**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÁO CÁO HỌC PHẦN**

**TÊN HỌC PHẦN: AN TOÀN BẢO MẬT THÔNG TIN**

**ĐỀ TÀI: ỨNG DỤNG TRIPLE DES VÀ AES ĐỂ BẢO VỆ THÔNG TIN NHẠY CẢM TRONG CƠ SỞ DỮ LIỆU**

**Giảng viên hướng dẫn: T.S Trần Đăng Công**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Mã Sinh Viên | Họ và Tên | Ngày Sinh | Lớp |
| 1 | 1771020601 | Phạm Ngọc Sơn | 29/03/2005 | CNTT 17-06 |
| 2 | 1771020540 | Bùi Thiên Phú | 21/07/2005 | CNTT 17-06 |
| 3 | 1771020550 | Nguyễn Văn Phước | 25/09/2005 | CNTT 17-06 |

### 

**Hà Nội, năm 2025**

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÁO CÁO HỌC PHẦN**

**TÊN HỌC PHẦN: AN TOÀN BẢO MẬT THÔNG TIN**

**ĐỀ TÀI: ỨNG DỤNG TRIPLE DES VÀ AES ĐỂ BẢO VỆ THÔNG TIN NHẠY CẢM TRONG CƠ SỞ DỮ LIỆU**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Mã Sinh Viên | Họ và Tên | Ngày Sinh | Điểm | |
| Bằng Số | Bằng Chữ |
| 1 | 1771020601 | Phạm Ngọc Sơn | 29/03/2005 |  |  |
| 2 | 1771020540 | Bùi Thiên Phú | 21/07/2005 |  |  |
| 3 | 1771020550 | Nguyễn Văn Phước | 25/09/2005 |  |  |

### 

### CÁN BỘ CHẤM THI 1 CÁN BỘ CHẤM THI 2

**Hà Nội, năm 2025**

**MỤC LỤC**

[**LỜI MỞ ĐẦU** 5](#_Toc202289354)

[**CHƯƠNG 1. GIỚI THIỆU CHUNG** 6](#_Toc202289355)

[**1.1. Lý do chọn đề tài** 6](#_Toc202289356)

[**1.2. Mục tiêu và phạm vi đề tài** 7](#_Toc202289357)

[**1.3. Ý nghĩa thực tiễn của đề tài** 7](#_Toc202289358)

[**CHƯƠNG 2. CƠ SỞ LÝ THUYẾT** 9](#_Toc202289359)

[**2.1. Tổng quan về bảo mật dữ liệu** 9](#_Toc202289360)

[**2.2. Mã hóa đối xứng** 10](#_Toc202289361)

[**2.3. Thuật toán Triple DES** 10](#_Toc202289362)

[**2.4. Thuật toán AES** 11](#_Toc202289363)

[**2.5. So sánh Triple DES và AES** 12](#_Toc202289364)

[**2.6. Ứng dụng thực tế của mã hóa đối xứng** 13](#_Toc202289365)

[**CHƯƠNG 3. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG** 14](#_Toc202289366)

[**3.1. Phân tích yêu cầu chức năng** 14](#_Toc202289367)

[**3.2. Phân tích yêu cầu bảo mật** 14](#_Toc202289368)

[**3.3. Thiết kế quy trình mã hóa/giải mã** 15](#_Toc202289369)

[**3.4. Thiết kế giao diện** 15](#_Toc202289370)

[**3.5. Thiết kế lưu trữ dữ liệu** 16](#_Toc202289371)

[**CHƯƠNG 4: CÀI ĐẶT VÀ GIẢI THÍCH MÃ NGUỒN** 18](#_Toc202289372)

[**4.1. Cấu trúc thư mục và các file chính** 18](#_Toc202289373)

[**4.2. Cài đặt thuật toán mã hóa/giải mã** 19](#_Toc202289374)

[**4.3. Xử lý đăng ký, đăng nhập, phân quyền** 21](#_Toc202289375)

[**4.4. Ghi log truy cập** 24](#_Toc202289376)

[**4.5. Giao diện người dùng và quản trị** 25](#_Toc202289377)

[**CHƯƠNG 5. THỬ NGHIỆM VÀ ĐÁNH GIÁ** 27](#_Toc202289378)

[**5.1. Kịch bản kiểm thử chức năng** 27](#_Toc202289379)

[5.2. Kịch bản kiểm thử bảo mật 29](#_Toc202289380)

[5.3. Kết quả thử nghiệm 29](#_Toc202289381)

