

BỘ LAO ĐỘNG THƯƠNG BINH VÀ XÃ HỘI
TRƯỜNG CAO ĐẲNG CÔNG NGHỆ THÔNG TIN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN – ĐIỆN TỬ



ĐỒ ÁN MÔN HỌC: CHUYÊN ĐỀ BACK-END
ĐỀ TÀI: XÂY DỰNG WEBSITE QUẢN LÝ NGƯỜI DÙNG



Giáo viên hướng dẫn :

Sinh viên thực hiện: **Họ tên SV - MSSV**

Khoá :

TP. Hồ Chí Minh, ngày tháng năm

LỜI MỞ ĐẦU

Ngày nay, việc quản lý người dùng hiệu quả là một yếu tố không thể thiếu đối với bất kỳ hệ thống phần mềm hay ứng dụng web nào. Từ việc lưu trữ thông tin cá nhân, phân quyền truy cập, đến đảm bảo tính bảo mật cho dữ liệu, một hệ thống quản lý người dùng đóng vai trò quan trọng trong hoạt động vận hành và phát triển của các tổ chức và doanh nghiệp.

Với mong muốn học hỏi và áp dụng kiến thức vào thực tế, chúng em thực hiện dự án xây dựng một website quản lý người dùng với mục tiêu tạo ra một hệ thống backend mạnh mẽ, linh hoạt và bảo mật. Dự án không chỉ mang tính học thuật mà còn mở ra cơ hội để chúng em tìm hiểu sâu hơn về các công nghệ hiện đại như Node.js, Express.js, MongoDB, và các công cụ hỗ trợ như JWT, Nodemailer.

Website quản lý người dùng này bao gồm các chức năng cơ bản như tạo, sửa, xóa, và tìm kiếm thông tin người dùng, đồng thời tích hợp cơ chế xác thực và phân quyền nhằm bảo vệ dữ liệu và đảm bảo người dùng chỉ được truy cập vào các phần phù hợp với vai trò của mình. Ngoài ra, dự án còn triển khai tính năng gửi email thông qua Nodemailer, hỗ trợ xác nhận tài khoản hoặc đặt lại mật khẩu.

Chúng em hy vọng dự án này không chỉ giúp hoàn thiện các yêu cầu học tập mà còn là một bước đệm giúp chúng em chuẩn bị tốt hơn cho công việc thực tế trong lĩnh vực phát triển ứng dụng web.

LỜI CẢM ƠN

Trước hết, chúng em xin gửi lời cảm ơn chân thành tới các thầy cô trong bộ môn Công nghệ Thông tin của trường Cao đẳng Công nghệ Thông Tin TPHCM, đặc biệt là thầy Nguyễn Minh Hải đã hướng dẫn, cung cấp kiến thức chuyên môn, hỗ trợ chúng em trong suốt quá trình thực hiện dự án. Những kiến thức mà thầy chia sẻ chính là nền tảng vững chắc để chúng em hoàn thành sản phẩm này.

Chúng em cũng xin bày tỏ lòng biết ơn đến các bạn trong nhóm đã nhiệt tình hỗ trợ, chia sẻ kinh nghiệm và giúp đỡ nhau trong những lúc gặp khó khăn.

Cuối cùng, chúng em muốn gửi lời cảm ơn đến gia đình và bạn bè đã luôn động viên, khích lệ tinh thần, tạo điều kiện tốt nhất để chúng em có thể tập trung vào việc học tập và hoàn thành dự án này. Chúng em tin rằng sản phẩm "Website quản lý người dùng" này không chỉ đánh dấu sự nỗ lực trong học tập mà còn là nền tảng để chúng em tiếp tục phát triển và hoàn thiện bản thân trong hành trình sự nghiệp phía trước.

MỤC LỤC

Chương 1. GIỚI THIỆU	1
1.1. Mục tiêu dự án.....	1
1.1.1. Các chức năng cơ bản	1
1.1.2. Mục đích phát triển ứng dụng Back-End.....	1
1.2. Công nghệ sử dụng.....	2
1.2.1. NodeJS	2
1.2.2. Express.....	2
1.2.3. MongoDB	3
1.2.4. JWT.....	4
1.2.5. Nodemailer.....	4
1.2.6. EJS	5
1.2.7. Dotenv.....	6
1.2.8. Bcrypt.....	6
1.3. Lý do chọn công nghệ	7
1.3.1. NodeJS	7
1.3.2. Express.....	7
1.3.3. MongoDB	7
1.3.4. JWT.....	7
1.3.5. Nodemailer.....	7
Chương 2. THIẾT KẾ KIẾN TRÚC HỆ THỐNG	8
2.1. Kiến trúc ứng dụng.....	8
2.1.1. Client (Frontend).....	8
2.1.2. Server (Backend)	8
2.1.3. Database (MongoDB).....	8
2.1.4. Middleware	8
2.1.5. Sơ đồ kiến trúc ứng dụng.....	9
2.2. Cấu trúc thư mục	9
2.2.1. config/	10
2.2.2. controllers/	10
2.2.3. middlewares/	10
2.2.4. models/	10
2.2.5. routes/.....	10
2.2.6. utils/	10
Chương 3. CÀI ĐẶT THỰC NGHIỆM VÀ KẾT QUẢ.....	11
3.1. Một số giao diện website.....	11
3.1.1. Giao diện đăng ký	11
3.1.2. Giao diện đăng nhập	11
3.1.3. Giao diện danh sách người dùng.....	12
3.1.4. Giao diện thêm mới người dùng	12

3.1.5.	Giao diện lỗi 404.....	13
3.2.	Các chức năng chính	13
3.2.1.	Chức năng đăng ký và đăng nhập	13
3.2.2.	Chức năng quên mật khẩu.....	17
3.2.3.	Chức năng thêm mới người dùng	19
3.2.4.	Chức năng xem chi tiết người dùng.....	20
3.2.5.	Chức năng sửa thông tin người dùng.....	20
3.2.6.	Chức năng xóa người dùng.....	21
3.2.7.	Chức năng tìm kiếm.....	21
Chương 4.	XỬ LÝ LỖI VÀ BẢO MẬT	22
4.1.	Xử lý lỗi trong ứng dụng.....	22
4.1.1.	Lỗi 400 – Bad Request.....	22
4.1.2.	Lỗi 401 – Unauthorized	22
4.1.3.	Lỗi 404 – Not Found.....	23
4.1.4.	Lỗi 500 – Internal Server Error.....	24
4.2.	Bảo mật ứng dụng	25
4.2.1.	Mã hóa mật khẩu.....	25
4.2.2.	Bảo vệ route	25
Chương 5.	KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	27
5.1.	Kết luận	27
5.2.	Hướng phát triển tương lai	28

DANH MỤC HÌNH

Hình 1.2.1 NodeJS	2
Hình 1.2.2 ExpressJS	3
Hình 1.2.3 MongoDB	4
Hình 1.2.4 JWT	4
Hình 1.2.5 Nodemailer.....	5
Hình 1.2.6 EJS	5
Hình 1.2.7 Dotenv.....	6
Hình 1.2.8 Bcrypt.....	6
Hình 2.1.5 Sơ đồ kiến trúc ứng dụng.....	9
Hình 2.2 Cấu trúc thư mục server	9
Hình 3.1.1 Giao diện đăng ký	11
Hình 3.1.2 Giao diện đăng nhập	11
Hình 3.1.3 Giao diện danh sách người dùng.....	12
Hình 3.1.4 Giao diện thêm mới người dùng	12
Hình 3.1.5 Giao diện lỗi 404.....	13
Hình 3.2.1.1a Đăng ký tài khoản	14
Hình 3.2.1.1b Thông báo đăng ký thành công.....	14
Hình 3.2.1.2a Email xác thực gửi tới.....	15
Hình 3.2.1.2b Đường dẫn để xác thực tài khoản	15
Hình 3.2.1.3 Khi chưa xác thực tài khoản	16
Hình 3.2.1.4 Khi chưa đăng nhập	16
Hình 3.2.2a Nhập thông tin người dùng mới.....	19
Hình 3.2.2b Thông báo thêm người dùng thành công	20
Hình 3.2.3 Giao diện xem chi tiết người dùng.....	20
Hình 3.2.4a Cập nhật hình ảnh người dùng	20
Hình 3.2.4b Thông báo thay đổi thành công.....	21
Hình 3.2.5 Thông báo khi muốn xóa người dùng	21
Hình 3.2.6 Tìm kiếm người dùng theo tên.....	21
Hình 4.1.1 Thông báo lỗi 400	22
Hình 4.1.2 Thông báo lỗi 401	23
Hình 4.1.3 Thông báo lỗi 404	24
Hình 4.1.4 Thông báo lỗi 500	25

Chương 1. GIỚI THIỆU

1.1. Mục tiêu dự án

Dự án này nhằm mục đích xây dựng một ứng dụng web quản lý người dùng (User Management Application) với backend được phát triển bằng Node.js và các công nghệ đã liệt kê ở trên.

