người dùng nhắn tin với người bán sẽ tạo tin nhắn |
| Otp | * UserId : chứa id của người dùng * Otp : chứa otp * createdAt : chứa thời gian tạo * expieredAt : là thời gian hết hạn |  |
| Request\_add (tin xin đăng bán xe) | * name : tên của form tin * year : thời gian mua xe của xe * km : số km đã đi của xe * fuel : loại nhiên liệu * seat\_capacity : số chỗ ngồi của xe * location : địa điểm * img\_demo : ảnh demo của xe * status : tình trạng của đơn (chờ, đã duyệt, từ chối, hoàn tất) * img\_src : ảnh của xe * sellerId : chứa id của người bán * slug : chứa slug của tin * userIds : chứa các id của nhân viên kiểm tra * price\_recommend\_low : chứa giá thấp khuyến nghị * price\_recommend\_high : chứa giá cao khuyến nghị * message : Nội dung chi tiết từ chối đơn – nếu từ chối * examine : chứa các chi tiết kiểm tra * createdAt : chứa ngày tạo đơn | - Tạo đơn xin đăng bán xe  Lấy thông tin xin đăng bán xe của người bán  - Lấy thông tin chi tiết của đơn bởi slug  - Phái nhân viên đến kiểm tra thêm vào userIds  - Đánh dấu là đã duyệt bởi nhân viên  - Đánh dấu là từ chối bởi nhân viên  - Đánh dấu là hoàn thành–done nếu người dùng chọn nhập giá, mô tả cho tin đăng sau đó tạo tin đăng (car model ở trên) |
| Views (thống kê lượt xem) | * sellerId : chứa id của người bán (chứa tin xem) * views : gồm một mảng các object ( counts : số lượng lượt xem, time : ngày dạng ‘dd/mm/yyyy’) | - Khi người dùng chọn xem chi tiết xe ở trang chủ thì sẽ tăng số lượng lượt xem lên cho chủ thể xe đó  - Lấy thống kê lượt xem kể từ lúc người bán tạo tài khoản đến lúc mới xem |
| Contacts (thống kê lượt tương tác – Bắt đầu chat) | * sellerId : chứa id của người bán (người được chat đến) * contacts : gồm một mảng các object (counts : số lượng lượt tương tác, time : ngày dạng ‘dd/mm/yyyy’) | - Khi người đã vào xem chi tiết xe nhấn vào nhắn với người bán thì sẽ thêm số lượng tương tác cho chủ thể xe đó  - Lấy thống kê tương tác kể từ lúc người bán tạo tài khoản đến lúc mới xem |
| Thread (chủ đề chat) | * user1Id : chứa id của người thứ nhất * user2Id : chứa id của người thứ hai * msgIds : chứa một mảng gồm các id của tin nhắn (chat model) | - Khi người dùng lần đầu nhắn tin với người bán xe thì sẽ tạo chủ đề chat  - Nhắn tin thì sẽ tạo nội dung tin nhắn và lưu id của nó thêm vào danh sách msgIds |
| User | * name : tên người dùng * email : email người dùng * phone : số điện thoại người dùng * pendingEmail : Email chờ trong quá trình đổi email mới cho tài khoản * password : mật khẩu tài khoản * role : vai trò của người dùng (user – người dùng, seller – người bán, staff – nhân viên, admin – quản trị viên) * address : chứa địa chỉ của người dùng * city : thành phố/ tỉnh của người dùng * district : quận/huyện của người dùng * avatar : chứa ảnh đại diện của người dùng * isVerified : người dùng đã xác thực email chưa * is2FAEnabled : người dùng có bật xác minh 2 bước không * contactFacebook : chứa thông tin liên hệ - đường dẫn facebook * contactZalo : chứa số điện thoại Zalo * contactEmail : chứa địa chỉ liên hệ email * contactLinkedin : chứa thông tin liên hệ - đường dẫn linkedin * slug : slug của người dùng | - Khi người dùng đăng ký/ đăng nhập bằng hệ thống web hay qua oauth sẽ tạo tài khoản  - Chỉnh sửa thông tin người dùng khi người dùng vào trang my\_account chỉnh sửa thông tin thêm ảnh/ chỉnh sửa tên/ thêm thông tin liên hệ    - Trở thành người bán khi là người dùng và nhập thông tin đầy đủ  - Nếu là người bán mà có đăng bán thì sẽ có thể được xem chi tiết thông tin bởi người dùng |
| UserToken | * userId: Id của user chủ token * token: token (đã được hash) * createdAt: thời điểm token được tạo * token hết hạn sau 1 ngày |  |

* 1. **Phân tích luồng hoạt động :**

|  |  |  |
| --- | --- | --- |
| **Trang** | **Thao tác** | **Kết quả** |
| Trang chủ | - - - | Hiển thị tất cả xe đang bán ra |
|  | Thực hiện lọc (địa điểm, giá, năm, số km, nhiên liệu, chỗ ngồi, tên) | Hiển thị các xe đáp ứng với bộ lọc tìm kiếm |
| Đăng ký | Đăng ký qua hệ thống (nhập họ tên, email, mật khẩu) | Chuyển hướng qua trang login |
|  | Đăng ký qua hệ thống Oauth | Đăng nhập thẳng vào hệ thống, chuyển hướng đến trang chủ |
| Đăng nhập | Đăng nhập qua hệ thống (nhập email, mật khẩu) | Đăng nhập thành công, chuyển hướng đến trang chủ |
|  | Đăng nhập qua Oauth | Đăng nhập thành công, chuyển hướng đến trang chủ |
| Dashboard(chỉnh sửa thông tin) | Chỉnh sửa thông tin (ảnh đại diện, họ và tên, số điện thoại, địa chỉ, tỉnh, thành phố, quận huyện), thêm thông tin liên hệ (facebook, email, zalo, linkedin) -> xác nhận | Hiển thị thông báo chỉnh sửa thành công và reload lại trang |
| Dashboard(Đổi Email, mật khẩu) | Nhấn vào nút chuyển email hoặc chuyển mật khẩu -> nhập mã otp chuyển qua email -> nhập mật khẩu/email mới -> xác nhận | Chuyển đổi mật khẩu/email thành công |
| Quản lý đăng bán xe | --- | Hiển thị tất cả tin do người bán (chủ sở hữu) đăng giá, trạng thái và chức năng (xem chi tiết, chỉnh sửa, xóa) đối với tin đăng đó |
| Lọc theo trạng thái, tìm kiếm, sort | Hiển thị tin tương ứng với các trạng thái, tên và giá được sắp xếp đúng với bộ lọc đã chọn |
| Nhấn vào thêm mới | Chuyển hướng tới trang đăng ký bán xe |
| Nhấn vào nút chi tiết | Chuyển hướng tới trang chi tiết xe |
| Nhấn vào nút chỉnh sửa | Chuyển hướng tới trang chỉnh sửa tin bán xe |
| Nhấn vào nút xóa (đối với xe chưa bán) -> xác nhận | Hiển thị thông báo xóa thành công xe |
| Chỉnh sửa tin bán xe | Nhập vào giá, Mô tả, trạng thái -> xác nhận | Thông báo xác nhận chỉnh sửa thành công -> load lại trang |
| Đăng ký bán xe | Nhập thông tin xe muốn bán (tên, năm mua, số km đã đi, loại nhiên liệu, số chỗ ngồi, địa điểm muốn kiểm tra và ảnh đại diện của xe) -> nút xác nhận | Tạo một yêu cầu bán xe với trạng thái chờ - pending, chuyển hướng về Thông tin/ lịch sử kiểm định |
| Thống kê bán hàng | - - - | Hiển thị biểu đồ cột doanh thu với người bán dựa trên thời gian (ngày). Cột dọc : triệu – VNĐ Cột ngang : ngày |
| Thống kê lượt xem/ tương tác | - - - | Hiển thị 2 biểu đồ đường hiển thị lượt xem, tương tác  Cột dọc : số lượng xem/ tương tác  Cột ngang : Thời gian (ngày) |
| Thông tin/ lịch sử kiểm định | - - - | Hiển thị tất cả đơn xin bán xe với tiêu đề, ngày làm đơn, trạng thái và chức năng (chỉnh sửa, xóa – đối với tin đang chờ duyệt) |
| Lọc theo trạng thái, tìm kiếm theo tiêu đề và sắp xếp theo ngày làm đơn | Hiển thị tin tương ứng với bộ lọc |
| Nhấn vào nút chỉnh sửa | Chuyển hướng tới trang chi tiết đơn đăng bán (tùy theo trạng thái đơn) mà chuyển hướng đến trang có tiến độ phù hợp |
| Chi tiết đơn đăng bán (điền thông tin) | - - - | Hiển thị thông tin lúc tạo đơn đăng bán xe |
| Chi tiết đơn đăng bán (đợi xác thực) | Đơn trong trạng thái chờ duyệt và chưa có chọn nhân viên kiểm tra | Hiển thị đang trong quá trình chọn nhân viên |
| Đơn trong trạng thái đã duyệt | Hiển thị thông tin nhân viên, hình ảnh xe, giá khuyến nghị, phiếu điền thông tin giá, mô tả xe trước khi bán |
| Đơn trong trạng thái hoàn thành | Hiển thị hình ảnh thông tin nhân viên, hình ảnh xe, giá khuyến nghị |
| Đơn trong trạng thái từ chối | Hiển thị thông tin nhân viên, lý do từ chối |
| Chi tiết đơn (Xác thực) | Đơn trong trạng thái đã duyệt/ hoàn thành | Hiển thị thông qua kèm pdf chi tiết kiểm tra xe với thông tin/ tiêu chí và con dấu công ty |
| Đơn trong trạng thái từ chối | Hiển thị không qua kèm pdf chi tiết kiểm tra xe với lý do từ chối kèm con dấu công ty |
| Thông báo | - - - | Hiển thị các thông báo với người dùng |
| Nhấn tắt thông báo | Tắt thông báo |
| Hướng dẫn/ FAQ (trong trang của tôi) | - - - | Biểu thị những câu hỏi thường gặp với website |
| FAQ (trong trang chủ) | - - - | Hiển thị những câu hỏi thường gặp, thông tin liên hệ/ hotline |
| Trang liên hệ | - - - | Hiển thị thông tin liên hệ, form – nếu muốn kết nối, link kết nối, google map |
| Trang bán xe | - - - | Hiển thị trang bán xe nhằm tăng uy tín cho trang |
| Trang giới thiệu | - - - | Hiển thị trang giới thiệu nhằm tăng uy tín cho trang |
| Chính sách, quyền riêng tư | - - - | Hiển thị chính sách về tài khoản, dữ liệu, thông tin người dùng |
| Điều khoản sử dụng | - - - | Hiển thị điều khoản sử dụng của website |
| Chi tiết xe | - - - | Hiển thị chi tiết xe gồm ảnh, tiêu đề, mô tả, giá, năm, nhiên liệu, số km, số chỗ ngồi, địa điểm, thông tin liên hệ người bán |
| Nhấn vào mua xe | Mở form cho phép người dùng chọn phương thức thanh toán, giá trị cọc (nếu chọn phương thức đặt cọc), chọn xong người dùng bấm tạo đơn hàng hệ thống sẽ tạo đơn hàng và xuất hóa đơn có mã QR thanh toán, người dùng quét mã thanh toán xong và xác nhận thanh toán thành công sẽ nhận form thông báo thanh toán thành công. |
| Nhấn vào nhắn người bán | Tạo cuộc hội thoại với người bán |
| Nhấn vào tên người bán | Tới trang chi tiết xem người bán |
| Chi tiết người bán | - - - | Hiển thị thông tin người bán, tổng đơn, với đánh giá trung bình |
| Trang nhắn tin | - - - | Hiển thị người nhắn và mô tả ngắn đoạn tin, trạng thái online hay không |
| Nhấn vào người nhắn bất kì | Hiển thị lịch sử tin nhắn, thời gian nhắn |
| Admin | Admin đăng nhập sẽ được đưa đến admin dashboard thay vì trang home | Trang Dashboard hiển thị các thống kê về doanh thu, người dùng, xe,… và hiển thị các hoạt động gần đây trên hệ thống |
| Admin có thể vào trang quản lý người dùng | Cho phép admin lọc, tìm kiếm người dùng, thực hiện các thao tác lên người dùng như xem, xóa, thay đổi vai trò của người dùng, vô hiệu hóa hoặc kích hoạt tài khoản người dùng |
| Admin có thể vào trang quản lý xe | Cho phép admin lọc, tìm kiếm xe, có thể xem chi tiết xe, phân công người kiểm định xe đăng bán trên hệ thống. |
| Admin có thể vào trang quản lý đơn hàng và trang quản lý thanh toán | Cho phép admin lọc, tìm kiếm các đơn hàng, các khoản thanh toán và theo dõi chúng, Admin có thể thay đổi trạng thái các đơn hàng và có thể xác minh hoặc từ chối các thanh toán trên hệ thống, khi đã xác minh một thanh toán, admin có thể chọn gửi thông báo đến người mua và người bán về thanh toán vừa xác minh và admin có thể chọn mở cuộc hội thoại giữa hệ thống với người mua và hệ thống với người bán để trao đổi thông tin thanh toán,… |