[5.4. Đánh giá hiệu quả bảo mật và hiệu suất 30](#_Toc202289382)

[**CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 31](#_Toc202289383)

[**6.1. Kết luận** 31](#_Toc202289384)

[**6.2. Hạn chế** 31](#_Toc202289385)

[**6.3. Đề xuất cải tiến** 32](#_Toc202289386)

[**TÀI LIỆU THAM KHẢO** 33](#_Toc202289387)

**LỜI MỞ ĐẦU**

Trong kỷ nguyên chuyển đổi số, dữ liệu cá nhân như số CCCD, tài khoản ngân hàng và số bảo hiểm xã hội đã trở thành mục tiêu hàng đầu của các cuộc tấn công mạng tinh vi. Mỗi lỗ hổng bảo mật đều có thể dẫn đến hậu quả nghiêm trọng về tài chính, uy tín và pháp lý cho cá nhân cũng như tổ chức.

Mã hóa đối xứng – điển hình là Triple DES và AES – là hàng rào bảo vệ hiệu quả, đảm bảo tính bí mật và toàn vẹn của dữ liệu trước khi lưu trữ và truyền tải. Triple DES tận dụng sự ổn định của DES truyền thống, trong khi AES mang lại tốc độ xử lý cao và độ an toàn được công nhận toàn cầu.

Đề tài “Ứng dụng Triple DES và AES để bảo vệ thông tin nhạy cảm trong cơ sở dữ liệu” tập trung xây dựng một hệ thống web đơn giản, cho phép người dùng đăng ký, đăng nhập, đồng thời cung cấp giao diện cho quản trị viên giải mã và giám sát. Trong đó, những trường dữ liệu như số CCCD được mã hóa bằng Triple DES còn địa chỉ và số tài khoản ngân hàng sử dụng AES nhằm tối ưu hiệu năng.

Bên cạnh cơ chế mã hóa và giải mã, hệ thống còn ghi nhận đầy đủ lịch sử truy cập và thao tác với dữ liệu nhạy cảm, hỗ trợ việc kiểm tra, giám sát một cách minh bạch. Đây không chỉ là cơ hội để sinh viên áp dụng kiến thức lý thuyết vào thực tiễn, mà còn cung cấp giải pháp tham khảo cho các ứng dụng quản lý dữ liệu nhạy cảm trong môi trường số.

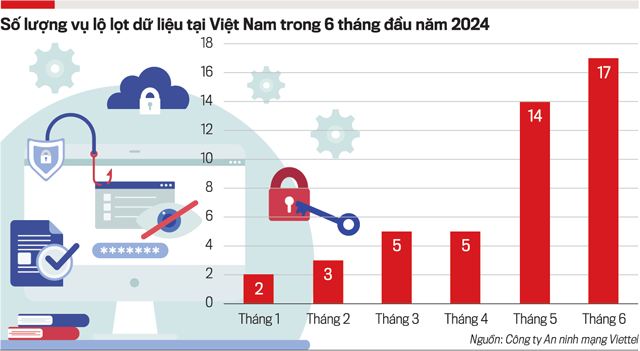
# **CHƯƠNG 1. GIỚI THIỆU CHUNG**

## **1.1. Lý do chọn đề tài**

Trong thời đại công nghệ thông tin phát triển mạnh mẽ, dữ liệu cá nhân ngày càng trở thành tài sản quý giá của mỗi cá nhân, tổ chức và doanh nghiệp. Việc lưu trữ, xử lý và bảo vệ thông tin nhạy cảm như số căn cước công dân (CCCD), số tài khoản ngân hàng, địa chỉ, số bảo hiểm xã hội… là một yêu cầu bắt buộc đối với mọi hệ thống thông tin hiện đại. Tuy nhiên, thực tế cho thấy các vụ rò rỉ, đánh cắp dữ liệu cá nhân vẫn liên tục xảy ra, gây ra nhiều hậu quả nghiêm trọng về tài chính, uy tín và an toàn cá nhân.

Theo báo cáo của các tổ chức an ninh mạng, số lượng các vụ tấn công nhằm vào dữ liệu cá nhân tại Việt Nam và trên thế giới đều tăng mạnh qua từng năm. Đặc biệt, các thông tin như số CCCD, số tài khoản ngân hàng, địa chỉ… nếu bị khai thác trái phép sẽ gây hậu quả nghiêm trọng, không chỉ cho cá nhân mà còn ảnh hưởng đến cả tổ chức, doanh nghiệp và xã hội. Các hình thức tấn công phổ biến như tấn công brute-force, phishing, man-in-the-middle, rò rỉ dữ liệu nội bộ, phần mềm độc hại… ngày càng tinh vi và khó kiểm soát.