1.1.1. Các chức năng cơ bản

- Đăng ký: Admin có thể tạo tài khoản mới.
- Đăng nhập: Admin có thể đăng nhập bằng tài khoản đã đăng ký.
- Đăng xuất: Admin có thể đăng xuất khỏi hệ thống.
- Tạo người dùng: Admin (hoặc người dùng có quyền) có thể tạo tài khoản người dùng mới.
- Sửa thông tin người dùng: Admin (hoặc người dùng có quyền) có thể cập nhật thông tin người dùng.
- Xóa người dùng: Admin (hoặc người dùng có quyền) có thể xóa tài khoản người dùng.
- Tìm kiếm người dùng: Tìm kiếm người dùng theo tên, email, hoặc các tiêu chí khác.

1.1.2. Mục đích phát triển ứng dụng Back-End

- Cung cấp API: Backend sẽ cung cấp các API (Application Programming Interfaces) để frontend (giao diện người dùng) có thể tương tác với dữ liệu người dùng.
- Xử lý logic nghiệp vụ: Backend sẽ xử lý các logic nghiệp vụ liên quan đến việc quản lý người dùng, chẳng hạn như xác thực, ủy quyền, kiểm tra dữ liệu đầu vào, v.v.
- Lưu trữ dữ liệu: Backend sẽ chịu trách nhiệm lưu trữ dữ liệu người dùng trong cơ sở dữ liệu (MongoDB).
- Bảo mật: Backend sẽ đảm bảo an toàn cho dữ liệu người dùng và ngăn chặn các truy cập trái phép.

1.2. Công nghệ sử dụng

1.2.1. NodeJS

NodeJS là một môi trường thực thi (runtime environment) JavaScript mã nguồn mở, chạy trên nền tảng V8 Engine (cũng được dùng trong trình duyệt Chrome). Nó cho phép ta chạy mã JavaScript bên ngoài trình duyệt web, chủ yếu để xây dựng các ứng dụng mạng phía máy chủ (server-side) nhanh chóng và có khả năng mở rộng.

❖ Tăng cường hiệu suất: Nodemon

Nodemon là một công cụ tiện ích giúp tự động khởi động lại ứng dụng Node.js khi phát hiện các thay đổi trong mã nguồn. Điều này giúp tăng tốc quá trình phát triển vì ta không cần phải dừng và chạy lại server thủ công mỗi khi sửa code.



Hình 1.2.1 NodeJS

1.2.2. Express

Express là một framework ứng dụng web tối giản và linh hoạt cho Node.js. Nó cung cấp một tập hợp các tính năng mạnh mẽ để xây dựng các ứng dụng web và API một cách dễ dàng và nhanh chóng. Có thể nói Express là framework phổ biến nhất trong hệ sinh thái Node.js.

❖ Tăng cường hiệu suất:

- **express-session:** Là một middleware dùng để quản lý phiên làm việc (session) trong Express. Nó cho phép ta lưu trữ dữ liệu người dùng giữa các request khác nhau (ví dụ: thông tin đăng nhập).

- `express-flash-messages`: (Thường được dùng với tên `connect-flash`) Là middleware cho phép ta lưu trữ các thông báo tạm thời (flash messages) và hiển thị chúng cho người dùng ở lần request tiếp theo (ví dụ: thông báo thành công/thất bại sau khi đăng ký/đăng nhập).
- `express-ejs-layouts`: Là một thư viện giúp quản lý layout (bố cục) cho các view sử dụng EJS trong Express. Nó cho phép ta định nghĩa layout chung và tái sử dụng cho nhiều trang khác nhau, giúp code gọn gàng và dễ bảo trì hơn.



Hình 1.2.2 ExpressJS

1.2.3. MongoDB

MongoDB là một hệ quản trị cơ sở dữ liệu (DBMS) NoSQL mã nguồn mở, hướng tài liệu (document-oriented). Thay vì lưu trữ dữ liệu trong các bảng với cấu trúc cố định như SQL, MongoDB lưu trữ dữ liệu trong các document có định dạng giống JSON (gọi là BSON).

❖ Tăng cường hiệu suất: Mongoose

Mongoose là một thư viện Object Document Modeling (ODM) cho MongoDB và Node.js. Nó cung cấp một cách thức đơn giản và trực quan để tương tác với cơ sở dữ liệu MongoDB, cho phép ta định nghĩa các schema (cấu trúc dữ liệu), thực hiện các truy vấn, và quản lý các mối quan hệ giữa các document.



Hình 1.2.3 MongoDB

1.2.4. JWT

JSON Web Token (JWT) là một tiêu chuẩn mở (RFC 7519) định nghĩa một cách thức nhỏ gọn và độc lập để truyền thông tin giữa các bên dưới dạng đối tượng JSON. Thông tin này có thể được xác minh và tin cậy vì nó được ký điện tử. JWT thường được sử dụng để xác thực (authentication) và ủy quyền (authorization) trong các ứng dụng web.



Hình 1.2.4 JWT

1.2.5. Nodemailer

Nodemailer là một module cho Node.js cho phép ta gửi email một cách dễ dàng. Nó hỗ trợ nhiều phương thức gửi mail khác nhau (SMTP, Sendmail, ...), có thể đính kèm file, và sử dụng HTML trong nội dung email.

❖ Tăng cường hiệu suất:

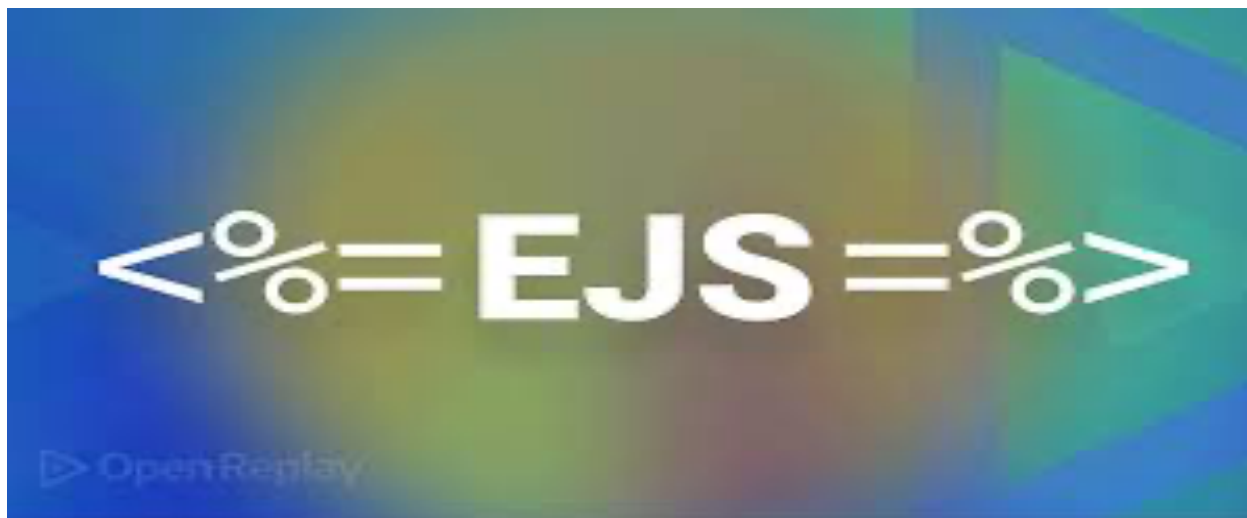
nodemailer-express-handlebars là một plugin cho Nodemailer cho phép ta sử dụng template engine Handlebars (tương tự EJS) để tạo nội dung email. Điều này giúp ta dễ dàng tạo ra các email động với nội dung được tùy biến theo dữ liệu từ ứng dụng.



Hình 1.2.5 Nodemailer

1.2.6. EJS

EJS (Embedded JavaScript templates) là một template engine đơn giản cho phép ta nhúng mã JavaScript vào trong các file HTML. Nó giúp ta dễ dàng tạo ra các trang web động bằng cách kết hợp dữ liệu từ server với các mẫu HTML.