* 1. **Các endpoint api :**

**AUTH:**

|  |
| --- |
|  |
| * **Cách hoạt động:** Nhận thông tin đăng ký (email, password, tên, v.v.), tạo user mới (chưa xác thực), sinh OTP gửi về email để xác thực tài khoản. * **Đầu vào:** JSON gồm email, password, name, v.v. * **Đầu ra:** 201 nếu thành công, 400 nếu dữ liệu không hợp lệ, 500 nếu lỗi server. * **Lưu ý:** Không giới hạn số lần gửi OTP khi đăng ký. |
|  |
| * **Cách hoạt động:** Nhận email và password, kiểm tra thông tin, nếu đúng và user đã xác thực thì trả về accessToken/refreshToken. Nếu bật 2FA thì gửi OTP về email. * **Đầu vào:** JSON gồm email, password. * **Đầu ra:** 200 (token hoặc yêu cầu xác thực OTP), 400 nếu sai thông tin, 500 nếu lỗi server. |
|  |
| * **Cách hoạt động:** Nhận OTP từ user, kiểm tra với DB, nếu đúng và chưa hết hạn thì xác thực user cho phép đăng nhập, xóa OTP sau khi xác thực. * **Đầu vào:** JSON gồm email, otp, type. * **Đầu ra:** 200 nếu xác minh thành công, 400 nếu sai OTP, 500 nếu lỗi server. |
|  |
| * **Cách hoạt động:** Sinh mã OTP mới, lưu vào DB (ghi đè nếu đã có), gửi email cho user. Không kiểm tra số lần gửi trong 1 khoảng thời gian. * **Đầu vào:** JSON gồm email, type (loại OTP: đăng ký, đổi mật khẩu, v.v.). * **Đầu ra:** 200 nếu gửi thành công, 400 nếu lỗi dữ liệu, 500 nếu lỗi server. * **Lưu ý:** Không có giới hạn số lần gửi OTP trong 5 phút. |
|  |
| * **Cách hoạt động:** Nhận OTP từ user, kiểm tra với DB, nếu đúng và chưa hết hạn thì xác thực user hoặc thực hiện hành động tương ứng (đăng ký, đổi email, v.v.), xóa OTP sau khi xác thực. * **Đầu vào:** JSON gồm email, otp, type. * **Đầu ra:** 200 nếu xác minh thành công, 400 nếu sai OTP, 500 nếu lỗi server. |
|  |
| * **Cách hoạt động:** Nhận mật khẩu cũ và mới, kiểm tra mật khẩu cũ, cập nhật mật khẩu mới (hash). * **Đầu vào:** JWT, JSON gồm oldPassword, newPassword. * **Đầu ra:** 200 nếu thành công, 400 nếu sai dữ liệu, 500 nếu lỗi server. |
|  |
| * **Cách hoạt động:** Nhận email mới, gửi OTP xác thực về email mới, xác minh OTP, cập nhật email. * **Đầu vào:** JWT, JSON gồm newEmail, otp. * **Đầu ra:** 200 nếu thành công, 400 nếu sai dữ liệu, 500 nếu lỗi server. |
|  |
| * **Cách hoạt động:** Nhận email, gửi OTP về email, xác minh OTP, cho phép đặt lại mật khẩu mới. * **Đầu vào:** JSON gồm email, otp, newPassword. * **Đầu ra:** 200 nếu thành công, 400 nếu sai dữ liệu, 500 nếu lỗi server. |
|  |
| * **Cách hoạt động:** Cho phép user đăng nhập bằng OAuth đặt mật khẩu cho tài khoản. * **Đầu vào:** JWT, JSON gồm newPassword. * **Đầu ra:** 200 nếu thành công, 400 nếu sai dữ liệu, 500 nếu lỗi server. |
|  |
| * **Cách hoạt động:** Cho phép user bật/tắt 2FA, cập nhật trạng thái trong DB. * **Đầu vào:** JWT, JSON gồm enable (true/false). * **Đầu ra:** 200 nếu thành công, 400 nếu sai dữ liệu, 500 nếu lỗi server. |
|  |
| * **Cách hoạt động:** Nhận refreshToken, kiểm tra hợp lệ, sinh accessToken mới, refreshToken mới (token rotation). * **Đầu vào:** JSON gồm refreshToken. * **Đầu ra:** 200 (accessToken, refreshToken mới), 400/401 nếu token không hợp lệ, 500 nếu lỗi server. |

**OAUTH:**

|  |  |
| --- | --- |
| |  | | --- | |  |   Người dùng gửi { code } lên query.  Không có code => chuyển hướng về CLIENT\_URL/login?error=oauth\_cancelled  Có code => Gọi api của oauth2 để lấy {id\_token, access\_token}  Từ {id\_token, access\_token} => gọi api của oauth2 để lấy thông tin người dùng trên google  Tìm xem người dùng đã có tài khoản trên hệ thống chưa   * Chưa có thì tạo tài khoản trên model User rồi đưa cho tempToken tên “temp\_oauth\_token” => chuyển hướng đến CLIENT\_URL/fill-info để điền lại thông tin * Đã có thì cấp cho accessToken với refreshToken cho đăng nhập => chuyển hướng đến CLIENT\_URL?oauth\_success=true&user = {thông tin user}   Lỗi trong quá trình chạy => chuyển hướng tới trang CLIENT/login?error=oauth\_failed |
| |  | | --- | |  |   Lấy token = cookies tên “temp\_oauth\_token”   * Không có token trả về  |  |  | | --- | --- | | Status = 401 | Message : “Chưa xác thực google” |   Giải mã token bằng jwt  Tìm user bằng userId   * Không tìm thấy user  |  |  | | --- | --- | | Status 400 | Message : “Người dùng không hợp lệ” |  * Thấy trả về  |  | | --- | | {  Name,  Email,  Avatar  } |   Lỗi trong quá trình chạy   |  |  | | --- | --- | | Status = 401 | Token không hợp lệ | |
| |  | | --- | |  |   Lấy token từ cookies “temp\_oauth\_token”  Lấy { name, phone, email, avatar } từ body  Tìm user và update từ thông tin  Cấp cho accessToken, refreshToken để cấp quyền chạy như một tài khoản chính thức  Hoàn thành trả về   |  | | --- | | {  Success,  Message : “cập nhật thành công”,  Data : {  User : {  Id, email, name, role, is2FAEnaled, isVerified, isOauthUser, hasSetPassword, avatar  }  }  } |   Lỗi trong quá trình chạy   |  |  | | --- | --- | | Status = 500 | Server Error! | |

**ORDER:**

|  |
| --- |
|  |
| **Chức năng**: Người dùng tạo đơn hàng mới cho xe với các phương thức thanh toán khác nhau.  **Cách hoạt động**:   * Kiểm tra xe còn khả dụng (status khác "sold") * Kiểm tra người bán có thông tin ngân hàng đã xác minh (nếu không phải direct\_transaction) * Không cho phép user tự mua xe của chính mình * Tính toán totalAmount, depositAmount, remainingAmount * Tạo Order với status phù hợp: awaiting\_payment hoặc pending\_meeting * Nếu là thanh toán online: tạo mã QR VietQR + bản ghi Payment * Gửi thông báo cho seller * Ghi log ORDER\_STATUS\_CHANGE và PAYMENT\_CREATED   **Đầu vào**:  {  "carId": "string (MongoDB ObjectId)",  "paymentMethod": "deposit | full\_payment | direct\_transaction",  "depositPercentage": 20 // optional, mặc định 20%  }   * JWT token: bắt buộc   **Đầu ra**:   * 201 Created: Trả về order, payment, mã QR và bước tiếp theo * 400 Bad Request: Seller thiếu bank info hoặc user mua xe của chính mình * 404 Not Found: Xe không tồn tại hoặc đã bán * 500 Internal Server Error: Lỗi tạo QR hoặc lỗi hệ thống   **Tác động**: Tạo bản ghi Order + Payment, gửi notification, log hành động |
|  |
| **Chức năng**: Người dùng xem danh sách đơn hàng của mình với bộ lọc và phân trang  **Cách hoạt động**:   * Nếu role="all": trả về 2 mảng buyerOrders và sellerOrders * Nếu role="buyer": chỉ lấy đơn hàng người dùng đã tạo * Nếu role="seller": chỉ lấy đơn hàng mà người khác mua xe của user * Có thể lọc theo status * Populate thông tin buyer, seller, car * Sắp xếp theo createdAt, updatedAt, totalAmount, status * Áp dụng phân trang nếu role khác "all"   **Đầu vào**:  {  "role": "buyer | seller | all",  "status": "string (optional)",  "page": 1,  "limit": 20,  "sortBy": "createdAt",  "sortOrder": "desc"  }   * JWT token: bắt buộc   **Đầu ra**:   * 200 OK:   + Nếu role = all: { buyerOrders[], sellerOrders[], summary }   + Nếu khác: { orders[], pagination } * 401 Unauthorized: Không có quyền truy cập * 500 Internal Server Error: Lỗi hệ thống   **Tác động**: Chỉ đọc dữ liệu |
|  |
| **Chức năng**: Lấy chi tiết một đơn hàng cụ thể  **Cách hoạt động**:   * Tìm order theo orderId * Kiểm tra quyền: chỉ buyer, seller, hoặc admin mới được xem * Populate buyer, seller, car * Lấy danh sách payments và 20 log gần nhất liên quan đến order   **Đầu vào**:   * Path: orderId (MongoDB ObjectId) * JWT token: bắt buộc   **Đầu ra**:   * 200 OK: { order, payments[], logs[] } * 403 Forbidden: Không có quyền xem * 404 Not Found: Không tìm thấy order * 500 Internal Server Error: Lỗi server   **Tác động**: Chỉ đọc dữ liệu |
|  |
| **Chức năng**: Người dùng xác nhận đã thực hiện thanh toán đơn hàng  **Cách hoạt động**:   * Hỗ trợ paymentId hoặc tempPayId * Nếu là temp ID: tìm order gần nhất awaiting\_payment * Cập nhật transactionInfo: bankTransactionId, payerName, transferMessage, evidence * Cập nhật payment.status = PENDING * Tạo log PAYMENT\_CREATED với action "confirm\_payment"   **Đầu vào**:  {  "paymentId": "string",  "transactionId": "string",  "evidence": ["url1", "url2"]  }   * JWT token: bắt buộc   **Đầu ra**:   * 200 OK: { payment, order, paymentId, nextStep } * 404 Not Found: Không tìm thấy payment hoặc order * 500 Internal Server Error: Lỗi server   **Tác động**: Cập nhật transaction info + trạng thái thanh toán |
|  |
| **Chức năng**: Admin xác minh thanh toán từ user  **Cách hoạt động**:   * Yêu cầu quyền admin * Tìm payment và populate order, user * Nếu approved = true:   + Cập nhật payment.status = COMPLETED, verifiedBy, verifiedAt, adminNote   + Cập nhật order status và paidAmount   + Ghi log PAYMENT\_VERIFIED * Nếu approved = false:   + Cập nhật payment.status = REJECTED   + Ghi log PAYMENT\_REJECTED   **Đầu vào**:  {  "paymentId": "string (MongoDB ObjectId)",  "approved": true,  "note": "optional string"  }   * JWT token (admin): bắt buộc   **Đầu ra**:   * 200 OK: { payment, order, nextStep } * 403 Forbidden: Không có quyền admin * 404 Not Found: Không tìm thấy payment hoặc order * 500 Internal Server Error: Lỗi server   **Tác động**: Cập nhật trạng thái payment và order, log lại hành động |