Bên cạnh đó, trong chương trình đào tạo ngành Công nghệ thông tin, việc học tập và thực hành các kỹ thuật bảo mật, đặc biệt là mã hóa dữ liệu, là một phần quan trọng giúp sinh viên hiểu rõ hơn về lý thuyết và ứng dụng thực tế của các thuật toán này. Việc xây dựng một hệ thống bảo mật thông tin nhạy cảm sử dụng các thuật toán mã hóa đối xứng như Triple DES và AES không chỉ giúp sinh viên củng cố kiến thức lý thuyết mà còn rèn luyện kỹ năng lập trình, thiết kế hệ thống và đánh giá hiệu quả bảo mật.



Chính vì vậy, nhóm chúng em lựa chọn đề tài: “Ứng dụng Triple DES và AES để bảo vệ thông tin nhạy cảm trong cơ sở dữ liệu” nhằm mục tiêu xây dựng một hệ thống webđơn giản, áp dụng các thuật toán mã hóa hiện đại để bảo vệ dữ liệu người dùng, đồng thời nâng cao nhận thức về an toàn thông tin trong môi trường số.

## **1.2. Mục tiêu và phạm vi đề tài**

Mục tiêu

* Xây dựng một hệ thống web cho phép người dùng đăng ký, đăng nhập và quản trị viên xem thông tin người dùng.
* Áp dụng các thuật toán mã hóa đối xứng (Triple DES, AES) để bảo vệ các trường dữ liệu nhạy cảm như số CCCD, địa chỉ, số tài khoản ngân hàng trước khi lưu trữ vào cơ sở dữ liệu.
* Đảm bảo chỉ người dùng hợp lệ hoặc quản trị viên có quyền mới có thể giải mã và truy xuất dữ liệu.
* Ghi log truy cập, thao tác với dữ liệu nhạy cảm để phục vụ kiểm tra, giám sát.
* Đánh giá hiệu quả bảo mật và hiệu suất của hệ thống thông qua các thử nghiệm thực tế.

Phạm vi

* Hệ thống sử dụng file *users.json* làm cơ sở dữ liệu, phù hợp cho bài tập lớn cấp sinh viên, không triển khai trên hệ quản trị cơ sở dữ liệu thực thụ.
* Chỉ tập trung vào bảo mật các trường dữ liệu nhạy cảm, chưa triển khai các tính năng bảo mật nâng cao như xác thực đa yếu tố, chữ ký số.
* Giao diện đơn giản, dễ sử dụng, phù hợp cho mục đích học tập và trình diễn.

## **1.3. Ý nghĩa thực tiễn của đề tài**

* Góp phần nâng cao nhận thức về bảo mật thông tin cá nhân trong môi trường số.
* Ứng dụng thực tế trong các hệ thống quản lý thông tin, ngân hàng, y tế, giáo dục, nơi dữ liệu nhạy cảm cần được bảo vệ nghiêm ngặt.
* Là nền tảng để phát triển các hệ thống bảo mật phức tạp hơn trong tương lai, như tích hợp xác thực đa yếu tố, sử dụng cơ sở dữ liệu lớn, hoặc triển khai trên môi trường thực tế.

**1.4. Phương pháp nghiên cứu**

* Nghiên cứu lý thuyết về các thuật toán mã hóa đối xứng, đặc biệt là Triple DES và AES.
* Tìm hiểu các mô hình bảo mật dữ liệu trong thực tế.
* Thiết kế và xây dựng hệ thống web sử dụng Node.js, Express, lưu trữ dữ liệu bằng file JSON.
* Cài đặt các hàm mã hóa/giải mã, tích hợp vào quy trình xử lý dữ liệu người dùng.
* Thử nghiệm hệ thống với các trường hợp thực tế, đánh giá hiệu quả bảo mật và hiệu suất.