Hình 1.2.6 EJS

1.2.7. Dotenv

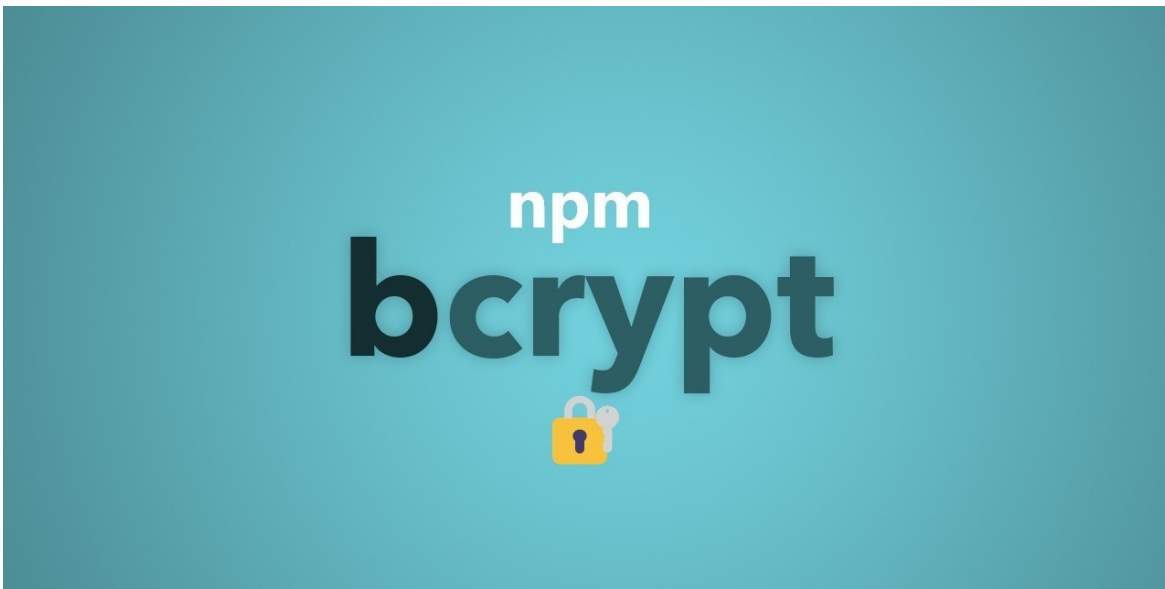
Dotenv là một module cho phép ta tải các biến môi trường (environment variables) từ một file .env vào trong ứng dụng Node.js. Việc này giúp ta quản lý các cấu hình (config) một cách dễ dàng và bảo mật hơn, đặc biệt là khi ta cần cấu hình khác nhau cho các môi trường phát triển, thử nghiệm, và production.



Hình 1.2.7 Dotenv

1.2.8. Bcrypt

Bcrypt là một hàm băm mật khẩu (password hashing function) được thiết kế để an toàn và chậm, làm cho việc tấn công brute-force (thử tất cả các mật khẩu có thể) trở nên khó khăn hơn. Nó thường được sử dụng để lưu trữ mật khẩu người dùng một cách an toàn trong cơ sở dữ liệu.



Hình 1.2.8 Bcrypt

1.3. Lý do chọn công nghệ

1.3.1. NodeJS

- JavaScript đồng nhất: Sử dụng JavaScript cho cả frontend và backend giúp đơn giản hóa việc phát triển và dễ dàng chia sẻ code giữa hai phần.
- Hiệu suất tốt: Node.js hoạt động nhanh và phù hợp với các ứng dụng cơ bản như API nhờ cơ chế xử lý bất đồng bộ (non-blocking).
- Hệ sinh thái lớn: Node.js có một hệ sinh thái npm (Node Package Manager) rất lớn với nhiều thư viện và công cụ hữu ích.

1.3.2. Express

- Đơn giản và linh hoạt: Express cung cấp một framework tối giản nhưng mạnh mẽ, cho phép ta dễ dàng xây dựng các API và ứng dụng web.
- Middleware: Hệ thống middleware của Express giúp ta dễ dàng mở rộng chức năng của ứng dụng.
- Phổ biến: Express là framework phổ biến nhất cho Node.js, có cộng đồng hỗ trợ lớn.

1.3.3. MongoDB

- Linh hoạt: Schema linh hoạt của MongoDB cho phép ta dễ dàng thay đổi cấu trúc dữ liệu khi cần thiết.
- Khả năng mở rộng: MongoDB được thiết kế để có thể mở rộng theo chiều ngang (horizontal scaling), phù hợp với các ứng dụng có lượng dữ liệu lớn.
- Hiệu suất tốt: MongoDB có hiệu suất đọc/ghi tốt, đặc biệt là với các ứng dụng có nhiều thao tác ghi.

1.3.4. JWT

- Xác thực và ủy quyền: Cung cấp một cách thức an toàn và hiệu quả để xác thực và ủy quyền người dùng.

1.3.5. Nodemailer

- Gửi email: Cho phép ứng dụng gửi email thông báo, xác nhận, v.v.

Chương 2. THIẾT KẾ KIẾN TRÚC HỆ THỐNG

2.1. Kiến trúc ứng dụng

Ứng dụng sẽ hoạt động dựa trên các thành phần chính sau:

2.1.1. Client (Frontend)

- Là nơi người dùng tương tác với hệ thống, gửi các yêu cầu (request) như đăng ký, đăng nhập, hoặc xem danh sách người dùng.
- Client giao tiếp với server thông qua API endpoints (ví dụ: */api/login*, */api/users*).
- Dữ liệu trả về từ server (response) thường ở định dạng JSON để dễ dàng xử lý trên client.

2.1.2. Server (Backend)

- Xử lý các yêu cầu từ phía client.
- Quản lý logic ứng dụng, xác thực người dùng, và kết nối với cơ sở dữ liệu.
- Phân tầng chính:
 - Routes: Xử lý các endpoint, định tuyến yêu cầu từ client đến controller phù hợp.
 - Controllers: Xử lý logic của ứng dụng, gọi dữ liệu từ Model, xử lý xong trả về response cho client.
 - Models: Quản lý dữ liệu, kết nối với cơ sở dữ liệu (MongoDB).

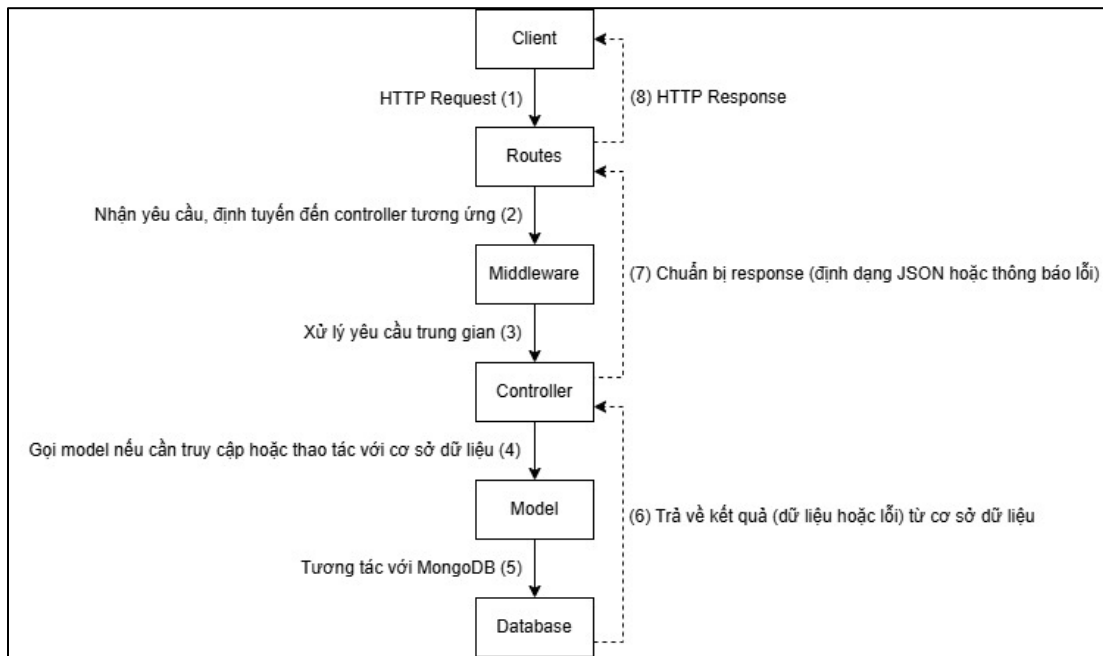
2.1.3. Database (MongoDB)

- Nơi lưu trữ dữ liệu người dùng, bao gồm thông tin đăng nhập, email, và các thông tin khác.
- Hỗ trợ các thao tác thêm, sửa, xóa và đọc dữ liệu (CRUD).

2.1.4. Middleware

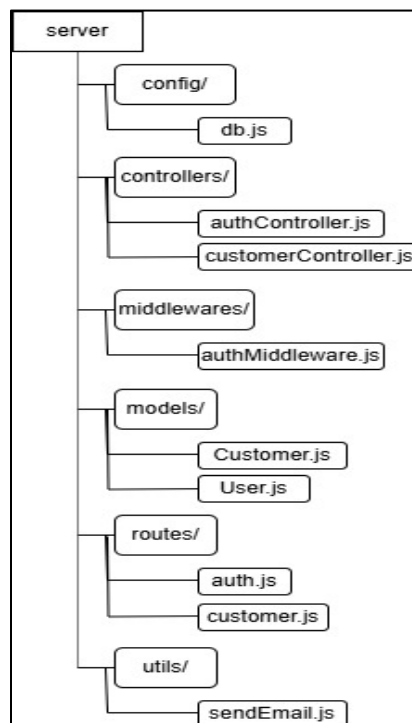
- Các lớp trung gian được sử dụng để xử lý yêu cầu trước khi chúng đến controller.
- Ví dụ: Middleware xác thực người dùng bằng JWT để bảo vệ các API nhạy cảm.

2.1.5. Sơ đồ kiến trúc ứng dụng



Hình 2.1.5 Sơ đồ kiến trúc ứng dụng

2.2. Cấu trúc thư mục



Hình 2.2 Cấu trúc thư mục server

2.2.1. config/

- Thư mục này chứa tệp *db.js* để cấu hình và thiết lập kết nối với cơ sở dữ liệu MongoDB.
- Vai trò chính: Quản lý kết nối cơ sở dữ liệu một cách tập trung và dễ bảo trì.