**PAYMENT:**

|  |
| --- |
|  |
| **Chức năng**: Người dùng lấy lịch sử thanh toán của chính mình  **Cách hoạt động**:   * Truy vấn danh sách payments theo userId từ JWT token * Lọc theo status, type nếu có * Phân trang với page, limit * Populate thông tin order (bao gồm car, seller) * Sắp xếp theo createdAt giảm dần * Trả về danh sách + thông tin phân trang   **Đầu vào**:   * JWT token (bắt buộc) * page, limit (tùy chọn) * status, type (tùy chọn)   **Đầu ra**:   * 200 OK: Danh sách payments + pagination * 401 Unauthorized: Không có quyền truy cập * 500 Internal Server Error: Lỗi server   **Tác động**: Chỉ đọc dữ liệu, không thay đổi trạng thái hệ thống |
|  |
| **Chức năng**: Admin lấy danh sách các thanh toán đang chờ xác minh  **Cách hoạt động**:   * Truy vấn payments với status = "pending" * Kiểm tra quyền truy cập của admin * Cho phép tìm kiếm theo orderCode, paymentCode * Populate order, buyer, seller, car, user * Sắp xếp theo createdAt giảm dần * Phân trang và trả về kết quả   **Đầu vào**:   * JWT token admin (bắt buộc) * page, limit (tùy chọn) * orderCode, paymentCode (tùy chọn)   **Đầu ra**:   * 200 OK: Danh sách payments pending + pagination * 401 / 403: Không có quyền admin * 500: Lỗi server   **Tác động**: Chỉ đọc dữ liệu, dành riêng cho admin |
|  |
| **Chức năng**: Admin lấy danh sách các payments theo trạng thái  **Cách hoạt động**:   * Nhận status từ query parameter (bắt buộc) * Kiểm tra quyền admin * Truy vấn danh sách payments theo trạng thái đã cho * Populate thông tin liên quan * Phân trang kết quả   **Đầu vào**:   * JWT token admin (bắt buộc) * status (bắt buộc) * page, limit (tùy chọn)   **Đầu ra**:   * 200 OK: Danh sách payments theo status + pagination * 400 Bad Request: Thiếu status parameter * 401 / 403: Không có quyền admin   **Tác động**: Chỉ đọc dữ liệu, phục vụ quản trị viên |
|  |
| **Chức năng**: Admin xác minh thanh toán thủ công  **Cách hoạt động**:   * Tìm payment theo paymentId * Kiểm tra trạng thái hiện tại là "pending" * Cập nhật status thành "completed" (bỏ qua quy trình tự động) * Lưu thông tin xác minh: bankTransactionId, transactionDate, evidence, notes * Cập nhật trạng thái order tương ứng * Ghi log thao tác của admin * Gửi thông báo đến user   **Đầu vào**:   * Path: paymentId (MongoDB ObjectId) * JWT token admin (bắt buộc) * Body:   {  "bankTransactionId": "string",  "transactionDate": "ISODate",  "evidence": ["url1", "url2"],  "notes": "optional string"  }  **Đầu ra**:   * 200 OK: Xác minh thành công, trả về thông tin payment + order * 400 Bad Request: Payment đã được xử lý hoặc dữ liệu không hợp lệ * 401 / 403: Không có quyền admin * 404 Not Found: Không tìm thấy payment   **Tác động**: Cập nhật trạng thái payment và order |
|  |
| **Chức năng**: Admin từ chối thanh toán  **Cách hoạt động**:   * Tìm payment theo paymentId * Kiểm tra trạng thái là "pending" * Cập nhật status thành "cancelled" hoặc "failed" * Lưu lý do từ chối trong notes * Ghi log thao tác admin * Gửi thông báo đến user   **Đầu vào**:   * Path: paymentId (MongoDB ObjectId) * JWT token admin (bắt buộc) * Body:   {  "reason": "string",  "notes": "optional"  }  **Đầu ra**:   * 200 OK: Từ chối thành công * 400: Dữ liệu không hợp lệ * 401 / 403: Không có quyền admin * 404: Không tìm thấy payment   **Tác động**: Kết thúc quy trình thanh toán, có thể mở quy trình hoàn tiền |
|  |
| **Chức năng**: Admin lấy thông tin chi tiết của một payment  **Cách hoạt động**:   * Truy vấn payment theo paymentId * Populate các thông tin liên quan: order, user, admin xác minh/từ chối * Trả về toàn bộ chi tiết giao dịch   **Đầu vào**:   * Path: paymentId (MongoDB ObjectId) * JWT token admin (bắt buộc)   **Đầu ra**:   * 200 OK: Chi tiết payment đầy đủ * 401 / 403: Không có quyền admin * 404: Không tìm thấy payment   **Tác động**: Chỉ đọc dữ liệu, phục vụ debug và kiểm tra |

**NOTIFICATION:**

|  |
| --- |
|  |
| **Chức năng**: Người dùng lấy danh sách thông báo cá nhân với bộ lọc và phân trang  **Cách hoạt động**:   * Tìm thông báo theo userId từ JWT token * Hỗ trợ lọc theo: category, type, read, priority * Phân trang với page, limit (mặc định 20/trang) * Sắp xếp theo createdAt, updatedAt, hoặc priority (mặc định: createdAt desc) * Populate thông tin người gửi: fullName, avatar * Tự động tính unreadCount * Trả về danh sách thông báo, unreadCount, và thông tin phân trang   **Đầu vào**:   * JWT token (bắt buộc) * Các filter tùy chọn: category, type, read, priority * Phân trang tùy chọn: page, limit, sortBy, sortOrder   **Đầu ra**:   * 200 OK: Trả về mảng notifications, unreadCount, và pagination * 401 Unauthorized: Không có quyền truy cập * 500 Internal Server Error: Lỗi server   **Tác động**: Hoàn toàn đọc dữ liệu, không thay đổi gì |
|  |
| **Chức năng**: Người dùng đánh dấu một thông báo cụ thể đã đọc  **Cách hoạt động**:   * Tìm thông báo theo notificationId và userId * Kiểm tra quyền sở hữu (chỉ được đánh dấu thông báo của chính mình) * Cập nhật read = true, readAt = current timestamp * Gửi thông báo cập nhật theo thời gian thực qua **Socket.IO** * Trả về thông báo đã được cập nhật   **Đầu vào**:   * Path: notificationId (MongoDB ObjectId) * JWT token (bắt buộc)   **Đầu ra**:   * 200 OK: Đánh dấu thành công, trả về notification đã cập nhật * 404 Not Found: Không tìm thấy hoặc không có quyền * 401 Unauthorized: Không có quyền truy cập * 500 Internal Server Error: Lỗi server   **Tác động**: Cập nhật trạng thái read và readAt, có truyền real-time |
|  |
| **Chức năng**: Đánh dấu tất cả thông báo (hoặc theo category) là đã đọc  **Cách hoạt động**:   * Tìm tất cả thông báo của người dùng có read = false * Lọc thêm theo category nếu có * Bulk update: tất cả thông báo được cập nhật read = true, readAt = now() * Gửi cập nhật thời gian thực qua **Socket.IO** * Trả về số lượng thông báo đã được cập nhật   **Đầu vào**:   * JWT token (bắt buộc) * Query parameter category (tùy chọn)   **Đầu ra**:   * 200 OK: Trả về modifiedCount * 401 Unauthorized: Không có quyền truy cập * 500 Internal Server Error: Lỗi server   **Tác động**: Cập nhật nhiều thông báo cùng lúc, có gửi real-time update |
|  |
| **Chức năng**: Người dùng xóa một thông báo cụ thể của chính mình  **Cách hoạt động**:   * Tìm thông báo theo notificationId và userId * Kiểm tra quyền sở hữu * Xóa thông báo bằng findOneAndDelete * Trả về kết quả xóa thành công   **Đầu vào**:   * Path: notificationId (MongoDB ObjectId) * JWT token (bắt buộc)   **Đầu ra**:   * 200 OK: Xóa thành công * 404 Not Found: Không tìm thấy hoặc không có quyền * 401 Unauthorized: Không có quyền truy cập * 500 Internal Server Error: Lỗi server   **Tác động**: Xóa vĩnh viễn thông báo khỏi database |

**ADMIN:**

|  |
| --- |
|  |
| **Chức năng:** Admin lấy thống kê tổng quan cho dashboard quản trị  **Cách hoạt động:**   * Đếm tổng số người dùng theo vai trò: totalUsers, totalSellers, totalBuyers * Thống kê xe: totalCars, activeCars, soldCars * Thống kê đơn hàng: totalOrders, completedOrders, pendingOrders * Thống kê yêu cầu: totalRequests, pendingRequests, approvedRequests, rejectedRequests * Tính tổng doanh thu từ các đơn hàng đã hoàn tất * Tạo dữ liệu biểu đồ doanh thu 12 tháng gần nhất (monthlyRevenue) * Lấy các hoạt động gần đây: đơn hàng, yêu cầu, người dùng * Thống kê trạng thái đơn hàng và danh mục xe * Thống kê thanh toán và hoàn tiền   **Đầu vào:**   * JWT token của admin (**required**)   **Đầu ra:**   * 200: Thành công + {overview, charts, recentActivities} * 403: Không có quyền admin * 500: Lỗi server hoặc database   **Tác động:** Chỉ đọc, không thay đổi dữ liệu |
|  |
| **Chức năng:** Admin lấy dữ liệu phân tích chi tiết theo thời gian  **Cách hoạt động:**   * Parse tham số period ("7d", "30d", "90d", "1y") * Tính toán dateRange từ period * Phân tích tăng trưởng người dùng theo thời gian (userGrowthTrend) * Thống kê tỷ lệ user seller vs buyer * Xu hướng đăng tin xe (carPostingTrends) * Thống kê xe theo loại nhiên liệu, năm sản xuất * Lấy logs hoạt động trong khoảng thời gian đó * Xu hướng đơn hàng và top xe bán tốt nhất * So sánh doanh thu hiện tại so với kỳ trước (revenueComparison)   **Đầu vào:**   * JWT token của admin (**required**) * period: "7d" | "30d" | "90d" | "1y" (mặc định "30d")   **Đầu ra:**   * 200: Thành công + dữ liệu phân tích chi tiết * 403: Không có quyền admin * 500: Lỗi server   **Tác động:** Chỉ đọc, không thay đổi dữ liệu |
|  |
| **Chức năng:** Admin lấy danh sách người dùng với các bộ lọc và phân trang.  **Cách hoạt động:**   * Parse các query parameters: page, limit, search, role, status, sortBy, sortOrder. * Tạo filter từ search (theo name, email, phone), role, status. * Thực hiện phân trang với skip và limit. * Sắp xếp theo trường được chỉ định. * Lấy danh sách users, không bao gồm trường password. * Tính thống kê cho từng user:   + Với buyer: totalOrders, totalSpent.   + Với seller: totalCarsPosted, totalSales. * Trả về danh sách users và thông tin phân trang.   **Đầu vào:**   * JWT token với quyền admin (bắt buộc) * Các query parameters (tùy chọn):   + page, limit   + search   + role, status   + sortBy, sortOrder   **Đầu ra:**   * 200 OK: Trả về { users[], pagination } * 403 Forbidden: Không có quyền admin * 500 Internal Server Error   **Tác động:** Chỉ đọc dữ liệu, không thay đổi gì trong hệ thống. |
|  |
| **Chức năng:** Admin xem chi tiết thông tin của một người dùng cụ thể.  **Cách hoạt động:**   * Tìm user theo ID, loại bỏ trường password. * Tính các thống kê chi tiết: totalOrders, totalSpent, totalCarsPosted, totalSales. * Lấy danh sách đơn hàng/cars gần đây (tối đa 5 bản ghi). * Populate các thông tin liên quan và trả về.   **Đầu vào:**   * Path param: id (MongoDB ObjectId) * JWT token với quyền admin (bắt buộc)   **Đầu ra:**   * 200 OK: Trả về user object + stats * 403 Forbidden: Không có quyền admin * 404 Not Found: Không tìm thấy user * 500 Internal Server Error   **Tác động:** Chỉ đọc thông tin người dùng, không thay đổi dữ liệu. |
|  |
| **Chức năng:** Admin xóa người dùng khỏi hệ thống.  **Cách hoạt động:**   * Tìm user theo ID. * Kiểm tra người dùng có đơn hàng đang hoạt động không (pending, confirmed, paid\_partial). * Nếu có đơn hàng đang hoạt động → từ chối xóa. * Nếu không → thực hiện xóa user khỏi database.   **Đầu vào:**   * Path param: id (MongoDB ObjectId) * JWT token với quyền admin (bắt buộc)   **Đầu ra:**   * 200 OK: Xóa thành công * 400 Bad Request: Có đơn hàng đang hoạt động * 403 Forbidden: Không có quyền admin * 404 Not Found: Không tìm thấy user * 500 Internal Server Error   **Tác động:** Xóa vĩnh viễn bản ghi người dùng khỏi hệ thống. |
|  |
| **Chức năng:** Admin cập nhật trạng thái của người dùng.  **Cách hoạt động:**   * Validate giá trị status (active, inactive, banned). * Cập nhật trường status và timestamp updatedAt. * Trả về đối tượng user đã được cập nhật.   **Đầu vào:**   * Path param: id (MongoDB ObjectId) * JWT token với quyền admin (bắt buộc) * Body:   {  "status": "active" | "inactive" | "banned"  }  **Đầu ra:**   * 200 OK: Trả về user đã được cập nhật * 400 Bad Request: Status không hợp lệ * 403 Forbidden: Không có quyền admin * 404 Not Found: Không tìm thấy user * 500 Internal Server Error   **Tác động:** Cập nhật trạng thái người dùng, ảnh hưởng đến khả năng truy cập của họ vào hệ thống. |
|  |
| **Chức năng:** Admin lấy danh sách xe trên hệ thống với filter và phân trang.  **Cách hoạt động:**   * Parse các query parameters: page, limit, search, status, sortBy, sortOrder. * Tạo bộ lọc từ search (tìm theo title, brand, model) và status. * Thực hiện phân trang với skip và limit. * Sắp xếp kết quả theo trường chỉ định. * Populate thông tin seller (user\_id). * Trả về danh sách cars[] với seller info và pagination.   **Đầu vào:**   * JWT token với quyền admin (bắt buộc) * Các query parameters (tùy chọn):   + page, limit   + search   + status   + sortBy, sortOrder   **Đầu ra:**   * 200 OK: { cars[], pagination } * 403 Forbidden: Không có quyền admin * 500 Internal Server Error   **Tác động:** Chỉ đọc dữ liệu, không thay đổi gì trong hệ thống. |
|  |
| **Chức năng:** Admin lấy danh sách yêu cầu đăng tin xe từ người bán, có hỗ trợ filter và phân trang.  **Cách hoạt động:**   * Parse các query parameters như: page, limit, search, status, sortBy, sortOrder. * Tạo bộ lọc từ search (theo name, brand, model) và status. * Thực hiện phân trang bằng skip và limit. * Sắp xếp theo trường chỉ định (mặc định createdAt desc). * Manually populate thông tin seller từ sellerId. * Populate danh sách inspectors từ mảng userIds.   **Đầu vào:**   * JWT token với quyền admin (bắt buộc) * Query parameters (tùy chọn):   + page, limit   + search   + status   + sortBy, sortOrder   **Đầu ra:**   * 200 OK: { requests[], pagination } * 403 Forbidden: Không có quyền admin * 500 Internal Server Error   **Tác động:** Chỉ đọc dữ liệu, không thay đổi hệ thống. |
|  |
| **Chức năng:** Admin duyệt một yêu cầu đăng tin và tự động tạo bài đăng xe.  **Cách hoạt động:**   * Cập nhật status = "checked", lưu approvedAt, approvalNotes nếu có. * Tạo bản ghi Car mới từ dữ liệu trong request:   + title = brand + model + year   + price = price\_recommend\_high   + status = "selling"   + sellerId = request.sellerId * Gán carId mới vào bản ghi request. * Populate thông tin seller và trả về kết quả.   **Đầu vào:**   * Path param: id (MongoDB ObjectId) * JWT token với quyền admin (bắt buộc) * Body:   {  "notes": "optional string"  }  **Đầu ra:**   * 200 OK: Duyệt thành công + request đã cập nhật với thông tin seller * 403 Forbidden: Không có quyền admin * 404 Not Found: Không tìm thấy request * 500 Internal Server Error: Lỗi tạo xe hoặc cập nhật request   **Tác động:** Cập nhật request status, tạo mới bản ghi xe trong hệ thống. |
|  |
| **Chức năng:** Admin từ chối một yêu cầu đăng tin với lý do cụ thể.  **Cách hoạt động:**   * Validate rằng field reason được cung cấp. * Cập nhật status = "rejected", rejectedAt, message = reason. * Populate thông tin seller và trả về request đã cập nhật.   **Đầu vào:**   * Path param: id (MongoDB ObjectId) * JWT token với quyền admin (bắt buộc) * Body:   {  "reason": "required string"  }  **Đầu ra:**   * 200 OK: Từ chối thành công + request đã cập nhật * 400 Bad Request: Thiếu lý do từ chối * 403 Forbidden: Không có quyền admin * 404 Not Found: Không tìm thấy request * 500 Internal Server Error   **Tác động:** Cập nhật trạng thái request, ghi lại lý do từ chối. |
|  |
| **Chức năng:** Admin lấy danh sách tất cả các inspector để phân công công việc.  **Cách hoạt động:**   * Query tất cả users có role là "inspector". * Chỉ lấy các trường cần thiết như: name, email, phone, createdAt. * Sắp xếp danh sách theo tên (name).   **Đầu vào:**   * JWT token với quyền admin (bắt buộc).   **Đầu ra:**   * 200 OK: Trả về mảng các inspector. * 403 Forbidden: Người dùng không có quyền admin. * 500 Internal Server Error: Lỗi server.   **Tác động:** Chỉ đọc dữ liệu, không thay đổi gì trong hệ thống. |
|  |
| **Chức năng:** Admin lấy danh sách đơn hàng với các tùy chọn lọc và phân trang.  **Cách hoạt động:**   * Parse query parameters: page, limit, search, status, sortBy, sortOrder. * Cho phép tìm kiếm theo mã đơn hàng (orderId) bằng regex. * Thực hiện phân trang và sắp xếp. * Manually populate thông tin car và buyer cho mỗi đơn hàng. * Enrich dữ liệu để trả về đầy đủ thông tin cho frontend.   **Đầu vào:**   * JWT token với quyền admin (bắt buộc) * Query parameters (tùy chọn):   + page, limit   + search (tìm theo mã đơn hàng)   + status   + sortBy, sortOrder   **Đầu ra:**   * 200 OK: { orders[], pagination } * 403 Forbidden: Không có quyền admin * 500 Internal Server Error   **Tác động:** Read-only, không thay đổi dữ liệu. |
|  |
| **Chức năng:** Admin cập nhật trạng thái đơn hàng.  **Cách hoạt động:**   * Validate giá trị status nằm trong danh sách hợp lệ: pending, confirmed, paid\_partial, paid\_full, completed, cancelled * Tạo updateData từ status và notes (nếu có). * Cập nhật order, trả về kết quả đã cập nhật.   **Đầu vào:**   * Path param: id (MongoDB ObjectId) * JWT token với quyền admin (bắt buộc) * Body:   {  "status": "required enum",  "notes": "optional string"  }  **Đầu ra:**   * 200 OK: Cập nhật thành công * 400 Bad Request: Status không hợp lệ * 403 Forbidden: Không có quyền admin * 404 Not Found: Không tìm thấy đơn hàng * 500 Internal Server Error   **Tác động:** Cập nhật order status, ảnh hưởng đến flow giao dịch. |
|  |
| **Chức năng:** Admin lấy danh sách thanh toán với các tùy chọn lọc và phân trang.  **Cách hoạt động:**   * Parse query: page, limit, search, status, sortBy, sortOrder. * Cho phép tìm theo nhiều trường: transactionId, buyerName, buyerEmail, carTitle. * Thực hiện phân trang và sắp xếp. * Populate thông tin user và order (với nested populate buyer, seller, car). * Làm đầy đủ dữ liệu còn thiếu để frontend hiển thị.   **Đầu vào:**   * JWT token với quyền admin (bắt buộc) * Query parameters (tùy chọn):   + page, limit   + search   + status   + sortBy, sortOrder   **Đầu ra:**   * 200 OK: { payments[], pagination } * 403 Forbidden: Không có quyền admin * 500 Internal Server Error   **Tác động:** Read-only, không thay đổi dữ liệu. |
|  |
| **Chức năng:** Admin lấy chi tiết một giao dịch thanh toán cụ thể.  **Cách hoạt động:**   * Tìm payment theo ID. * Populate user và order (nested populate buyer, seller, car). * Làm giàu dữ liệu cho frontend nếu cần. * Trả về payment đã được populate đầy đủ.   **Đầu vào:**   * Path param: id (MongoDB ObjectId) * JWT token với quyền admin (bắt buộc)   **Đầu ra:**   * 200 OK: Trả về payment object đã được enrich * 403 Forbidden: Không có quyền admin * 404 Not Found: Không tìm thấy thanh toán * 500 Internal Server Error   **Tác động:** Chỉ đọc dữ liệu, không thay đổi trạng thái. |
|  |
| **Chức năng:** Admin tạo mã QR chuyển khoản để chuyển tiền cho người bán.  **Cách hoạt động:**   * Tìm payment và populate order với thông tin xe. * Tìm thông tin ngân hàng của seller. * Tạo dữ liệu QR VietQR với:   + accountNo, accountName, bankCode, amount, transferDescription * Generate QR URL và nội dung chuyển khoản * Trả về dữ liệu QR   **Đầu vào:**   * Path param: paymentId (MongoDB ObjectId) * JWT token với quyền admin (bắt buộc)   **Đầu ra:**   * 200 OK: { payment, seller, qrData, qrUrl, transferContent } * 400 Bad Request: Seller chưa có thông tin ngân hàng * 403 Forbidden: Không có quyền admin * 404 Not Found: Không tìm thấy payment/order/seller * 500 Internal Server Error: Lỗi tạo QR   **Tác động:** Read-only, hỗ trợ quá trình chuyển khoản thủ công. |