# **CHƯƠNG 2. CƠ SỞ LÝ THUYẾT**

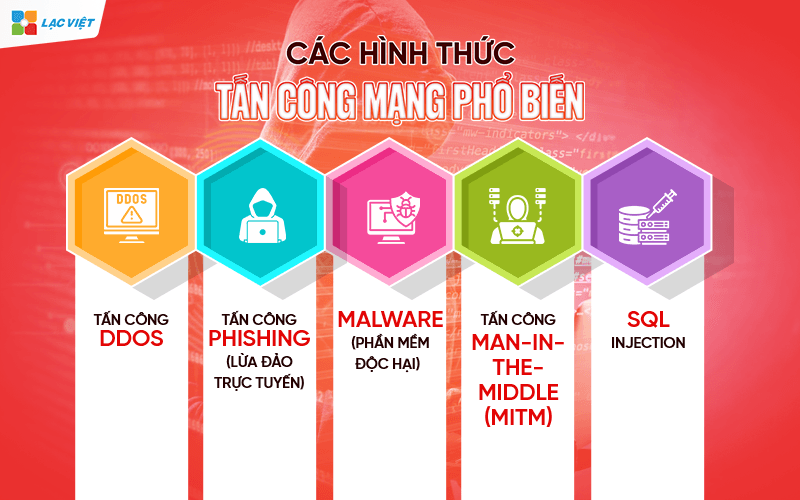
## **2.1. Tổng quan về bảo mật dữ liệu**

Bảo mật dữ liệu là tập hợp các biện pháp kỹ thuật và quy trình nhằm bảo vệ dữ liệu khỏi các truy cập trái phép, đảm bảo tính riêng tư, toàn vẹn và sẵn sàng của thông tin. Trong các hệ thống thông tin hiện đại, bảo mật dữ liệu đóng vai trò then chốt, giúp ngăn chặn các nguy cơ rò rỉ, đánh cắp hoặc thay đổi dữ liệu nhạy cảm.

Các nguyên tắc cơ bản của bảo mật dữ liệu bao gồm:

* Bảo mật (Confidentiality): Đảm bảo chỉ những người được phép mới có thể truy cập thông tin.
* Toàn vẹn (Integrity): Đảm bảo thông tin không bị thay đổi, chỉnh sửa trái phép.
* Sẵn sàng (Availability): Đảm bảo thông tin luôn sẵn sàng khi người dùng hợp lệ cần truy cập.

Các mối đe dọa phổ biến đối với bảo mật dữ liệu gồm: tấn công brute-force, phishing, man-in-the-middle, rò rỉ dữ liệu nội bộ, phần mềm độc hại…



Vai trò của mã hóa trong bảo mật dữ liệu

Mã hóa là một trong những biện pháp quan trọng nhất để bảo vệ dữ liệu. Khi dữ liệu được mã hóa, kể cả khi bị đánh cắp, kẻ tấn công cũng không thể đọc được nội dung nếu không có khóa giải mã. Mã hóa được sử dụng rộng rãi trong bảo vệ dữ liệu cá nhân, giao dịch ngân hàng, truyền thông an toàn, lưu trữ dữ liệu trên đám mây…

## **2.2. Mã hóa đối xứng**

Mã hóa đối xứng là phương pháp sử dụng cùng một khóa để mã hóa và giải mã dữ liệu. Đây là phương pháp mã hóa phổ biến nhờ tốc độ nhanh, dễ triển khai và phù hợp với các hệ thống cần xử lý lượng lớn dữ liệu.

Ưu điểm:

* Tốc độ mã hóa/giải mã nhanh.
* Dễ dàng triển khai trong các hệ thống thực tế.

Nhược điểm:

* Khó khăn trong việc quản lý và phân phối khóa an toàn.
* Không phù hợp cho các hệ thống truyền thông công khai nếu không có biện pháp bảo vệ khóa.

Ứng dụng thực tế:

Mã hóa file, bảo vệ dữ liệu truyền nội bộ, mã hóa ổ cứng, bảo vệ dữ liệu cá nhân trong các hệ thống quản lý.

Quy trình mã hóa đối xứng

1. Người gửi và người nhận thống nhất một khóa bí mật.
2. Dữ liệu gốc được mã hóa bằng khóa này trước khi lưu trữ hoặc truyền đi.
3. Khi cần truy xuất, dữ liệu được giải mã bằng chính khóa đã dùng để mã hóa.

## **2.3. Thuật toán Triple DES**

Triple DES (3DES) là phiên bản nâng cao của thuật toán DES, được phát triển nhằm khắc phục các điểm yếu về bảo mật của DES. Triple DES thực hiện ba lần mã hóa liên tiếp với ba khóa khác nhau, giúp tăng cường độ an toàn của dữ liệu.