2.2.2. controllers/

- Chứa các tệp xử lý logic chính của ứng dụng.
- *authController.js*: Xử lý xác thực người dùng, bao gồm đăng nhập, đăng ký và cấp phát JWT token.
- *customerController.js*: Xử lý các thao tác liên quan đến khách hàng như thêm mới, sửa đổi, xóa hoặc lấy danh sách khách hàng.

2.2.3. middlewares/

- Thư mục này chứa các middleware – các lớp trung gian xử lý trước khi yêu cầu tới controller.
- *authMiddleware.js*: Xác thực token JWT để bảo vệ các route yêu cầu quyền truy cập.

2.2.4. models/

- Chứa các schema của MongoDB, định nghĩa cấu trúc dữ liệu lưu trong cơ sở dữ liệu.
- *Customer.js*: Định nghĩa schema và model của khách hàng (Customer).
- *User.js*: Định nghĩa schema và model của người dùng (User).

2.2.5. routes/

- Định nghĩa các endpoint API và liên kết với controller tương ứng.
- *auth.js*: Chứa các endpoint liên quan đến xác thực (đăng nhập, đăng ký).
- *customer.js*: Chứa các endpoint quản lý khách hàng (CRUD khách hàng).

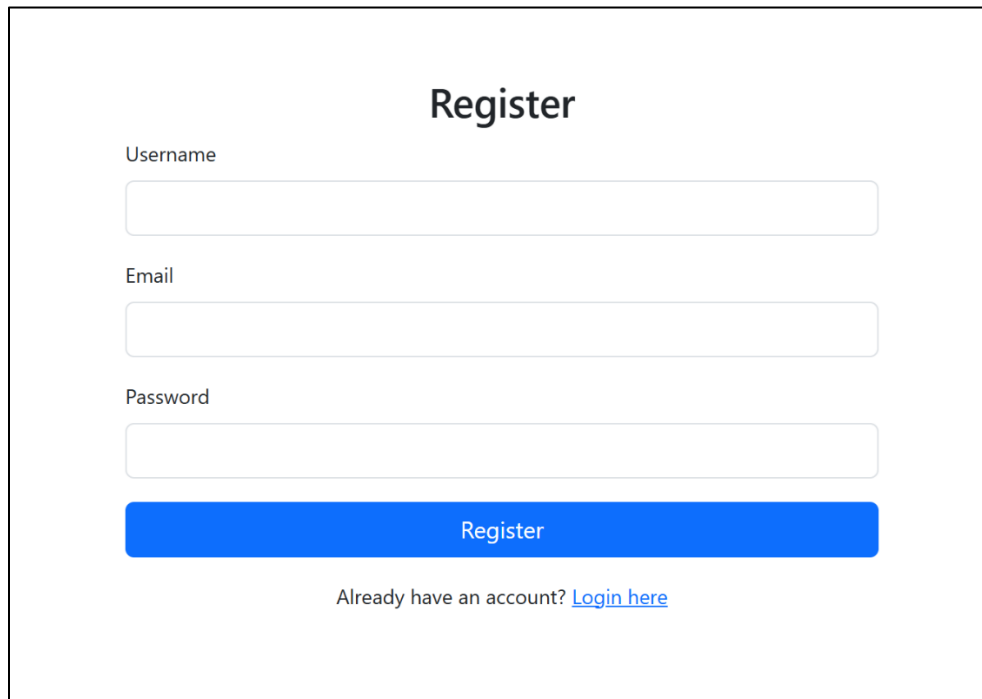
2.2.6. utils/

- Chứa các tiện ích dùng chung trong ứng dụng.
- *sendEmail.js*: Thư viện gửi email (sử dụng Nodemailer), thường dùng cho xác thực tài khoản hoặc gửi thông báo.

Chương 3. CÀI ĐẶT THỰC NGHIỆM VÀ KẾT QUẢ

3.1. Một số giao diện website

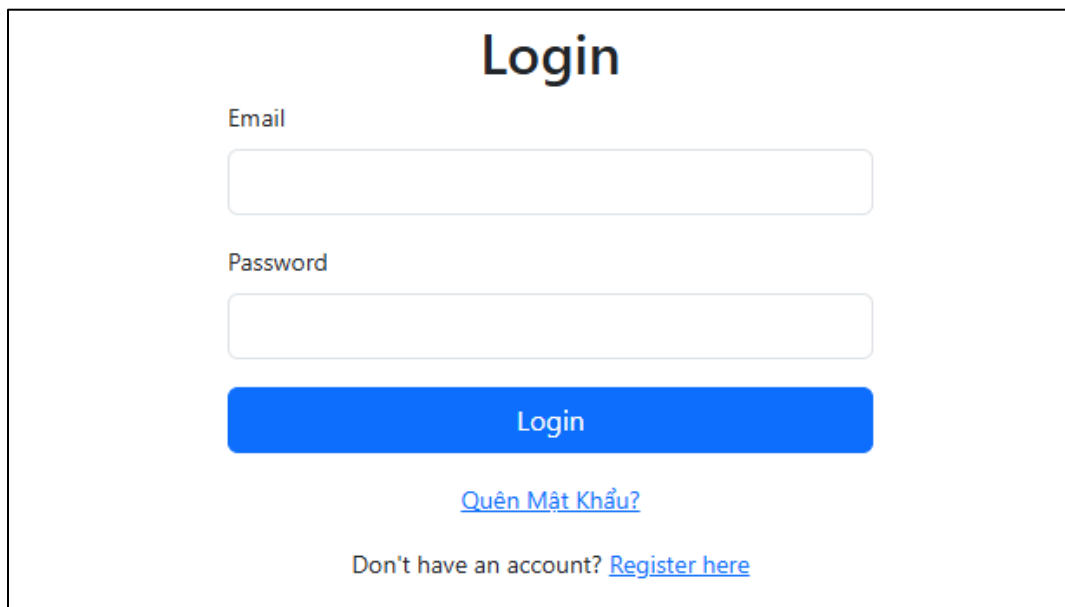
3.1.1. Giao diện đăng ký



The image shows a web registration form titled "Register". It contains three input fields: "Username", "Email", and "Password". Below these fields is a blue button labeled "Register". At the bottom, there is a link that says "Already have an account? [Login here](#)".

Hình 3.1.1 Giao diện đăng ký

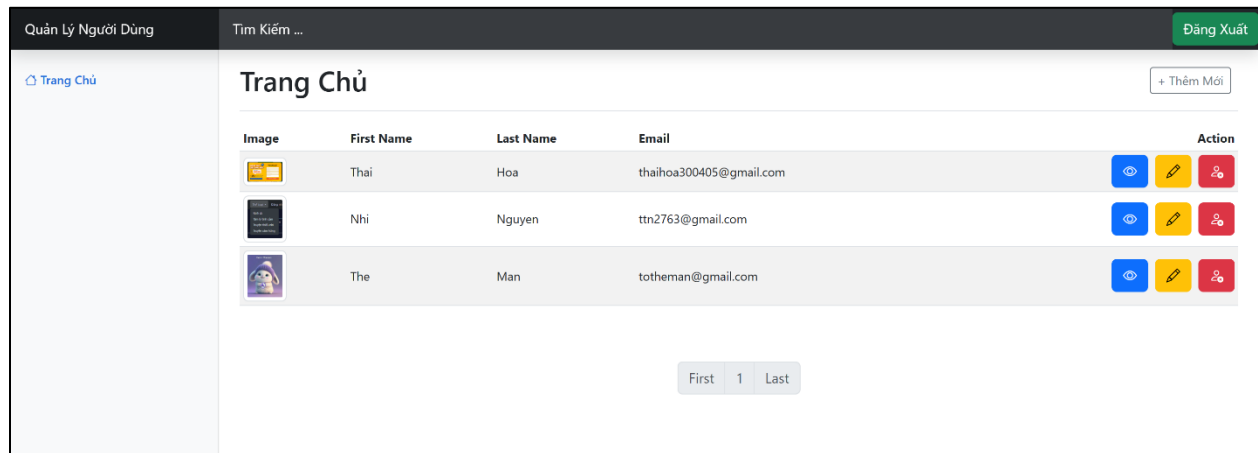
3.1.2. Giao diện đăng nhập



The image shows a web login form titled "Login". It contains two input fields: "Email" and "Password". Below these fields is a blue button labeled "Login". Below the button is a link that says "[Quên Mật Khẩu?](#)". At the bottom, there is a link that says "Don't have an account? [Register here](#)".