**CAR:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | |  |  * Tìm xe theo các tiêu chí trên thanh query * Tìm tất cả xe có deleted = false, status = “selling” * Nếu có các tiêu chí { pricemin, pricemax, yearmin, yearmax, kmmin, kmmax, fuel\_type, seat\_capacity, keyword } thì thêm tiêu chí đó vô bộ tìm kiếm * Sau khi tìm kiếm thì dựa vào số trang đang ở thì lọc nó theo đúng số trang nó đang ở (số xe trên trang : 6) * Trả về dữ liệu  |  | | --- | | {  Num : số lượng xe thỏa mãn yêu cầu,  minPrice : giá thấp nhất của tất cả xe,  maxPrice : giá cao nhất của tất cả xe,  cars : Một mảng gồm nhiều xe  } |   Lỗi trong quá trình chạy   |  |  | | --- | --- | | Status = 500 | Server Error! | |
| |  | | --- | |  |   Lấy slug từ thanh params   * Không có slug  |  |  | | --- | --- | | Status = 404 | Message : “Không có slug car” |   Tìm kiếm xe có deleted = false, status = “selling”, slug : slug từ params  Nếu đang là người bán và chủ sỡ hữu xe đó thì thêm vào tiêu chí tìm kiếm $or, deleted : false, slug : slug của xe, sellerId : Id người bán  Tìm xe   * Không tìm thấy dữ liệu xe  |  |  | | --- | --- | | Status = 404 | Message : “Car not found!” |  * Gọi view model thêm vô để làm thống kê   Trả về dữ liệu   |  | | --- | | {      "location": Địa điểm bán xe      "fuel\_use": Nhiên liệu xe      "sellerBankInfo": Thông tin ngân hàng      "title": Tên xe,      "category\_id": id của hãng xe,      "price": giá,      "year": năm mua,      "km": số km đã đi,      "fuel": nhiên liệu sử dụng,      "seat\_capacity": số ghế ngồi,      "comment": Mô tả của người bán,      "img\_src": Một mảng các ảnh xe,      "slug": slug của xe      "sellerId": id của người bán,      "status": trạng thái xe,      "time\_sold": thời gian bán xe (nếu đã bán),      "createdAt": Thời gian tạo,      "user": Thông tin người bán  } |   Lỗi trong quá trình chạy   |  |  | | --- | --- | | Status = 500 | Server Error! | |
| |  | | --- | |  |   Lấy id người dùng qua giải mã middleware jwt   * Không tìm thấy người dùng  |  |  | | --- | --- | | Status = 404 | Message : “Không tìm thấy thông tin người dùng” |   Kiểm tra thông tin ngân hàng   * Không có  |  |  | | --- | --- | | Status = 400 | message:            "Bạn cần cập nhật và xác minh thông tin ngân hàng trước khi đăng xe" |   Tạo xe và lưu lại trong data base   * Thành công  |  | | --- | | message: "Create Successfully" |   Lỗi trong quá trình chạy   |  |  | | --- | --- | | Status = 500 | message: "Server Error!!!" | |
| |  | | --- | |  |   Lấy userId từ middleware  Lấy thông tin người dùng   * Không tìm thấy thông tin người dùng * Người dùng không có role là người bán   Tìm kiếm xe theo sellerId : id của người bán  Trả về kết quả :  Một mảng các xe   |  | | --- | | {  Title : hãng xe,  Price : giá,  Img\_src : ảnh đại diện xe,  Status : tình trạng xe,  Slug : slug của xe  } |   Lỗi trong quá trình chạy   |  |  | | --- | --- | | Status = 500 | message: "Server Error!!!" | |
| |  | | --- | |  |   Lấy slug từ params   * Không có  |  |  | | --- | --- | | Status = 400 | Message : “Không có slug” |   Lấy thông tin người dùng từ userId   * Không có  |  |  | | --- | --- | | Status = 404 | Message : “Không tìm thấy người dùng” |   Tìm xem car có slug giống đã đưa không, deleted = false và thuộc sỡ hữu người dùng   * Không có  |  |  | | --- | --- | | Status = 404 | Message : “Không tìm thấy xe!” |   Kiểm tra không có thay đổi | sai định dạng   |  |  | | --- | --- | | Status = 400 | Message : Không có thay đổi gì | giá bán phải là một số |   Nếu xe đã trong tình trạng đã bán (sold)   * Không thể chỉnh sửa, trả về  |  |  | | --- | --- | | Status = 403 | Message : Xe đã bán không thể chỉnh sửa |   Cập nhật thông tin xe   |  |  | | --- | --- | | Status = 200 | Message : Đã chỉnh sửa thông tin thành công! |   Lỗi trong quá trình chạy   |  |  | | --- | --- | | Status = 500 | message: "Server Error!!!" | |

**CATEGORY:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | |  |   Lấy tất cả danh mục và trả chúng về dạng cây   |  | | --- | | [  {  Thông tin category cha,  Children : Một mảng thông tin category con dựa trên parentId, có index đánh thứ tự  },  {}…  ] |   Lỗi trong quá trình chạy   |  |  | | --- | --- | | Status = 500 | message: "Server Error!!!" | |
| |  | | --- | |  |   Tạo danh mục mới bằng những gì trong req.body   * Thành công  |  |  | | --- | --- | | Stautus = 200 | message: "Create Successfully!" | |