* **Nguyên lý hoạt động:**  
  Dữ liệu được mã hóa bằng khóa thứ nhất, giải mã bằng khóa thứ hai, rồi lại mã hóa bằng khóa thứ ba.
* **Độ dài khóa:**  
  112 hoặc 168 bit (tùy cấu hình).
* **Số vòng lặp:**  
  48 vòng (3 lần 16 vòng của DES).
* **Ưu điểm:**  
  Bảo mật tốt hơn DES, dễ triển khai trên nền tảng cũ.
* **Nhược điểm:**  
  Tốc độ chậm hơn AES, dần bị thay thế bởi các thuật toán mới.
* **Ứng dụng:**  
  Ngân hàng, hệ thống tài chính, các hệ thống cũ cần nâng cấp bảo mật.

**Ví dụ mã hóa với Triple DES**  
 Giả sử dữ liệu gốc là số CCCD: 012345678901  
 Khóa 1: key1, Khóa 2: key2, Khóa 3: key3  
**Quy trình:**

1. Mã hóa lần 1: E1 = DES\_Encrypt(data, key1)
2. Giải mã lần 2: D2 = DES\_Decrypt(E1, key2)
3. Mã hóa lần 3: E3 = DES\_Encrypt(D2, key3)  
   Kết quả cuối cùng là chuỗi mã hóa lưu vào cơ sở dữ liệu.

## **2.4. Thuật toán AES**

AES (Advanced Encryption Standard) là thuật toán mã hóa khối đối xứng hiện đại, được NIST công nhận là tiêu chuẩn mã hóa vào năm 2001. AES sử dụng các khóa có độ dài 128, 192 hoặc 256 bit, với số vòng lặp tương ứng là 10, 12 hoặc 14.

* **Nguyên lý hoạt động:**  
  Dữ liệu được chia thành các khối 128 bit, trải qua nhiều vòng biến đổi (substitution, permutation, mixing, key addition).
* **Ưu điểm:**  
  Tốc độ nhanh, bảo mật cao, tiêu chuẩn quốc tế, được sử dụng rộng rãi.
* **Nhược điểm:**  
  Thuật toán phức tạp hơn DES/3DES, yêu cầu phần cứng mạnh hơn nếu xử lý lượng lớn dữ liệu.
* **Ứng dụng:**  
  Bảo mật web, VPN, bảo vệ file, ngân hàng, chính phủ, thương mại điện tử…

**Ví dụ mã hóa với AES**  
 Giả sử dữ liệu gốc là số tài khoản: 1234567890  
 Khóa AES: key\_aes\_256  
**Quy trình:**

1. Dữ liệu được chuyển thành khối 128 bit.
2. Thực hiện 10 vòng biến đổi (với khóa 128 bit).
3. Kết quả là chuỗi mã hóa lưu vào cơ sở dữ liệu.

## **2.5. So sánh Triple DES và AES**

|  |  |  |
| --- | --- | --- |
| **Tiêu chí** | **Triple DES** | **AES** |
| Độ dài khóa | 112/168 bit | 128/192/256 bit |
| Số vòng lặp | 48 | 10/12/14 |
| Tốc độ | Chậm hơn | Nhanh hơn |
| Bảo mật | Thấp hơn AES | Cao hơn |
| Ứng dụng | Hệ thống cũ, ngân hàng | Hệ thống hiện đại, thương mại điện tử |

Nhận xét: AES vượt trội về tốc độ và bảo mật, phù hợp với các hệ thống hiện đại. Triple DES vẫn được sử dụng trong các hệ thống cũ hoặc nơi yêu cầu tương thích.

## **2.6. Ứng dụng thực tế của mã hóa đối xứng**

Triple DES và AES được sử dụng rộng rãi trong các hệ thống bảo mật thông tin như:

* Ngân hàng: bảo vệ giao dịch, mã hóa dữ liệu khách hàng.
* Chính phủ: bảo vệ tài liệu mật, dữ liệu công dân.
* Doanh nghiệp: bảo vệ dữ liệu nội bộ, thông tin khách hàng.
* Hệ thống quản lý thông tin cá nhân: bảo vệ số CCCD, số tài khoản, địa chỉ…

# **CHƯƠNG 3. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**

## **3.1. Phân tích yêu cầu chức năng**

Hệ thống được xây dựng với các chức năng chính sau:

* Đăng ký tài khoản:  
  Người dùng nhập thông tin cá nhân (họ tên, số CCCD, địa chỉ, số tài khoản, mật khẩu). Các trường nhạy cảm sẽ được mã hóa trước khi lưu vào cơ sở dữ liệu.
* Đăng nhập:  
  Người dùng nhập tên và mật khẩu để xác thực. Nếu đúng, hệ thống giải mã và hiển thị thông tin cá nhân.
* Xem thông tin cá nhân:  
  Sau khi đăng nhập, người dùng có thể xem lại thông tin của mình (đã được giải mã).
* Quản trị viên:  
  Có thể xem danh sách người dùng (ẩn thông tin nhạy cảm mặc định), xem chi tiết thông tin (giải mã khi nhập đúng mật khẩu admin).
* Ghi log truy cập:  
  Hệ thống ghi lại các thao tác đăng ký, đăng nhập, xem chi tiết, ai thao tác, thời gian.