Hình 3.1.2 Giao diện đăng nhập

3.1.3. Giao diện danh sách người dùng



Hình 3.1.3 Giao diện danh sách người dùng

3.1.4. Giao diện thêm mới người dùng

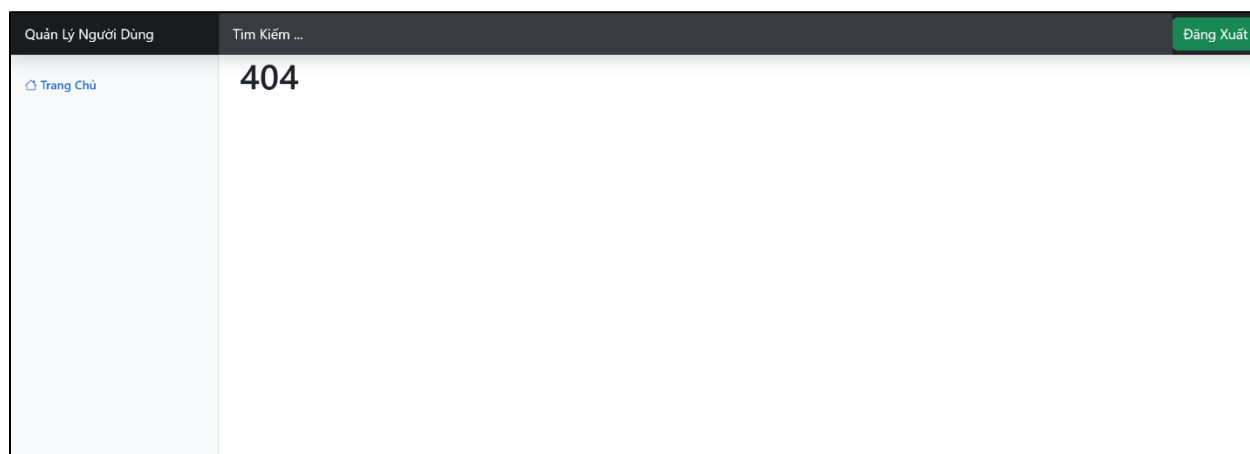
The screenshot shows the 'Thêm Người Dùng' (Add User) form. The top navigation bar is identical to the previous screenshot. The main content area is titled 'Thêm Người Dùng' and includes a breadcrumb 'Trang Chủ / Thêm Người Dùng' and a 'Mã Người Dùng' field. The form contains several input fields:

- First Name:
- Last Name:
- Telephone:
- Email:
- Customer Details:
- Hình Ảnh:

A blue 'Add Customer' button is located at the bottom of the form.

Hình 3.1.4 Giao diện thêm mới người dùng

3.1.5. Giao diện lỗi 404



Hình 3.1.5 Giao diện lỗi 404

3.2. Các chức năng chính

3.2.1. Chức năng đăng ký và đăng nhập

3.2.1.1. Đăng ký tài khoản

❖ Mục tiêu:

- Thu thập thông tin từ người dùng bao gồm: username, email, password.
- Mã hóa mật khẩu trước khi lưu vào cơ sở dữ liệu.
- Gửi email xác thực để kích hoạt tài khoản.

❖ Luồng hoạt động:

1. Người dùng nhập thông tin cần thiết (username, email, password).
2. Hệ thống kiểm tra:
 - Email đã tồn tại hay chưa.
 - Email có hợp lệ hay không.
3. Nếu thông tin hợp lệ:
 - Lưu người dùng vào cơ sở dữ liệu với trạng thái chưa được kích hoạt.
 - Tạo mã thông báo (token) để xác thực tài khoản qua email.
 - Gửi email xác thực.
4. Người dùng nhận email và nhấp vào liên kết để kích hoạt tài khoản

Register

Username

Email

Password

Register

Already have an account? [Login here](#)

Hình 3.2.1.1a Đăng ký tài khoản

Register

Registration successful! Please check your email to verify your account.

Username

Email

Password

Register

Already have an account? [Login here](#)

Hình 3.2.1.1b Thông báo đăng ký thành công

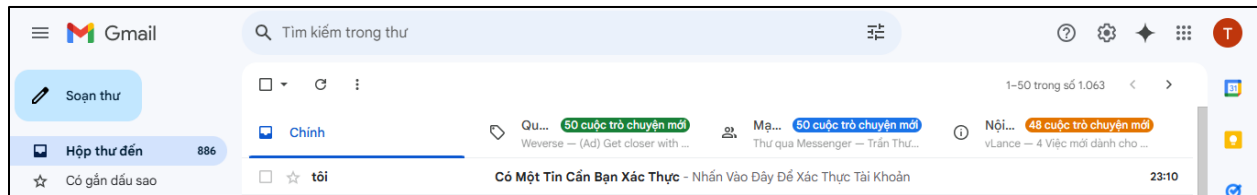
3.2.1.2. Xác thực email

❖ Mục tiêu:

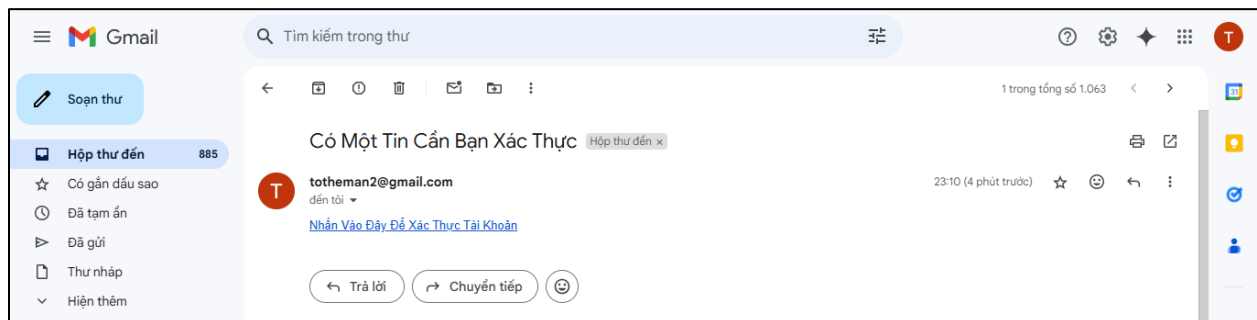
- Xác thực tài khoản người dùng khi nhấp vào liên kết trong email.

❖ Luồng hoạt động:

1. Người dùng nhấp vào liên kết xác thực trong email.
2. Hệ thống kiểm tra token hợp lệ hay không.
3. Nếu token hợp lệ:
 - Cập nhật trạng thái tài khoản sang isVerified: true.
 - Cho phép người dùng đăng nhập.
4. Nếu token không hợp lệ, thông báo lỗi.



Hình 3.2.1.2a Email xác thực gửi tới



Hình 3.2.1.2b Đường dẫn để xác thực tài khoản

3.2.1.3. Đăng nhập

❖ Mục tiêu:

- Cho phép người dùng đăng nhập nếu tài khoản đã được xác thực.
- Tạo JSON Web Token (JWT) để duy trì phiên đăng nhập.

❖ Luồng hoạt động:

1. Người dùng nhập email và mật khẩu.
2. Hệ thống kiểm tra:
 - Email có tồn tại không.

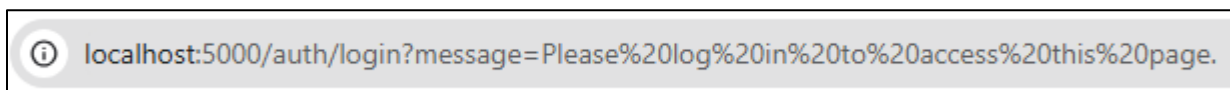
- Tài khoản đã được xác thực chưa (isVerified: true).
 - Mật khẩu có hợp lệ không.
3. Nếu hợp lệ:
- Tạo và trả về token JWT.
4. Nếu không hợp lệ, trả về lỗi tương ứng.

The image shows a login form with the title "Login" at the top. Below the title is a red error message box that says "Please verify your email before logging in." Underneath this, there are two input fields: "Email" and "Password". Below the password field is a blue "Login" button. At the bottom of the form, there is a link that says "Don't have an account? [Register here](#)".

Hình 3.2.1.3 Khi chưa xác thực tài khoản

3.2.1.4. Kiểm tra trạng thái đăng nhập

- ❖ Mục tiêu:
 - Xác định người dùng đã đăng nhập chưa trước khi truy cập các route yêu cầu quyền.
 - Bảo vệ các tài nguyên quan trọng, chỉ cho phép người dùng được xác thực mới có quyền truy cập.
 - Tăng cường bảo mật, tránh các truy cập trái phép hoặc không hợp lệ.



Hình 3.2.1.4 Khi chưa đăng nhập

3.2.2. Chức năng quên mật khẩu

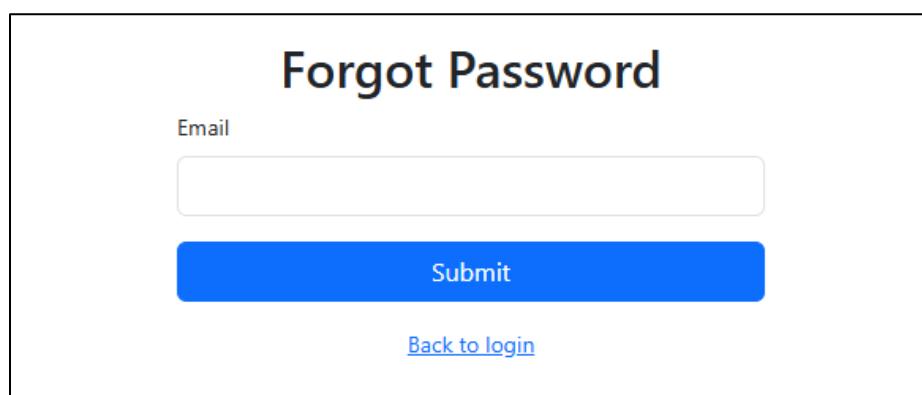
❖ Mục tiêu:

- Hỗ trợ người dùng đặt lại mật khẩu khi quên mật khẩu đăng nhập.
- Gửi email chứa liên kết đặt lại mật khẩu để xác minh.
- Cập nhật mật khẩu mới một cách an toàn và bảo mật.