**REQUEST:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  |   Lấy userId từ middleware  Lấy thông tin người dùng bằng userId   * Không có  |  |  | | --- | --- | | Status = 404 | Message : Không tìm thấy người dùng |  * Người dùng không có role seller  |  |  | | --- | --- | | Status = 403 | Message : Bạn không có quyền tạo yêu cầu |   Kiểm tra thông tin ngân hàng  Nếu thiếu thông tin   |  |  | | --- | --- | | Status = 400 | message:          "Bạn cần cập nhật thông tin ngân hàng trước khi đăng xe để có thể nhận thanh toán từ người mua”,  needBankInfo: true |   Lấy thông tin từ req.body  Tạo mới thông tin lưu vô trong database  Thành công   * “Tạo thành công yêu cầu!”   Lỗi trong quá trình tạo   |  |  | | --- | --- | | Status = 500 | Server Error | |
| |  | | --- | |  |   Lấy userId từ middleware  Lấy thông tin người dùng bằng userId   * Không có  |  |  | | --- | --- | | Status = 404 | Message : Không tìm thấy người dùng |  * Người dùng không có role seller  |  |  | | --- | --- | | Status = 403 | Message : Bạn không có quyền tạo yêu cầu |   Lấy request từ sellerId = userId  Trả thông tin một mảng các yêu cầu có id đươc mã hóa bởi md5   |  | | --- | | [  {  Id : id được mã hóa bởi md5  Name : tên của yêu cầu,  Status : trạng thái của yêu cầu (đang chờ - pending, checked – đã kiểm tra, done – đã hoàn thành, reject – đã từ chối),  Img\_demo : ảnh đại diện của yêu cầu,  createdAt : ngày tạo yêu cầu,  slug : slug của yêu cầu  }  ] |   Lỗi trong quá trình tạo   |  |  | | --- | --- | | Status = 500 | Server Error | |
| |  | | --- | |  |   Lấy userId từ middleware  Lấy thông tin người dùng bằng userId   * Không có  |  |  | | --- | --- | | Status = 404 | Message : Không tìm thấy người dùng |  * Người dùng không có role staff  |  |  | | --- | --- | | Status = 400 | Message : Không phải là nhân viên,  Success : false |   Lấy danh sách các yêu cầu, trong đó userId có trong userIds trong những yêu cầu đó (được phân công)   * Trả về kết quả (dù không hay có)  |  |  | | --- | --- | | Status = 200 | Success : true,  Data : mảng các request |   Lỗi trong quá trình tạo   |  |  | | --- | --- | | Status = 500 | Message : Server Error,  Success : false | |
| |  | | --- | |  |   Lấy userId bằng middleware  Lấy slugRequest từ params  Nếu không có slugRequest   |  |  | | --- | --- | | Status = 400 | Message : Không có slug request |   Tìm thông tin user   * Không thấy  |  |  | | --- | --- | | Status = 404 | Message : Không tìm thấy người dùng |  * Người dùng có role = user  |  |  | | --- | --- | | Status = 403 | Message : Người dùng không có quyền truy cập |   Tạo yêu cầu tìm kiếm  Nếu người dùng có role là seller : chỉ chủ sỡ hữu mới truy cập được  Nếu người dùng có role là staff : chỉ những nhân viên được phái đến mới truy cập được  Yêu cầu truy cập có slug phải là slugRequest   * Nếu không tìm thấy  |  |  | | --- | --- | | Status = 404 | Message : Không tìm thấy yêu cầu |   Nếu tìm thấy   |  | | --- | | {  Thông tin của yêu cầu : tên, năm mua, km đã đi, nhiên liệu, địa điểm ,số chỗ ngồi, ảnh đại diện, tình trạng, ảnh đã chụp bởi nhân viên (nếu đã kiểm tra),  Thông tin của nhân viên : một mảng gồm thông tin của nhân viên kiểm tra : [  Tên nhân viên,  Email nhân viên,  Số điện thoại nhân viên,  Avatar nhân viên  ]  } |   Lỗi trong quá trình lấy thông tin   |  |  | | --- | --- | | Status = 500 | Message : Server Error | |
| |  | | --- | |  |   Chuyển yêu cầu thành checked(bởi nhân viên kiểm tra)  Lấy userId từ middleware  Tìm kiếm thông tin người dùng   * Không có  |  |  | | --- | --- | | Status = 404 | Message : Không tìm thấy người dùng |   Lấy slug từ params   * Không có  |  |  | | --- | --- | | Status = 400 | message: "Chưa có thông tin form request! " |   Lấy request từ slug và userIds của request có chứa slug   * Không có  |  |  | | --- | --- | | Status = 404 | message: "Không tìm thấy request yêu cầu!" |   Tìm kiếm thông tin người bán   * Không có  |  |  | | --- | --- | | Status = 404 | message: "Không tìm thấy người bán !" |  * Nếu đơn không phải trong tình trạng “pending”  |  |  | | --- | --- | | Status = 403 | message: "Đã duyệt rồi không thể duyệt nữa!" |   Xác thực tài khoản ngân hàng   * Lưu lại là đã xác thực tài khoản ngân hàng   Tạo file pdf xác thực yêu cầu kiểm tra  Tạo thông báo đã duyệt  Hoàn tất   |  |  | | --- | --- | | Status = 200 | message: Nội dung trả về,  bankVerified: Đã xác thực tài khoản ngân hàng chưa,  hasBankInfo: Có thông tin ngân hàng chưa |   Lỗi trong quá trình lấy thông tin   |  |  | | --- | --- | | Status = 500 | Message : Server Error | |
| |  | | --- | |  |   Lấy userId từ middleware và tìm kiếm thông tin người dùng   * Không có  |  |  | | --- | --- | | Status = 404 | Message : Không tìm thấy người dùng |   Lấy slug từ params   * Không có  |  |  | | --- | --- | | Status = 400 | Message : “chưa có thông tin form request” |   Lấy thông tin yêu cầu từ slug và userId của nhân viên (được phái đến kiểm tra)   * Không có  |  |  | | --- | --- | | Status = 404 | message: "Không tìm thấy request yêu cầu!" |   Lấy thông tin người bán   * Không tìm thấy  |  |  | | --- | --- | | Status = 404 | message: "Không tìm thấy người bán !" |   Yêu cầu khác trạng thái “pending”   |  |  | | --- | --- | | Status = 403 | message: "Đã duyệt rồi không thể duyệt nữa!" |  * Tạo pdf với nội dung kiểm tra   Tạo thông báo đã từ chối   * Hoàn thành : "Đã chuyển trạng thái sang từ chối xong thành công!”   Lỗi trong quá trình lấy thông tin   |  |  | | --- | --- | | Status = 500 | Message : Server Error | |
| |  | | --- | |  |   Lấy userId từ middleware  Tìm thông tin người dùng   * Không có  |  |  | | --- | --- | | Status = 404 | Message : Không tìm thấy người dùng |  * Người dùng không phải người bán ( role != “seller”)  |  |  | | --- | --- | | Status = 403 | Message : Người dùng không có quyền đăng bán |  * Kiểm tra thấy chưa xác thực tài khoản ngân hàng  |  |  | | --- | --- | | Status = 400 | message : "Người dùng chưa xác thực thông tin tài khoản ngân hàng!" |   Lấy slug từ param  Tìm yêu cầu từ : phải là của chủ sỡ hữu, trạng thái đã kiểm tra rồi (checked) và có slug đúng với slug đã đưa   * Không tìm thấy  |  |  | | --- | --- | | Status = 404 | message: "Không tìm thấy yêu cầu !" |   Tìm thấy rồi thì tạo bên model xe(car) với comment và price lấy từ req.body (quá trình đăng bán)  Chuyển trạng thái yêu cầu thành done   * Hoàn tất trả về : “Đã đăng bán thành công”   Lỗi trong quá trình lấy thông tin   |  |  | | --- | --- | | Status = 500 | Message : Server Error | |
| |  | | --- | |  |   Lấy danh sách id người dùng (userIds) từ req.body  Lấy slug từ slugRequest  Tìm kiếm yêu cầu theo tiêu chí slug   * Không có  |  |  | | --- | --- | | Status = 404 | message: "Không tìm thấy yêu cầu ! " |   Lọc mảng userIds lấy những cái hợp lệ  Push những id hợp lệ vào sau danh sách userIds và lưu lại  Tạo thông báo tới những người được phân công   * Hoàn tất : “Đã phái nhân viên tới thành công”   Lỗi trong quá trình lấy thông tin   |  |  | | --- | --- | | Status = 500 | Message : Server Error | |

**THREAD:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | |  |   Lấy userId từ middleware  Tìm kiếm thông tin người dùng   * Không có  |  |  | | --- | --- | | Status = 404 | Message : Không tìm thấy người dùng |   Tìm danh sách bằng user1Id = userId or userId2 = userId  Format lại đoạn chat (loop qua từng danh sách):   * Lấy thông tin người chat 1 * Lấy thông tin người chat 2 * Lấy thông tin chat từng nội dung có id chứa trong msgIds * Lấy lastMessage đoạn chat mới nhất   Trả về dữ liệu đã format ở trên gồm một mảng các chủ đề đã loop và định dạng  Lỗi trong quá trình lấy thông tin   |  |  | | --- | --- | | Status = 500 | Message : Server Error | |
| |  | | --- | |  |   Kết nối giữa người mua với người bán thông qua các bài đăng xe  Tìm thông tin người mua = userId lấy từ middleware   * Không có  |  |  | | --- | --- | | Status = 404 | message: "Không tìm thấy người dùng! |   Tìm thông tin người bán từ sellerId lấy từ req.body   * Không có  |  |  | | --- | --- | | Status = 404 | message: "Người dùng đã bị cấm hoặc xóa tài khoản !" |  * Người bán là bản thân mình  |  |  | | --- | --- | | Status = 401 | message: "Không thể chat với chính bản thân mình!" |   Lấy thread giữa 2 người họ  Nếu không có thread => Tạo mới thread và thêm tin nhắn bắt đầu chat  Tạo contact bên statistic để thống kê  Trả về Hoàn tất : “Thread ready!”  Lỗi trong quá trình lấy thông tin   |  |  | | --- | --- | | Status = 500 | Message : Server Error | |

**STATISTIC:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  |   Lấy userId từ middleware  Lấy thông tin người dùng   * Không có  |  |  | | --- | --- | | Status = 404 | Message : Không tìm thấy người dùng |  * Không phải người bán  |  |  | | --- | --- | | Status = 403 | message : "Bạn không phải là người bán! " |   Tìm kiếm model views thông qua sellerId = userId  Tạo một chuỗi các ngày từ ngày tạo tài khoản -> hiện tại  Trả về lượt xem   |  | | --- | | date : một mảng các ngày = [‘16/06/2025’, ‘17/06/2025’, ‘18/06/2025’,…],  views : một mảng, các ô trong mảng tương ứng với số lượt truy cập xem ở trên = [1, 2, 1, 20, 25…] |   Lỗi trong quá trình lấy thông tin   |  |  | | --- | --- | | Status = 500 | Message : Server Error | |
| |  | | --- | |  |   Lấy userId từ middleware  Lấy thông tin người dùng   * Không có  |  |  | | --- | --- | | Status = 404 | Message : Không tìm thấy người dùng |  * Không phải người bán  |  |  | | --- | --- | | Status = 403 | message : "Bạn không phải là người bán! " |   Tìm kiếm model contacts thông qua sellerId = userId  Tạo một chuỗi các ngày từ ngày tạo tài khoản -> hiện tại  Trả về lượt tương tác   |  | | --- | | date : một mảng các ngày = [‘16/06/2025’, ‘17/06/2025’, ‘18/06/2025’,…],  contacts : một mảng, các ô trong mảng tương ứng với số lượt kết nối chat với người bán theo từng ngày ở trên = [1, 2, 1, 20, 25…] |   Lỗi trong quá trình lấy thông tin   |  |  | | --- | --- | | Status = 500 | Message : Server Error | |
| |  | | --- | |  |   Lấy userId từ middleware  Lấy thông tin người dùng   * Không có  |  |  | | --- | --- | | Status = 404 | Message : Không tìm thấy người dùng |  * Không phải người bán  |  |  | | --- | --- | | Status = 403 | message : "Bạn không phải là người bán! " |   Lấy danh sách các xe thuộc chủ sỡ hữu trong tình trạng đã bán  Lập danh sách các ngày từ ngày tạo tài khoản -> hiện tại  Duyệt qua những xe đã bán đó, dùng map nếu trùng ngày thì cộng thành một mảng tương ứng cùng ngày (cộng doanh thu theo ngày đó)  Trả về doanh thu   |  | | --- | | date : một mảng các ngày = [‘16/06/2025’, ‘17/06/2025’, ‘18/06/2025’,…],  revenues: một mảng, các ô trong mảng tương ứng với doanh thu của người bán theo từng ngày ở trên = [10000, 200000, 0, 0…] |   Lỗi trong quá trình lấy thông tin   |  |  | | --- | --- | | Status = 500 | Message : Server Error | |

# Chương 3 : Thiết kế hệ thống

## 3.1. Thiết kế cơ sở dữ liệu :

|  |
| --- |
|  |

Hệ thống sử dụng cơ sở dữ liệu **phi quan hệ (NoSQL)** – MongoDB, với cấu trúc dữ liệu được thiết kế để tối ưu cho các nghiệp vụ chính như: đăng bán xe, quản lý người dùng, nhắn tin, và xử lý yêu cầu định giá xe. Dưới đây là mô tả các bảng (collection) chính:

**1. Car Model**

Mô tả: Quản lý thông tin xe được rao bán, bao gồm tiêu đề, thương hiệu, giá, năm sản xuất, số km, loại nhiên liệu, dung lượng ghế, thông tin ngân hàng của người bán, trạng thái và thời gian bán.

Chức năng chính: Lưu trữ chi tiết xe, tự động tạo slug từ tiêu đề, liên kết với người bán thông qua sellerId.

**2. Category Model**

Mô tả: Quản lý danh mục xe, bao gồm tiêu đề, danh mục cha, mô tả, hình ảnh và trạng thái.

Chức năng chính: Phân loại xe theo danh mục, hỗ trợ tạo slug tự động và quản lý trạng thái danh mục.

**3. Chat Model**

Mô tả: Quản lý tin nhắn giữa người dùng, bao gồm thông tin người gửi, nội dung tin nhắn, trạng thái đã đọc và thời gian gửi.

Chức năng chính: Hỗ trợ giao tiếp trực tiếp giữa người mua và người bán.