## **3.2. Phân tích yêu cầu bảo mật**

* Mã hóa dữ liệu nhạy cảm:  
  Các trường như số CCCD, địa chỉ, số tài khoản phải được mã hóa bằng Triple DES hoặc AES trước khi lưu vào file users.json.
* Phân quyền truy cập:  
  Chỉ người dùng hợp lệ hoặc quản trị viên mới được giải mã dữ liệu.
* Bảo vệ mật khẩu:  
  (Khuyến nghị: nên hash mật khẩu, tuy nhiên hệ thống hiện tại lưu plain text để đơn giản hóa).
* Ghi log truy cập:  
  Ghi lại mọi thao tác với dữ liệu nhạy cảm để phục vụ kiểm tra, giám sát.

## **3.3. Thiết kế quy trình mã hóa/giải mã**

Khi người dùng đăng ký:

1. Nhập thông tin cá nhân.
2. Áp dụng mã hóa:

* Số CCCD: Triple DES
* Địa chỉ, số tài khoản: AES

1. Lưu vào file users.json với các trường đã mã hóa.

Khi người dùng đăng nhập và xem thông tin:

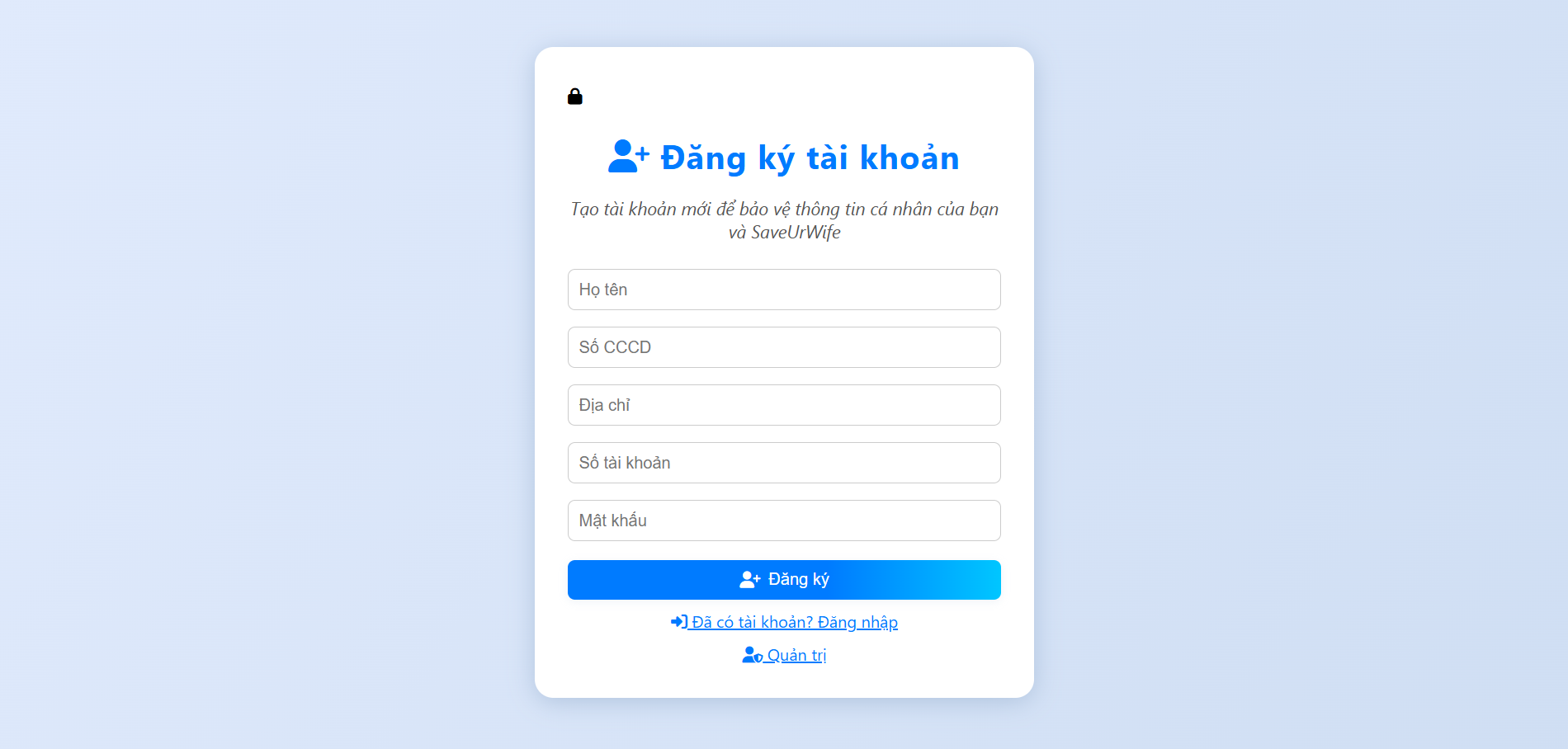
1. Xác thực người dùng.
2. Giải mã các trường nhạy cảm.
3. Hiển thị kết quả trên giao diện.