❖ Luồng hoạt động:

1. Người dùng yêu cầu đặt lại mật khẩu:

- Trên trang đăng nhập, người dùng chọn "Quên Mật Khẩu?".
- Hệ thống hiển thị trang nhập email.

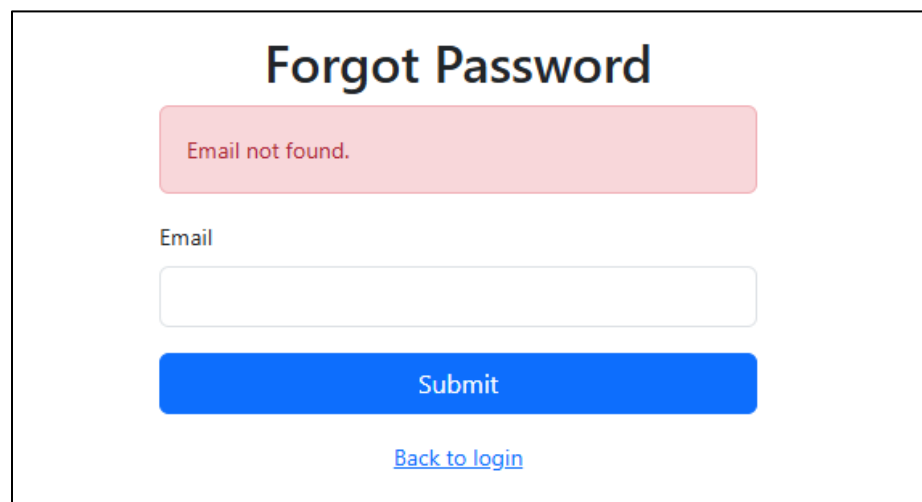


The screenshot shows a web form titled "Forgot Password". Below the title is a label "Email" followed by a text input field. Under the input field is a prominent blue button labeled "Submit". At the bottom of the form is a blue hyperlink labeled "Back to login".

Hình 3.2.2a Giao diện trang quên mật khẩu

2. Hệ thống kiểm tra email:

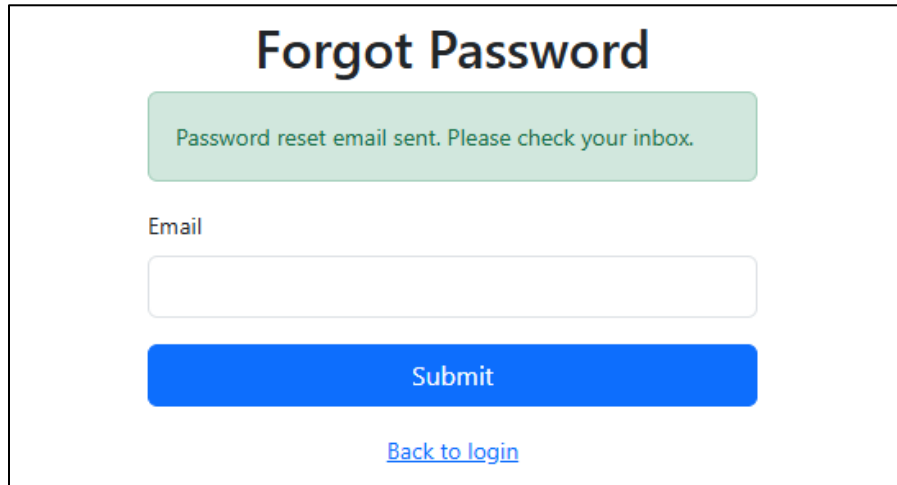
- Kiểm tra email có tồn tại trong cơ sở dữ liệu không.
- Nếu không tồn tại: Thông báo lỗi.



This screenshot shows the same "Forgot Password" form as in Figure 3.2.2a, but with an error message. A red rectangular box at the top of the form contains the text "Email not found." in red. Below this, the "Email" label, input field, "Submit" button, and "Back to login" link are visible, identical to the previous figure.

Hình 3.2.2b Thông báo lỗi khi email không tồn tại

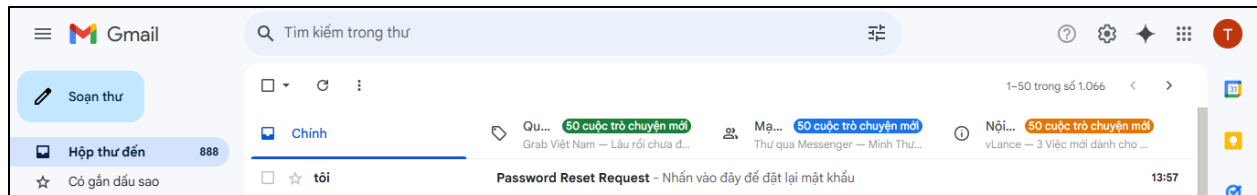
- Nếu tồn tại: Tạo token bảo mật (resetPasswordToken) kèm thời gian hết hạn.



Hình 3.2.2c Thông báo khi email tồn tại

3. Gửi email xác nhận:

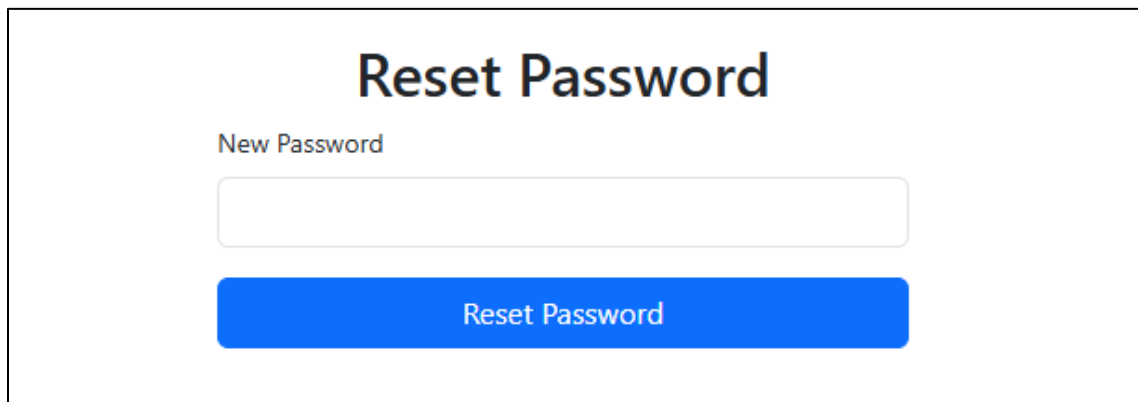
- Gửi email chứa liên kết đặt lại mật khẩu



Hình 3.2.2d Email đặt lại mật khẩu

4. Người dùng nhấp vào liên kết:

- Hệ thống xác minh token:
 - Nếu hợp lệ: Hiển thị form nhập mật khẩu mới.



Hình 3.2.2e Giao diện trang đặt lại mật khẩu

- Nếu không hợp lệ hoặc token hết hạn: Thông báo lỗi.

5. Đặt lại mật khẩu:

- Người dùng nhập mật khẩu mới.
- Hệ thống mã hóa mật khẩu và lưu vào cơ sở dữ liệu.
- Xóa token đặt lại mật khẩu.

3.2.3. Chức năng thêm mới người dùng

The screenshot shows a web application interface for adding a new user. The header includes 'Quản Lý Người Dùng' and a search bar. The main content area is titled 'Thêm Người Dùng'. It contains several input fields: 'First Name' (Kieu), 'Last Name' (Anh), 'Telephone' (09818116841), and 'Email' (kieuanh@gmail.com). There is a 'Customer Details' section with a text area containing 'Cao đẳng Công nghệ Thông tin TPHCM'. Below this is a 'Hình Ảnh' (Image) section with a file upload button and a selected file 'anh-dep-44.jpg.webp'. A blue 'Add Customer' button is at the bottom.

Hình 3.2.2a Nhập thông tin người dùng mới

The screenshot shows the 'Trang Chủ' (Home) page of the web application. A green success message 'User added successfully!' is displayed at the top. Below it is a table listing users. The table has columns for 'Image', 'First Name', 'Last Name', 'Email', and 'Action'. The bottom of the page shows pagination controls: 'First 1 Last'.