**4. Log Model**

Mô tả: Lưu trữ nhật ký các sự kiện trong hệ thống như thay đổi trạng thái đơn hàng, thanh toán, gửi thông báo, v.v.

Chức năng chính: Ghi lại lịch sử hành động để theo dõi và kiểm tra, với các trường như loại sự kiện, dữ liệu trước/sau thay đổi và metadata.

**5. Notification Model**

Mô tả: Quản lý thông báo gửi đến người dùng, bao gồm loại thông báo, tiêu đề, nội dung, độ ưu tiên, trạng thái đã đọc và thời gian hết hạn.

Chức năng chính: Gửi và quản lý thông báo hệ thống, hỗ trợ các phương thức như đánh dấu đã đọc, đếm thông báo chưa đọc và dọn dẹp thông báo hết hạn.

**6. NotiUser Model**

Mô tả: Quản lý thông báo đơn giản dành cho người dùng, bao gồm nội dung thông báo và ID người nhận.

Chức năng chính: Lưu trữ thông báo cá nhân hóa cho người dùng.

**7. Order Model**

Mô tả: Quản lý đơn hàng, bao gồm mã đơn hàng, thông tin người mua, người bán, xe, trạng thái đơn hàng, phương thức thanh toán, thông tin giao xe và hợp đồng.

Chức năng chính: Tự động tạo mã đơn hàng, theo dõi trạng thái giao dịch và quản lý thông tin xác nhận từ người mua/người bán.

**8. OTP Model**

Mô tả: Quản lý mã OTP dùng để xác thực người dùng, bao gồm ID người dùng, mã OTP, thời gian tạo và thời gian hết hạn.

Chức năng chính: Hỗ trợ xác thực bảo mật trong quá trình đăng nhập hoặc giao dịch.

**9. Payment Model**

Mô tả: Quản lý thông tin thanh toán, bao gồm mã thanh toán, đơn hàng liên quan, số tiền, loại thanh toán, trạng thái và thông tin giao dịch ngân hàng.

Chức năng chính: Tự động tạo mã thanh toán, theo dõi trạng thái thanh toán và lưu trữ thông tin xác nhận từ admin.

**10. RequestAdd Model**

Mô tả: Quản lý yêu cầu thêm xe mới, bao gồm thông tin xe, trạng thái yêu cầu, hình ảnh và thông tin kiểm tra tính chính xác của dữ liệu.

Chức năng chính: Hỗ trợ người bán gửi yêu cầu thêm xe, với các trường kiểm tra dữ liệu và tạo slug tự động.

## 3.2. Thiết kế kiến trúc hệ thống :

Mô tả khái quát hệ thống

|  |
| --- |
|  |

Hệ thống website cho phép người dùng đăng bán và tìm mua xe được thiết kế theo mô hình **Client - Server**, sử dụng các công nghệ hiện đại như **ReactJS**, **Node.js**, **Express** và **MongoDB**. Giao tiếp giữa các thành phần diễn ra thông qua giao thức **HTTP**, dữ liệu được trao đổi theo định dạng **JSON** thông qua các API RESTful.

**Các thành phần chính:**

* **Trình duyệt (Client - ReactJS):**  
  Giao diện người dùng, gửi yêu cầu và nhận dữ liệu thông qua Axios. Người dùng có thể đăng nhập, xem xe, trò chuyện...
* **Backend (Server - ExpressJS):**  
  Xử lý các yêu cầu từ client, xác thực người dùng bằng JWT, thao tác với cơ sở dữ liệu và phản hồi dữ liệu.
* **MongoDB (CSDL NoSQL):**  
  Lưu trữ toàn bộ thông tin: người dùng, xe, tin nhắn, yêu cầu định giá... với tính mở rộng cao, phù hợp dữ liệu dạng linh hoạt.

|  |
| --- |
|  |

Hệ thống trang web sử dụng **WebSocket** (thông qua thư viện **socket.io**) để xây dựng chức năng **nhắn tin thời gian thực** giữa người mua và người bán. WebSocket cho phép **giao tiếp hai chiều** giữa trình duyệt và máy chủ mà không cần liên tục gửi yêu cầu HTTP như cách truyền thống.

Khi người dùng mở trang trò chuyện, một kết nối WebSocket được thiết lập giữa **client (trình duyệt)** và **server**. Sau đó, các tin nhắn được gửi và nhận **ngay lập tức**, tạo cảm giác giống như trò chuyện thật sự.

Việc sử dụng WebSocket giúp:

* Giảm độ trễ khi gửi/nhận tin nhắn.
* Cập nhật trạng thái người dùng như “đang gõ”, “đã xem”, “đang online”.
* Tiết kiệm tài nguyên hệ thống so với phương pháp long-polling.

Tính năng này mang lại trải nghiệm tương tác tốt hơn cho người dùng và là một phần quan trọng trong hệ thống giao tiếp trực tiếp trên trang web.

|  |
| --- |
|  |
| Sơ đồ kiến trúc triển khai hệ thống web |

Sơ đồ dưới đây mô tả kiến trúc tổng thể của hệ thống web, trong đó người dùng truy cập website thông qua tên miền. Yêu cầu được chuyển đến máy chủ VPS thông qua DNS. Tại VPS, Nginx đóng vai trò reverse proxy, phân luồng request tới các dịch vụ nội bộ như frontend (giao diện người dùng), backend (xử lý logic nghiệp vụ), WebSocket và các API phụ trợ khác.

|  |
| --- |
|  |
| Sơ đồ tự động triển khai với github action |

Sơ đồ thể hiện quy trình triển khai mã nguồn tự động sau mỗi lần cập nhật (push) lên thực hiện pull request. Thông qua GitHub Actions, hệ thống sẽ tự động SSH vào VPS, pull mã nguồn mới về, cài đặt các gói cần thiết (nếu có) và khởi động lại dịch vụ web, đảm bảo hệ thống luôn ở trạng thái mới nhất mà không cần thao tác thủ công.

## 3.3. Các sơ đồ khác

|  |
| --- |
|  |
| Sơ đồ hoạt động người dùng |

|  |
| --- |
|  |
| Sơ đồ hoạt động người bán |

|  |
| --- |
|  |
| Sơ đồ hoạt động nhân viên |

# Chương 4 : Cài đặt – triển khai

## 4.1. Môi trường phát triển :

Hệ thống được phát triển trên nền tảng công nghệ hiện đại, đảm bảo dễ mở rộng và triển khai. Cụ thể:

* **Hệ điều hành phát triển**: Windows 10
* **Trình soạn thảo mã nguồn**: Visual Studio Code
* **Trình duyệt kiểm thử**: Google Chrome, Cốc cốc, Microsoft Edge
* **Ngôn ngữ lập trình**:
  + Frontend: JavaScript (React.js)
  + Backend: JavaScript (Node.js + Express)
* **Cơ sở dữ liệu**: MongoDB
* **Hệ thống quản lý mã nguồn**: Git + GitHub
* **Công cụ triển khai**: VPS Ubuntu 22.04, Nginx, GitHub Actions, PM2

## 4.2. Cài đặt Frontend :

**Công nghệ sử dụng:**

* **React.js**: Xây dựng giao diện người dùng động và hiện đại.
* **Axios**: Giao tiếp với backend qua REST API.
* **React Router**: Điều hướng trang.
* **React Redux**: quản lý trạng thái tốt hơn

Các bước cài đặt:

|  |
| --- |
| cd frontend  npm install  npm run build  pm2 start ecosystem.config.cjs |

Router của client (code lại thì copy paste vào lại):

|  |
| --- |
| import Default from "../../Layout/Default";  import BanXe from "../../pages/BanXe";  import ChiTietXe from "../../pages/ChiTietXe";  import GioiThieu from "../../pages/GioiThieu";  import Home from "../../pages/Home";  import ClientAuth from "../../pages/Auth/client";  import Message from "../../pages/Message";  import ClientRegister from "../../pages/Auth/client/ClientRegister";  import ForgotPassword from "../../pages/Auth/client/ForgotPassword";  import ChangeEmail from "../../pages/Auth/client/ChangeEmail";  import ChangePassword from "../../pages/Auth/client/ChangePassword";  import SetupPassword from "../../pages/Auth/client/SetupPassword";  import OauthFillInfo from "../../pages/Auth/client/OauthFillInfo";  import MyAccount from "../../pages/MyAccount";  import DashBoard from "../../pages/MyAccount/child/DashBoard";  import ManageCar from "../../pages/MyAccount/child/ManageCar";  import Sales from "../../pages/MyAccount/child/Sales";  import Notify from "../../pages/MyAccount/child/Notify";  import Insight from "../../pages/MyAccount/child/Insight";  import History from "../../pages/MyAccount/child/History";  import Faq from "../../pages/MyAccount/child/Faq";  import MaintenancePage from "../../pages/Maintenance";  import RequestForm from "../../pages/RequestForm/child/RequestForm";  import RequestDetail from "../../pages/RequestForm";  import RequestInfo from "../../pages/RequestForm/child/RequestInfo";  import RequestVerify from "../../pages/RequestForm/child/RequestVerify";  import RequestDone from "../../pages/RequestForm/child/RequestDone";  import Error404 from "../../pages/Error404";  import EditCar from "../../pages/MyAccount/child/EditCar";  // Footer Pages  import DieuKhoanSuDung from "../../pages/FooterPages/DieuKhoanSuDung";  import ChinhSachQuyenRiengTu from "../../pages/FooterPages/ChinhSachQuyenRiengTu";  import CauHoiThuongGap from "../../pages/FooterPages/CauHoiThuongGap";  import LienHe from "../../pages/FooterPages/LienHe";  import CoHoiViecLam from "../../pages/FooterPages/CoHoiViecLam";  import SellerInfo from "../../pages/SellerInfo";  export const routes = [    {      path: "/maintenance",      element: <MaintenancePage />,    },    {      path: "/",      element: <Default />,      children: [        {          index: true,          element: <Home />,        },        {          path: "ban-xe",          element: <BanXe />,        },        {          path: "gioi-thieu",          element: <GioiThieu />,        },        {          path: "login",          element: <ClientAuth />,        },        {          path: "register",          element: <ClientRegister />,        },        {          path: "forgot-password",          element: <ForgotPassword />,        },        {          path: "change-email",          element: <ChangeEmail />,        },        {          path: "change-password",          element: <ChangePassword />,        },        {          path: "setup-password",          element: <SetupPassword />,        },        {          path: "chat",          element: <Message />,        },        {          path: "fill-info",          element: <OauthFillInfo />,        },        {          path: "my\_account",          element: <MyAccount />,          children: [            {              index: true,              element: <DashBoard />,            },            {              path: "manage-car",              element: <ManageCar />,            },            {              path: "statistics/sales",              element: <Sales />,            },            {              path: "statistics/insights",              element: <Insight />,            },            {              path: "inspection-history",              element: <History />,            },            {              path: "notifications",              element: <Notify />,            },            {              path: "faq",              element: <Faq />,            },            {              path: "edit-car/:slugCar",              element: <EditCar />,            },          ],        },        {          path: "request-form",          element: <RequestForm />,        },        {          path: "request-detail/:slugRequest",          element: <RequestDetail />,          children: [            {              path: "info",              element: <RequestInfo />,            },            {              path: "verify",              element: <RequestVerify />,            },            {              path: "done",              element: <RequestDone />,            },            {              index: true,              path: "\*",              element: <Error404 />,            },          ],        },        {          path: "chi-tiet-xe/:slugCar",          element: <ChiTietXe />,        },        {          path: "dieu-khoan-su-dung",          element: <DieuKhoanSuDung />,        },        {          path: "chinh-sach-quyen-rieng-tu",          element: <ChinhSachQuyenRiengTu />,        },        {          path: "cau-hoi-thuong-gap",          element: <CauHoiThuongGap />,        },        {          path: "lien-he",          element: <LienHe />,        },        {          path: "co-hoi-viec-lam",          element: <CoHoiViecLam />,        },        {          path: "nguoi-ban/:slugSeller",          element: <SellerInfo/>        },        {          path: "\*",          element: <Error404 />,        },      ],    },  ]; |

Router admin (code lại thì copy paste lại)

|  |
| --- |
| import Dashboard from "../../pages/Admin/dashboard";  import AdminDefault from "../../Layout/AdminDefault";  import Error404 from "../../pages/Error404";  export const routesAdmin = [    {      path: "/admin",      element: <AdminDefault />,      children: [        {          path: "dashboard",          element: <Dashboard />,        },        {          path: "\*",          element: <Error404 />,        },      ],    },  ]; |

## 4.3. Cài đặt Backend :

**Công nghệ sử dụng:**

* **Node.js + Express**: Tạo API xử lý nghiệp vụ.
* **Socket.io**: Giao tiếp thời gian thực, hỗ trợ chức năng chat giữa người dùng.
* **Mongoose**: ORM kết nối MongoDB.
* **PM2** *(trong triển khai)*: Quản lý tiến trình backend.
* **JWT**: Xác thực người dùng.
* **CORS + cookie-parser**: Hỗ trợ xác thực và bảo mật.

**Các bước cài đặt:**

|  |
| --- |
| cd backend  npm install  pm2 start ecosystem.config.js |

* Môi trường được quản lý qua biến .env
* Có tích hợp CORS, dotenv, bcrypt, jsonwebtoken...