Khi quản trị viên xem chi tiết:

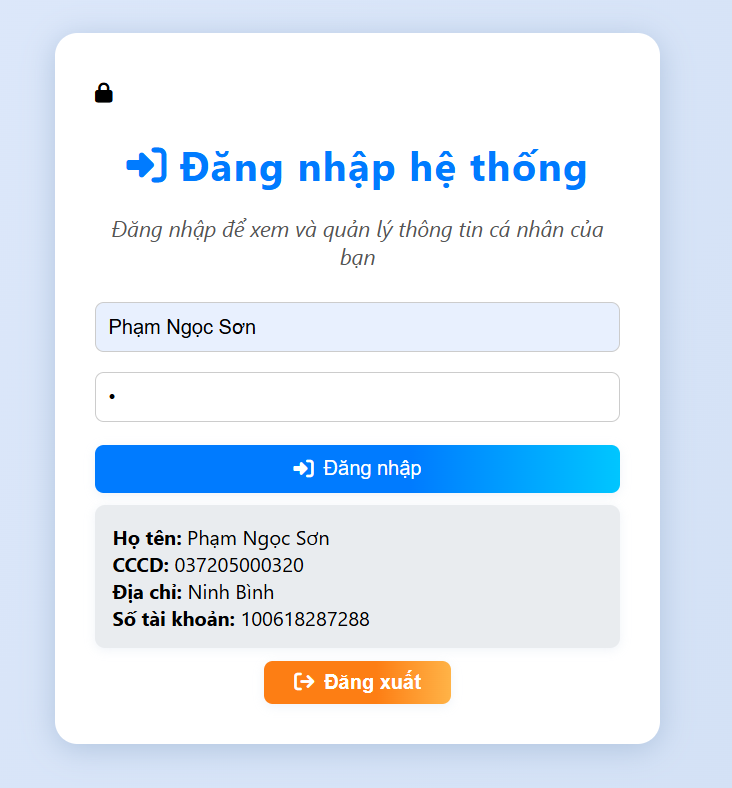
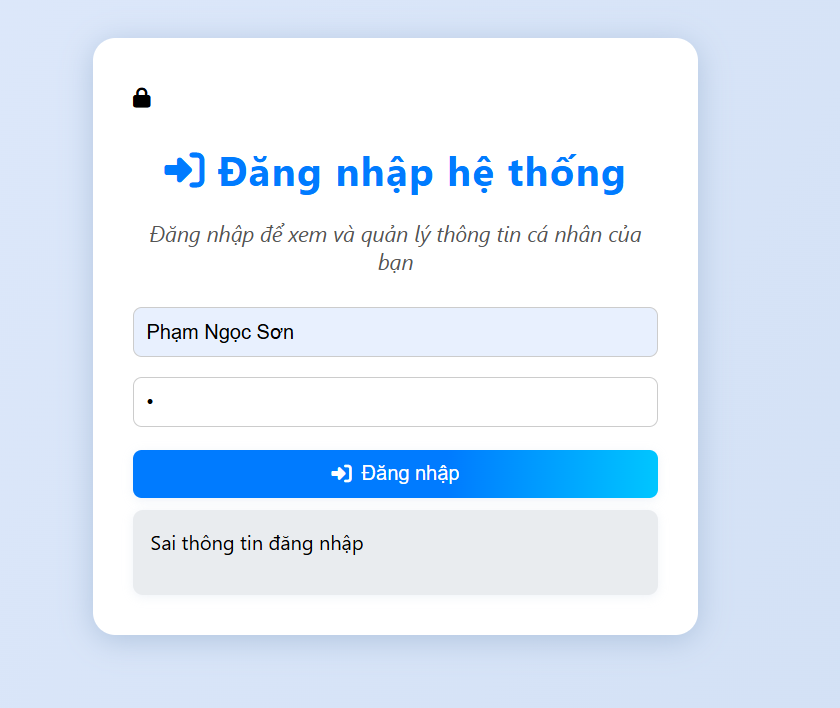
1. Nhập mật khẩu admin.
2. Giải mã các trường nhạy cảm.
3. Hiển thị kết quả.