Image	First Name	Last Name	Email	Action
	Kieu	Anh	kieuanh@gmail.com	
	Thai	Hoa	thaihoa300405@gmail.com	
	Nhi	Nguyen	ttn2763@gmail.com	
	The	Man	totheman@gmail.com	

Hình 3.2.2b Thông báo thêm người dùng thành công

3.2.4. Chức năng xem chi tiết người dùng

Quản Lý Người Dùng


Tim Kiếm ...

Đăng Xuất

[Trang Chủ](#)

The Man

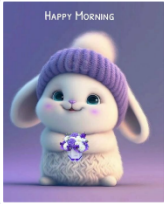
ShareExport



[Trang Chủ](#) / The

Lần Cuối Cập Nhật: Thu, 23 Jan 2025 16:39:53 GMT Mã Người Dùng: 678736aa3e74e78311a421bc

Image:



Name:

The Man

Tel:

079138108

Email:

totheman@gmail.com

Details:

Tôi là Mẫn

Date Created:

2025-01-15 11:16:38

Date Modified:

2025-01-23 23:39:53

Hình 3.2.3 Giao diện xem chi tiết người dùng

3.2.5. Chức năng sửa thông tin người dùng

Quản Lý Người Dùng

Tim Kiếm ...

Đăng Xuất

[Trang Chủ](#)

Editing: The Man

?

[Trang Chủ](#) / The Man

Lần Cuối Cập Nhật: Thu, 23 Jan 2025 16:39:53 GMT Mã Người Dùng: 678736aa3e74e78311a421bc

First Name

The

Last Name

Man

Telephone

079138108

Email

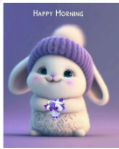
totheman@gmail.com

Customer Details

Tôi là Mẫn

Cập Nhật Hình Ảnh

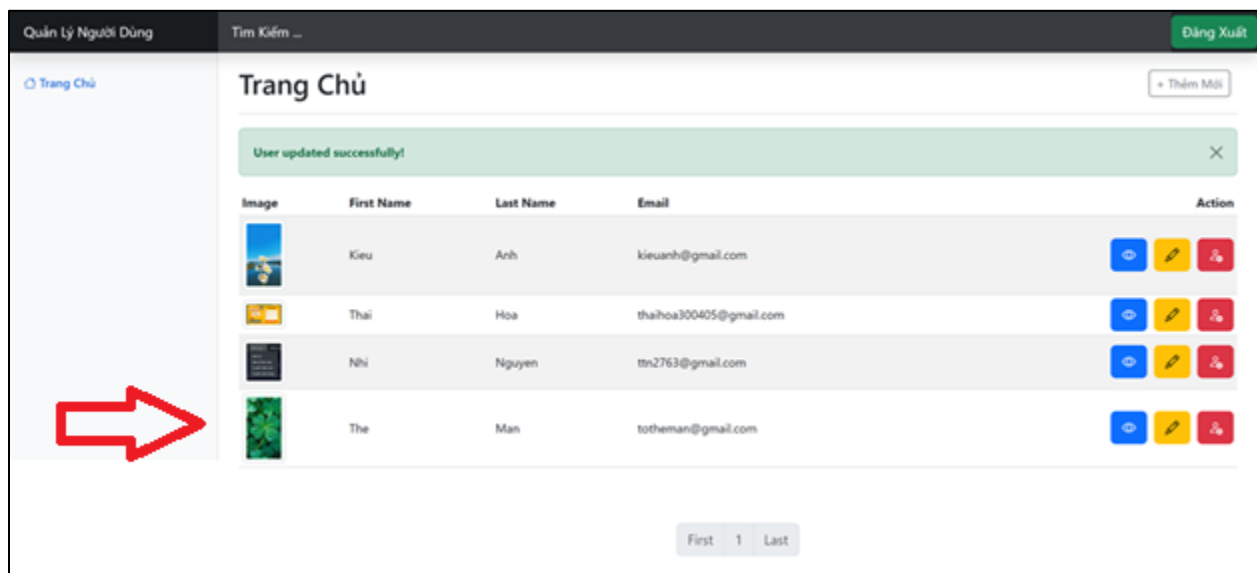
Chọn tệp hình-anh-may-man-trong-thi-cu-1.jpg



Update Customer

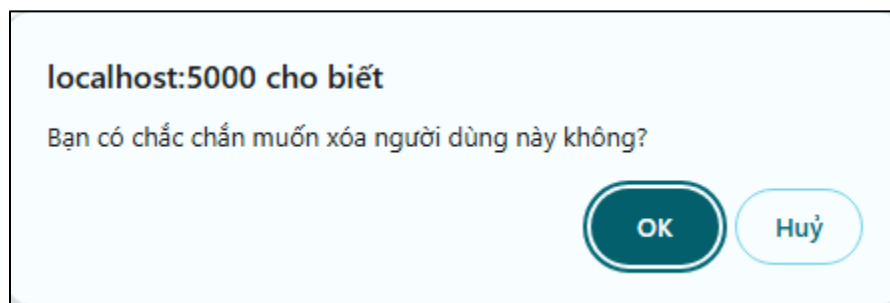
Delete Customer

Hình 3.2.4a Cập nhật hình ảnh người dùng



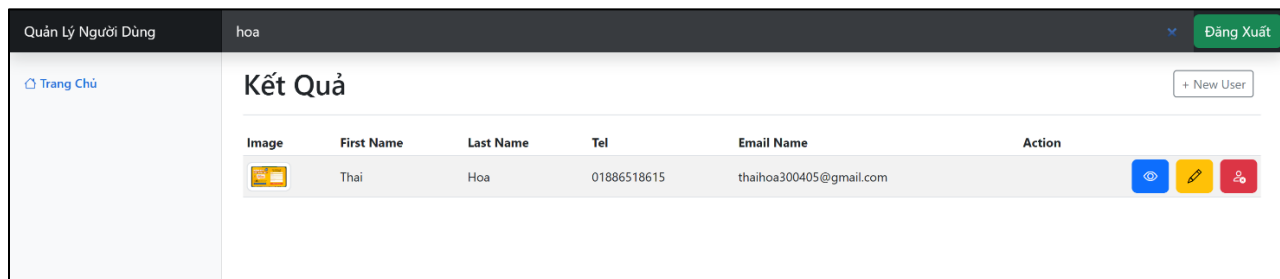
Hình 3.2.4b Thông báo thay đổi thành công

3.2.6. Chức năng xóa người dùng



Hình 3.2.5 Thông báo khi muốn xóa người dùng

3.2.7. Chức năng tìm kiếm



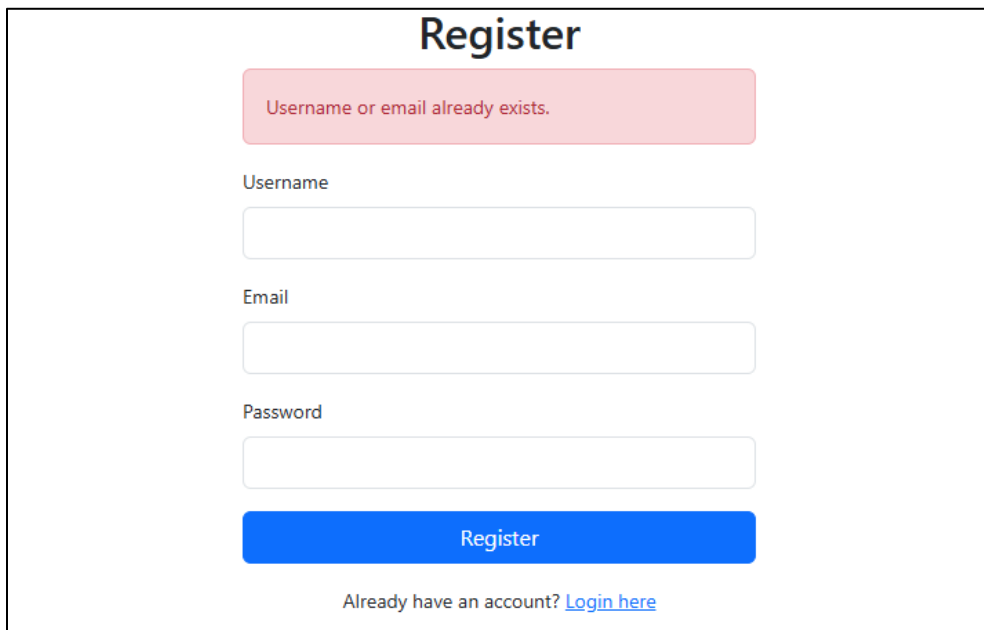
Hình 3.2.6 Tìm kiếm người dùng theo tên

Chương 4. XỬ LÝ LỖI VÀ BẢO MẬT

4.1. Xử lý lỗi trong ứng dụng

4.1.1. Lỗi 400 – Bad Request

- ❖ Nguyên nhân:
 - Dữ liệu đầu vào từ client không hợp lệ (ví dụ: thiếu email, password không đúng định dạng).
 - Thông tin tài khoản đã tồn tại (username hoặc email trùng).
- ❖ Cách xử lý:
 - Phản hồi lỗi với HTTP status code 400 kèm thông báo chi tiết: *"Username or email already exists."*



The image shows a web form titled "Register". At the top, there is a red error message box that says "Username or email already exists." Below this, there are three input fields labeled "Username", "Email", and "Password". At the bottom of the form is a blue button labeled "Register". Below the button, there is a link that says "Already have an account? [Login here](#)".