## 4.4. Kết nối và xử lý dữ liệu :

Hệ thống sử dụng cơ sở dữ liệu **MongoDB** để lưu trữ và quản lý thông tin người dùng, bài đăng, tin nhắn và các dữ liệu liên quan. Việc kết nối giữa **backend** và cơ sở dữ liệu được thực hiện thông qua thư viện **Mongoose**, giúp mô hình hóa dữ liệu dưới dạng các schema rõ ràng và dễ kiểm soát.

### 4.4.1. Kết nối cơ sở dữ liệu

Mỗi khi backend khởi động, hệ thống sẽ thực hiện kết nối đến MongoDB thông qua URI được cấu hình trong tệp .env.

|  |
| --- |
| MONGO\_URI=mongodb+srv://nhattienlam2005:0TxCQlToB7J9NIPE@cluster0.4rt7yxj.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0 |

Kết nối backend :

|  |
| --- |
| const connectDB = async () => {    try {      await mongoose.connect(process.env.MONGO\_URI);      console.log("MongoDB connected");    } catch (err) {      console.error("Error connecting to MongoDB:", err.message);      process.exit(1);    }  }; |

## 4.5. Đóng gói và triển khai :

### 4.5.1. Mô hình triển khai

Hệ thống được triển khai trên một máy chủ ảo riêng (VPS) chạy hệ điều hành **Ubuntu 22.04**. Kiến trúc triển khai bao gồm:

* **Frontend**: Được build sẵn (React) và phục vụ qua **Nginx**.
* **Backend**: Node.js chạy với **PM2**, hỗ trợ WebSocket.
* **Reverse Proxy**: **Nginx** điều hướng các request HTTP tới đúng dịch vụ.
* **CI/CD**: Tự động triển khai thông qua **GitHub Actions** mỗi khi cập nhật code.

### ****4.5.2. Đóng gói hệ thống****

**a) Frontend (React + Vite)**

Sau khi hoàn tất lập trình, frontend được đóng gói bằng lệnh

|  |
| --- |
| cd frontend  npm install  npm run build |

Kết quả build nằm trong thư mục dist/.

**b) Backend (Node.js)**

* Mã nguồn backend không cần build. Sau khi copy lên VPS, chỉ cần cài đặt và khởi chạy:

|  |
| --- |
| * cd backend * npm install * pm2 start ecosystem.config.js |

Các biến môi trường như cổng chạy, MongoDB URI, URL frontend,... được cấu hình trong file .env.

### 4.5.3. Cấu hình Reverse Proxy bằng Nginx

Để hệ thống hoạt động ổn định và phân tách rõ frontend - backend, Nginx được sử dụng làm **reverse proxy** kết hợp với **HTTPS** thông qua **Let's Encrypt (Certbot)**.

**Cấu hình hệ thống tên miền**

Hệ thống sử dụng 2 tên miền riêng biệt:

|  |  |  |  |
| --- | --- | --- | --- |
| Thành phần | Tên miền | Cổng nội bộ | Mục đích |
| Frontend | Fakeauto.id.vn | 4173 | Phục vụ giao diện React |
| Backend | Api.fakeauto.id.vn | 3000 | Xử lý API và Websocket |

Cấu trúc thư mục cấu hình Nginx

* Cấu hình chính : /etc/nginx/nginx.conf
* Cấu hình virtual host:
  + /etc/nginx/sites-available/fakeauto.id.vn
  + /etc/nginx/sites-available/api.fakeauto.id.vn
* Kích hoạt cấu hình thông qua symbolic link:

|  |
| --- |
| sudo ln -s /etc/nginx/sites-available/fakeauto.id.vn /etc/nginx/sites-enabled/  sudo ln -s /etc/nginx/sites-available/api.fakeauto.id.vn /etc/nginx/sites-enabled/ |

1. Cấu hình frontend : fakeauto.id.vn

|  |
| --- |
| server {  root /var/www/html;  index index.html;  server\_name fakeauto.id.vn www.fakeauto.id.vn;  location / {  proxy\_pass http://localhost:4173;  proxy\_http\_version 1.1;  proxy\_set\_header Upgrade $http\_upgrade;  proxy\_set\_header Connection 'upgrade';  proxy\_set\_header Host $host;  proxy\_cache\_bypass $http\_upgrade;  }  listen 443 ssl http2;  listen [::]:443 ssl ipv6only=on;  ssl\_certificate /etc/letsencrypt/live/fakeauto.id.vn/fullchain.pem;  ssl\_certificate\_key /etc/letsencrypt/live/fakeauto.id.vn/privkey.pem;  include /etc/letsencrypt/options-ssl-nginx.conf;  ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem;  }  server {  listen 80;  listen [::]:80;  server\_name fakeauto.id.vn www.fakeauto.id.vn;  # Tự động chuyển HTTP sang HTTPS  return 301 https://$host$request\_uri;  } |

1. Cấu hình backend : api.fakeauto.id.vn

|  |
| --- |
| server {  root /var/www/html;  index index.html;  server\_name api.fakeauto.id.vn www.api.fakeauto.id.vn;  location / {  proxy\_pass http://localhost:3000;  proxy\_http\_version 1.1;  proxy\_set\_header Upgrade $http\_upgrade;  proxy\_set\_header Connection 'upgrade';  proxy\_set\_header Host $host;  proxy\_cache\_bypass $http\_upgrade;  }  listen 443 ssl http2;  listen [::]:443 ssl;  ssl\_certificate /etc/letsencrypt/live/fakeauto.id.vn/fullchain.pem;  ssl\_certificate\_key /etc/letsencrypt/live/fakeauto.id.vn/privkey.pem;  include /etc/letsencrypt/options-ssl-nginx.conf;  ssl\_dhparam /etc/letsencrypt/ssl-dhparams.pem;  }  server {  listen 80;  listen [::]:80;  server\_name api.fakeauto.id.vn www.api.fakeauto.id.vn;  # Tự động chuyển hướng sang HTTPS  return 301 https://$host$request\_uri;  } |

**Cấu hình SSL**

Cả hai server đều sử dụng chứng chỉ SSL miễn phí do **Let's Encrypt** cung cấp, được cài đặt và quản lý bởi **Certbot**. Điều này giúp đảm bảo tính bảo mật và tin cậy khi người dùng truy cập hệ thống.

**Kiểm tra và khởi động lại Nginx**

|  |
| --- |
| sudo nginx -t # Kiểm tra cấu hình (syntax hợp lệ chưa)  sudo systemctl reload nginx # Áp dụng cấu hình mới |

=> Việc sử dụng Nginx làm reverse proxy giúp tách biệt tầng giao diện và tầng xử lý, đồng thời cải thiện hiệu năng và bảo mật cho toàn bộ hệ thống.

### 4.5.4. Tự động triển khai bằng GitHub Actions (CI/CD)

Để tăng tính chuyên nghiệp, hạn chế triển khai thủ công, và đảm bảo hệ thống luôn cập nhật mới nhất sau mỗi lần thay đổi mã nguồn, nhóm đã áp dụng mô hình **CI/CD** (Continuous Integration / Continuous Deployment) bằng **GitHub Actions**.

**Mô tả hệ thống CI/CD**

* **Trigger:** Mỗi khi có push hoặc merge pull request vào nhánh main, GitHub tự động thực hiện quy trình triển khai.
* **Máy chủ đích:** VPS chạy hệ điều hành Ubuntu.
* **Quy trình triển khai:**
  1. SSH vào VPS.
  2. Pull code mới nhất từ GitHub.
  3. Cài đặt lại dependencies (nếu có thay đổi).
  4. Build lại frontend (React + Vite).
  5. Copy dist/ vào thư mục /var/www/frontend/.
  6. Khởi động lại backend bằng PM2.
* Gửi **thông báo thành công hoặc thất bại lên Discord** qua webhook.

#### **Cấu hình GitHub Actions**

Tệp cấu hình CI/CD đặt tại đường dẫn:

|  |
| --- |
| .github/workflows/deploy.yml |

Nội dung cấu hình :

|  |
| --- |
| name: Deploy WebCuoiKy  on:    push:      branches: ['main']    pull\_request:      branches: ['main']  jobs:    deploy:      runs-on: ubuntu-latest      steps:        - name: Checkout source code          uses: actions/checkout@v4        - name: Deploy to VPS          uses: appleboy/ssh-action@v1.0.3          with:            host: ${{ secrets.HOST }}            username: ${{ secrets.USERNAME }}            key: ${{ secrets.PRIVATE\_KEY }}            script: |              set -e              export NVM\_DIR="$HOME/.nvm"              [ -s "$NVM\_DIR/nvm.sh" ] && \. "$NVM\_DIR/nvm.sh"              echo "Pulling latest code..."              cd /home/prod/cuoi-ky              git reset --hard              git pull origin main              echo "Deploying Backend..."              cd backend              npm install              pm2 startOrRestart ecosystem.config.js --name ServerExpress              echo "Deploying Frontend..."              cd ../frontend              npm install              npm run build              pm2 startOrRestart ecosystem.config.cjs --name ReactVite              echo "Deploy Complete!"        - name: ✅ Notify Discord Success!          if: success()          run: |            COMMIT\_MESSAGE=$(git log -1 --pretty=format:"%s")            COMMIT\_URL="https://github.com/${{ github.repository }}/commit/${{ github.sha }}"            curl -H "Content-Type: application/json" \              -X POST \              -d '{                "username": "Peter Parker 🕸️",                "avatar\_url": "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSFhtUfX2QPakxwIHOK4G1zf9ZIVi5FYhqyoQ&s",                "embeds": [                  {                    "title": "✅ Deploy thành công!",                    "description": "\*\*Branch\*\*: `${{ github.ref\_name }}`\n\*\*Commit\*\*: [`${{ github.sha }}`]('"$COMMIT\_URL"')\n\*\*Người thực hiện\*\*: `${{ github.actor }}`\n\*\*Nội dung commit\*\*: '"${COMMIT\_MESSAGE}"'",                    "color": 3066993,                    "footer": {                      "text": "GitHub Actions • Deploy Completed"                    },                    "timestamp": "'$(date --utc +%Y-%m-%dT%H:%M:%SZ)'"                  }                ]              }' \              ${{ secrets.DISCORD\_WEBHOOK }}        - name: ❌ Notify Discord Failure          if: failure()          run: |            COMMIT\_MESSAGE=$(git log -1 --pretty=format:"%s")            COMMIT\_URL="https://github.com/${{ github.repository }}/commit/${{ github.sha }}"            curl -H "Content-Type: application/json" \              -X POST \              -d '{                "username": "Peter Parker 🕸️",                "avatar\_url": "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSFhtUfX2QPakxwIHOK4G1zf9ZIVi5FYhqyoQ&s",                "embeds": [                  {                    "title": "❌ Deploy thất bại!",                    "description": "\*\*Branch\*\*: `${{ github.ref\_name }}`\n\*\*Commit\*\*: [`${{ github.sha }}`]('"$COMMIT\_URL"')\n\*\*Người thực hiện\*\*: `${{ github.actor }}`\n\*\*Nội dung commit\*\*: '"${COMMIT\_MESSAGE}"'",                    "color": 15158332,                    "footer": {                      "text": "GitHub Actions • Deploy Failed"                    },                    "timestamp": "'$(date --utc +%Y-%m-%dT%H:%M:%SZ)'"                  }                ]              }' \              ${{ secrets.DISCORD\_WEBHOOK }} |

**Bảo mật và quản lý secrets**

Toàn bộ thông tin nhạy cảm được lưu trong GitHub Secrets:

|  |  |
| --- | --- |
| Tên biến secret | Mô tả |
| HOST | |  | | --- | |  |  |  | | --- | | IP VPS | |
| USERNAME | |  | | --- | |  |  |  | | --- | | Tên tài khoản đăng nhập VPS | |
| PRIVATE\_KEY | |  | | --- | |  |  |  | | --- | | SSH private key để kết nối VPS | |
| DISCORD\_WEBHOOK | Webhook Discord gửi thông báo |

**Lợi ích đạt được**

* Không cần đăng nhập VPS để triển khai.
* Luôn đồng bộ phiên bản mới nhất sau mỗi lần đẩy code.
* Tự động gửi cảnh báo lên Discord nếu có lỗi.
* Phù hợp với môi trường DevOps và nhóm phát triển nhiều thành viên.

Việc tích hợp CI/CD giúp đảm bảo quy trình triển khai trở nên **nhanh chóng, tự động, an toàn và chuyên nghiệp**, đặc biệt phù hợp với các ứng dụng web hiện đại có frontend - backend riêng biệt.

# Chương 5 : Kết quả đạt được

## 5.1. Các phần đạt được:

|  |
| --- |
|  |
|  |
| Trang đăng nhập và trang đăng ký |

|  |
| --- |
|  |
| Trang chủ |

|  |
| --- |
|  |
| 👉 Lọc theo địa điểm |
|  |
| 👉 Lọc theo giá |
|  |
| 👉 Lọc theo năm sản xuất |
|  |
| 👉 Lọc theo km đã đi |
|  |
| 👉 Lọc theo số chỗ ngồi |
|  |
| 👉 Lọc theo tên |

|  |
| --- |
|  |
| 👉 Chi tiết xe |

|  |
| --- |
|  |
| Trang tổng quan |

|  |
| --- |
|  |
| Trang quản lý đăng bán xe (nếu có nhiều hơn 5 xe thì hiện phân trang) |

|  |
| --- |
|  |
| Trang thống kê bán hàng |

|  |
| --- |
|  |
| Trang thống kê lượt xe/ bán hàng |

|  |
| --- |
|  |
| Trang hướng dẫn / FAQ |

|  |
| --- |
|  |
| Trang hướng dẫn / FAQ |

|  |
| --- |
|  |
|  |
| Trang FAQ – trang chủ |

|  |
| --- |
|  |
| Trang liên hệ - trang chủ |

|  |
| --- |
|  |
| Trang bán xe |

|  |
| --- |
|  |
| Trang giới thiệu |

|  |
| --- |
|  |
|  |
| Trang nhắn tin |

|  |
| --- |
|  |
| Trang thông tin người bán |

|  |
| --- |
|  |
| Webhook thông báo tự động deploy thành công |

|  |  |
| --- | --- |
|  |  |
|  | |
| Giao diện khi mua xe: tạo đơn hàng, thanh toán, biểu mẫu xác nhận thanh toán | |

|  |  |
| --- | --- |
|  |  |
| Trang xem lịch sử đơn hàng và lịch sử giao dịch | |

|  |
| --- |
|  |
|  |
| Trang xem thông báo |

|  |
| --- |
|  |
|  |
| Dashboard dành cho Admin để xem thống kê các số liệu trên hệ thống |

|  |
| --- |
|  |
|  |
|  |
|  |
| Dashboard dành cho Admin để quản lý người dùng trên hệ thống |

|  |
| --- |
|  |
| Dashboard dành cho Admin để quản lý xe trên hệ thống |

|  |  |
| --- | --- |
|  | |
| Dashboard dành cho Admin để quản lý đơn hàng | |
|  | |
|  |  |
| Dashboard dành cho Admin để quản lý đơn hàng | |

## 5.2. Khó khăn, hạn chế

Trong quá trình thiết kế, lập trình và triển khai hệ thống, nhóm đã gặp phải một số khó khăn và tồn tại những hạn chế như sau:

* **Thiếu kinh nghiệm ban đầu** với các công nghệ như Socket.IO, Nginx, PM2, khiến việc tích hợp real-time chat và reverse proxy gặp nhiều lỗi và mất thời gian tìm hiểu.
* **Triển khai hệ thống lần đầu trên VPS** gặp khó khăn về phân quyền, bảo mật SSH, cấu hình SSL với Let's Encrypt.
* **Quản lý quy trình build-deploy phức tạp**, do hệ thống phân chia frontend/backend riêng biệt, mỗi bên có quy trình cài đặt và restart khác nhau.
* **Chưa kiểm thử trên quy mô lớn** do hạn chế về thiết bị, thời gian và người dùng thực tế.
* **Giao diện trên mobile chưa tối ưu hoàn toàn**, một số thành phần giao diện hiển thị chưa chính xác ở kích thước màn hình nhỏ.
* **Tính năng chat chưa hỗ trợ gửi ảnh/video**, chỉ dừng lại ở việc gửi văn bản.

## 5.3. Hướng mở rộng trong tương lai

Để hoàn thiện và phát triển hệ thống theo hướng chuyên nghiệp, nhóm đề xuất một số định hướng mở rộng trong tương lai như sau:

* **Bổ sung chức năng gửi ảnh và video trong chat**, kết hợp lưu trữ đám mây như Cloudinary hoặc Firebase Storage.
* **Tích hợp bản đồ (Google Maps)** để hiển thị vị trí đăng bán xe
* **Xây dựng ứng dụng di động (React Native)** để mở rộng phạm vi người dùng và nâng cao trải nghiệm trên mobile.
* **Thêm tính năng thống kê nâng cao** cho người quản trị: số lượng người dùng, số lượt đăng tin, hoạt động theo ngày/tuần.
* **Tách quyền người dùng chi tiết hơn**: người dùng, người bán, admin, kiểm duyệt viên.

Những hướng mở rộng này sẽ giúp hệ thống tiếp cận gần hơn với mô hình thực tế, phục vụ được lượng người dùng lớn và chuyên nghiệp hơn.

# Chương 6: Kết luận và Hướng phát triển

## 6.1. Kết luận

Sau một quá trình tìm hiểu, phân tích, thiết kế và xây dựng, nhóm đã hoàn thiện hệ thống website **FakeAuto – Nền tảng đăng tin mua bán xe trực tuyến** với các tính năng cơ bản và cần thiết như:

* Hệ thống phân quyền người dùng: người dùng, người bán.
* Tính năng đăng nhập, đăng ký, xác thực qua email.
* Chức năng đăng tin mua bán xe, phân loại theo danh mục.
* Hệ thống chat thời gian thực (real-time) giữa người bán và người mua.
* Trang quản lý người dùng, thống kê, duyệt tin dành cho quản trị viên.
* Cơ chế triển khai CI/CD hiện đại qua GitHub Actions.

Trong quá trình thực hiện, nhóm đã vận dụng nhiều công nghệ hiện đại như **React, Node.js, MongoDB, Socket.IO, PM2, Nginx**, đồng thời rèn luyện được kỹ năng làm việc nhóm, xử lý lỗi, triển khai thực tế trên môi trường VPS.

Dù còn một số hạn chế, nhưng hệ thống đã đạt được mục tiêu đặt ra ban đầu và có thể tiếp tục phát triển, hoàn thiện thêm nhiều tính năng nâng cao.

## 6.2. Hướng phát triển

Trong tương lai, nhóm dự định tiếp tục cải thiện và mở rộng hệ thống theo các hướng sau:

* Tối ưu giao diện người dùng, đặc biệt trên thiết bị di động.
* Bổ sung chức năng gửi ảnh, video trong chat.
* Triển khai phiên bản mobile (React Native hoặc Flutter).
* Tích hợp bản đồ và tìm kiếm nâng cao theo vị trí.
* Cải thiện bảo mật, thêm xác thực 2 bước.
* Phát triển hệ thống thông báo (email, push).
* Kiểm thử hiệu năng với số lượng lớn người dùng thực tế.

Qua đồ án này, nhóm không chỉ củng cố kiến thức chuyên môn mà còn có thêm kinh nghiệm thực tế quý báu để phát triển các sản phẩm web hoàn chỉnh trong tương lai.

**Tài liệu tham khảo**

**Mã nguồn Github** : <https://github.com/Bai-t-p-l-p-trinh-web/cuoi-ky>

**Backend**

[1] **axios (^1.9.0)**: Mã nguồn: <https://www.npmjs.com/package/axios>  
[2] **bcrypt (^5.1.1)**: Mã nguồn: <https://www.npmjs.com/package/bcrypt>  
[3] **cloudinary (^2.6.1)**: Mã nguồn: <https://www.npmjs.com/package/cloudinary>  
[4] **cookie-parser (^1.4.7)**: Mã nguồn: <https://www.npmjs.com/package/cookie-parser>  
[5] **dotenv (^16.5.0)**: Mã nguồn: <https://www.npmjs.com/package/dotenv>  
[6] **express (^5.1.0)**: Mã nguồn: <https://www.npmjs.com/package/express>  
[7] **express-session (^1.18.1)**: Mã nguồn: <https://www.npmjs.com/package/express-session>  
[8] **fontkit (^2.0.4)**: Mã nguồn: <https://www.npmjs.com/package/fontkit>  
[9] **fs (^0.0.1-security)**: Mã nguồn: <https://www.npmjs.com/package/fs>  
[10] **jsonwebtoken (^9.0.2)**: Mã nguồn: <https://www.npmjs.com/package/jsonwebtoken>  
[11] **md5 (^2.3.0)**: Mã nguồn: <https://www.npmjs.com/package/md5>  
[12] **mongodb (^6.16.0)**: Mã nguồn: <https://www.npmjs.com/package/mongodb>  
[13] **mongoose (^8.14.0)**: Mã nguồn: <https://www.npmjs.com/package/mongoose>  
[14] **mongoose-slug-updater (^3.3.0)**: Mã nguồn: <https://www.npmjs.com/package/mongoose-slug-updater>  
[15] **node-cron (^4.0.6)**: Mã nguồn: <https://www.npmjs.com/package/node-cron>  
[16] **node-telegram-bot-api (^0.66.0)**: Mã nguồn: <https://www.npmjs.com/package/node-telegram-bot-api>  
[17] **nodemailer (^7.0.3)**: Mã nguồn: <https://www.npmjs.com/package/nodemailer>  
[18] **nodemon (^3.1.10)**: Mã nguồn: <https://www.npmjs.com/package/nodemon>  
[19] **passport (^0.7.0)**: Mã nguồn: <https://www.npmjs.com/package/passport>  
[20] **passport-google-oauth20 (^2.0.0)**: Mã nguồn: [https://www.npmjs.com/package/passport-google-oauth20/](https://www.npmjs.com/package/passport-google-oauth20)  
[21] **pdf-lib (^1.17.1)**: Mã nguồn: <https://www.npmjs.com/package/pdf-lib>  
[22] **socket.io (^4.8.1)**: Mã nguồn: <https://www.npmjs.com/package/socket.io>  
[23] **streamifier (^0.1.1)**: Mã nguồn: <https://www.npmjs.com/package/streamifier>  
[24] **swagger-ui-express (^5.0.1)**: Mã nguồn: [https://www.npmjs.com/package/swagger-ui-express/](https://www.npmjs.com/package/swagger-ui-express)  
[25] **validator (^13.15.15)**: Mã nguồn: <https://www.npmjs.com/package/validator>  
[26] **yamljs (^0.3.0)**: Mã nguồn: <https://www.npmjs.com/package/yamljs>

* **Optional Dependencies**: [27] **bufferutil (^4.0.9)**: Mã nguồn: <https://www.npmjs.com/package/bufferutil>  
  [28] **utf-8-validate (^6.0.5)**: Mã nguồn: <https://www.npmjs.com/package/utf-8-validate>

**Frontend**

[1] **@cloudinary/react (^1.14.3)**: Mã nguồn: <https://www.npmjs.com/package/@cloudinary/react>  
[2] **@cloudinary/url-gen (^1.21.0)**: Mã nguồn: <https://www.npmjs.com/package/@cloudinary/url-gen>  
[3] **@react-google-maps/api (^2.20.6)**: Mã nguồn: <https://www.npmjs.com/package/@react-google-maps/api>  
[4] **@reduxjs/toolkit (^2.8.2)**: Mã nguồn: <https://www.npmjs.com/package/@reduxjs/toolkit>  
[5] **@tailwindcss/vite (^4.1.4)**: Mã nguồn: <https://www.npmjs.com/package/@tailwindcss/vite>  
[6] **axios (^1.9.0)**: Mã nguồn: <https://www.npmjs.com/package/axios>  
[7] **bootstrap (^5.3.5)**: Mã nguồn: <https://www.npmjs.com/package/bootstrap>  
[8] **chart.js (^4.4.9)**: Mã nguồn: <https://www.npmjs.com/package/chart.js>  
[9] **lucide-react (^0.503.0)**: Mã nguồn: <https://www.npmjs.com/package/lucide-react>  
[10] **react (^19.0.0)**: Mã nguồn: <https://www.npmjs.com/package/react>  
[11] **react-chartjs-2 (^5.3.0)**: Mã nguồn: <https://www.npmjs.com/package/react-chartjs-2>  
[12] **react-dom (^19.1.0)**: Mã nguồn: <https://www.npmjs.com/package/react-dom>  
[13] **react-icons (^5.5.0)**: Mã nguồn: <https://www.npmjs.com/package/react-icons>  
[14] **react-loading-skeleton (^3.5.0)**: Mã nguồn: <https://www.npmjs.com/package/react-loading-skeleton>  
[15] **react-redux (^9.2.0)**: Mã nguồn: <https://www.npmjs.com/package/react-redux>  
[16] **react-router-dom (^7.5.1)**: Mã nguồn: <https://www.npmjs.com/package/react-router-dom>  
[17] **react-toastify (^11.0.5)**: Mã nguồn: <https://www.npmjs.com/package/react-toastify>  
[18] **sass (^1.86.3)**: Mã nguồn: <https://www.npmjs.com/package/sass>  
[19] **swiper (^11.2.6)**: Mã nguồn: <https://www.npmjs.com/package/swiper>

* **Dev Dependencies**: [20] **@eslint/js (^9.22.0)**: Mã nguồn: <https://www.npmjs.com/package/@eslint/js>  
  [21] **@types/react (^19.0.10)**: Mã nguồn: <https://www.npmjs.com/package/@types/react>  
  [22] **@types/react-dom (^19.0.4)**: Mã nguồn: <https://www.npmjs.com/package/@types/react-dom>  
  [23] **@vitejs/plugin-react (^4.3.4)**: Mã nguồn: <https://www.npmjs.com/package/@vitejs/plugin-react>  
  [24] **eslint (^9.22.0)**: Mã nguồn: <https://www.npmjs.com/package/eslint>  
  [25] **eslint-plugin-react-hooks (^5.2.0)**: Mã nguồn: <https://www.npmjs.com/package/eslint-plugin-react-hooks>  
  [26] **eslint-plugin-react-refresh (^0.4.19)**: Mã nguồn: <https://www.npmjs.com/package/eslint-plugin-react-refresh>  
  [27] **globals (^16.0.0)**: Mã nguồn: <https://www.npmjs.com/package/globals>  
  [28] **vite (^6.3.1)**: Mã nguồn: <https://www.npmjs.com/package/vite>