Hình 4.1.1 Thông báo lỗi 400

4.1.2. Lỗi 401 – Unauthorized

- ❖ Nguyên nhân:
 - Người dùng không xác thực (chưa đăng nhập hoặc token không hợp lệ).
 - Người dùng chưa xác minh email trước khi đăng nhập.
- ❖ Cách xử lý:
 - Xác thực thông qua JWT hoặc kiểm tra isVerified trong cơ sở dữ liệu.

- Phản hồi với HTTP status code 401 kèm thông báo chi tiết:
 - *"Invalid email or password."*
 - *"Please verify your email before logging in.""*

Login

Invalid email or password.

Email

Password

Login

Don't have an account? [Register here](#)

Login

Please verify your email before logging in.

Email

Password

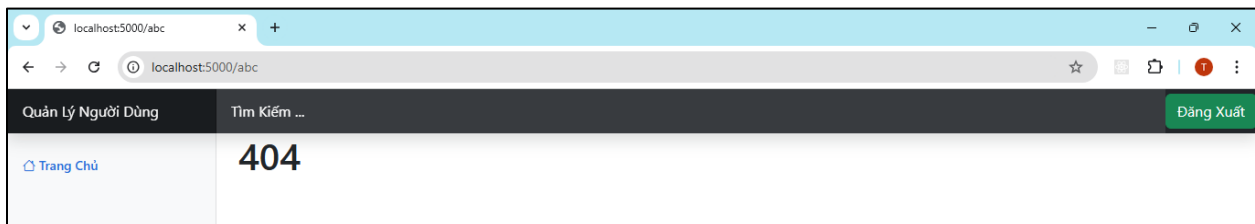
Login

Don't have an account? [Register here](#)

Hình 4.1.2 Thông báo lỗi 401

4.1.3. Lỗi 404 – Not Found

- ❖ Nguyên nhân:
 - Không tìm thấy người dùng dựa trên token hoặc ID trong cơ sở dữ liệu.
 - Truy cập một route không tồn tại.
- ❖ Cách xử lý:
 - Kiểm tra sự tồn tại của tài nguyên trước khi thực hiện logic.
 - Phản hồi với HTTP status code 404 và thông báo như: *"User not found. Please register again."*



Login

User not found. Please register again.

Email

Password

Login

Don't have an account? [Register here](#)

Hình 4.1.3 Thông báo lỗi 404

4.1.4. Lỗi 500 – Internal Server Error

❖ Nguyên nhân:

- Lỗi logic trong ứng dụng (ví dụ: lưu người dùng thất bại).
- Lỗi không mong đợi khi gửi email hoặc xử lý token.

❖ Cách xử lý:

- Sử dụng khối try-catch để bắt lỗi và ngăn ứng dụng bị crash.
- Ghi log chi tiết để phục vụ việc gỡ lỗi.
- Phản hồi với HTTP status code 500 và thông báo thân thiện: *"An unexpected error occurred. Please try again later."*

Register

An unexpected error occurred. Please try again later.

Username

Email

Password

Register

Already have an account? [Login here](#)

Hình 4.1.4 Thông báo lỗi 500

4.2. Bảo mật ứng dụng

4.2.1. Mã hóa mật khẩu

- Mục đích: Bảo mật thông tin nhạy cảm như mật khẩu để tránh trường hợp bị đánh cắp hoặc lộ lọt.
- Phương pháp:
 - Sử dụng thư viện bcryptjs để mã hóa mật khẩu trước khi lưu vào cơ sở dữ liệu.
 - Khi người dùng đăng nhập, mật khẩu được nhập sẽ được so sánh với mật khẩu đã được mã hóa trong cơ sở dữ liệu thay vì lưu trữ mật khẩu dạng thuần văn bản.
- Lợi ích: Ngay cả khi dữ liệu cơ sở dữ liệu bị rò rỉ, kẻ tấn công không thể sử dụng trực tiếp các mật khẩu được mã hóa.

4.2.2. Bảo vệ route

- Mục đích: Ngăn người dùng không xác thực hoặc không có quyền truy cập vào các route bảo mật.
- Phương pháp:

- Sử dụng JWT (JSON Web Token) để xác thực danh tính người dùng.
 - Mỗi khi người dùng đăng nhập thành công, server tạo một JWT chứa thông tin người dùng và gửi lại phía client.
 - Middleware được sử dụng để kiểm tra JWT trong mỗi yêu cầu tới các route bảo mật. Nếu JWT không hợp lệ, yêu cầu sẽ bị từ chối.
- Lợi ích:
- Cung cấp cách xác thực hiệu quả mà không cần lưu trữ trạng thái trên server (stateless authentication).
 - Giảm nguy cơ các yêu cầu không hợp lệ truy cập vào hệ thống.

Chương 5. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1. Kết luận

❖ Kết luận đạt được

- Đã xây dựng hoàn thiện Website quản lý người dùng với đầy đủ các tính năng cơ bản.
- Đáp ứng được các yêu cầu cơ bản của một hệ thống quản lý người dùng.
- Hệ thống API được thiết kế rõ ràng, dễ sử dụng và dễ mở rộng trong tương lai.
- Đã ứng dụng thành công các công nghệ hiện đại như Node.js, Express.js, MongoDB.
- Gửi email tự động qua Nodemailer, hỗ trợ xác thực và thông báo đến người dùng.
- Nhóm đã vận dụng và củng cố các kiến thức đã học, đồng thời rèn luyện kỹ năng phát triển dự án thực tế.

❖ Lý thuyết, kỹ năng và kinh nghiệm

- Biết cách áp dụng các công nghệ backend và sử dụng thành thạo Visual Studio Code để lập trình.
- Hiểu rõ quy trình xây dựng một hệ thống backend quản lý người dùng, từ thiết kế kiến trúc đến triển khai các chức năng.
- Nâng cao kiến thức về các công nghệ như Node.js, Express.js, JWT, MongoDB.
- Hiểu thêm việc sử dụng các thư viện và công cụ quan trọng như:
 - Mongoose: Quản lý cơ sở dữ liệu MongoDB.
 - JWT Authentication: Xác thực và phân quyền bảo mật.
 - Nodemailer: Gửi email tự động.
- Hiểu rõ cách tổ chức cấu trúc thư mục và tái sử dụng code trong dự án.

❖ Khuyết điểm

- Một số chức năng còn chưa tối ưu hiệu suất, tốc độ xử lý vẫn còn chậm với dữ liệu lớn.
- Chức năng quản lý chưa đa dạng, chưa có phân quyền người dùng chi tiết.
- Chưa triển khai website lên môi trường thực tế trên Internet, hiện chỉ dừng ở môi trường phát triển cục bộ.
- Cơ sở dữ liệu và code còn cần cải thiện để đạt hiệu quả tốt hơn.

- Độ bảo mật của hệ thống chưa cao, chưa triển khai đầy đủ các biện pháp phòng chống tấn công.

❖ Kinh nghiệm đúc kết

- Cần tăng cường khả năng tự học và nắm bắt các công nghệ mới trong lĩnh vực phát triển web.
- Tăng cường tư duy logic và kỹ năng giải quyết vấn đề.
- Cải thiện kiến thức chuyên môn về lập trình backend và quản lý cơ sở dữ liệu.
- Rèn luyện kỹ năng làm việc nhóm và chia sẻ kiến thức trong nhóm để nâng cao hiệu quả làm việc.

5.2. Hướng phát triển tương lai

❖ Phát triển đề tài

- Thêm chức năng phân quyền người dùng
- Mở rộng chức năng quản lý: Thêm tính năng quản lý chi tiết hơn như nhóm người dùng, lịch sử hoạt động.
- Tích hợp API bên thứ ba: Sử dụng các API để hỗ trợ các quy trình nâng cao như đồng bộ hóa dữ liệu hoặc tích hợp dịch vụ thanh toán.
- Tích hợp Mail Server: Xây dựng tính năng gửi email tự động như thông báo quan trọng, xác nhận tài khoản hoặc đặt lại mật khẩu.

❖ Phát triển bản thân

- Tăng cường khả năng tự học và nắm bắt các công nghệ mới.
- Phát triển khả năng tư duy logic và giải quyết các vấn đề phức tạp trong dự án.
- Cải thiện kiến thức chuyên môn về bảo mật hệ thống và tối ưu hóa hiệu suất.
- Tham gia các dự án thực tế để rèn luyện kỹ năng làm việc nhóm và quản lý dự án hiệu quả.

DANH MỤC TÀI LIỆU THAM KHẢO

- [1] <https://github.com/nodejs>
- [2] <https://github.com/itssubhodiprooy/User-Management-System>
- [3] <https://github.com/DevikaRajesh22/UserManagement>
- [4] <https://nodejs.org/en>
- [5] <https://select2.org/>
- [6] <https://stackoverflow.com/>
- [7] <https://vi.wikipedia.org/>
- [8] <https://www.w3schools.com/>