## **3.4. Thiết kế giao diện**

Giao diện đăng ký/đăng nhập: Form nhập thông tin, nút chuyển đổi giữa hai form.



Giao diện người dùng: Hiển thị thông tin cá nhân sau khi đăng nhập, nếu sai sẽ không hiển thị.

Giao diện quản trị: Bảng danh sách người dùng, nút "Xem chi tiết", log truy cập.

## **3.5. Thiết kế lưu trữ dữ liệu**

- Cơ sở dữ liệu: Sử dụng file users.json để lưu thông tin người dùng.

- Cấu trúc bản ghi:

{

  "name": "Nguyen Van A",

  "password": "123456",

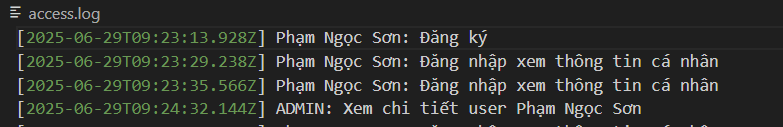
  "cccd": "ENCRYPTED\_STRING",

  "address": "ENCRYPTED\_STRING",

  "bank": "ENCRYPTED\_STRING"

}

- Log truy cập: File access.log ghi lại các thao tác với dữ liệu.



# **CHƯƠNG 4: CÀI ĐẶT VÀ GIẢI THÍCH MÃ NGUỒN**

## **4.1. Cấu trúc thư mục và các file chính**

BTL/

├── controllers/

│   └── userController.js

├── logs/

│   └── access.log

├── models/

│   └── users.json

├── public/

│   ├── admin.html

│   ├── index.html

│   ├── main.js

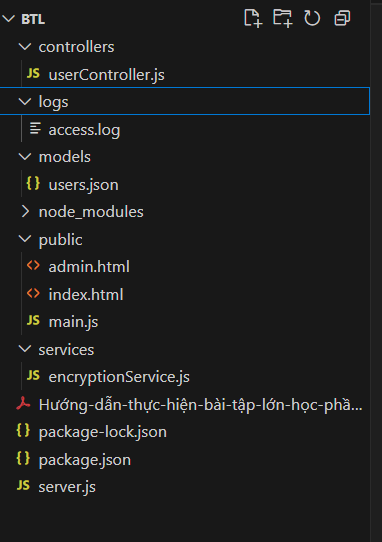
├── services/

│   └── encryptionService.js

├── server.js

├── package.json

* controllers/userController.js: Xử lý logic người dùng, đọc/ghi file, mã hóa/giải mã.
* services/encryptionService.js: Cài đặt các hàm mã hóa/giải mã Triple DES và AES.
* public/index.html, admin.html, main.js: Giao diện và xử lý phía client.
* server.js: Khởi tạo server, định tuyến API.
* models/users.json: Lưu thông tin người dùng.
* logs/access.log: Lưu log truy cập.



## **4.2. Cài đặt thuật toán mã hóa/giải mã**

*File: encryptionService.js*  
Sử dụng thư viện crypto của Node.js để mã hóa/giải mã Triple DES và AES.

const crypto = require('crypto');

const tripleDesKey = Buffer.from('123456789012345678901234', 'utf8'); // 24 bytes

const aesKey = crypto.createHash('sha256').update('my\_aes\_secret').digest();

* Import thư viện mã hóa crypto.
* Tạo khóa 24 byte cho Triple DES.
* Tạo khóa 32 byte cho AES từ chuỗi bí mật.

exports.encryptTripleDES = (text) => {

  const cipher = crypto.createCipheriv('des-ede3', tripleDesKey, null);

  let encrypted = cipher.update(text, 'utf8', 'base64');

  encrypted += cipher.final('base64');

  return encrypted;

};

* Mã hóa chuỗi text bằng Triple DES, trả về chuỗi base64.

exports.decryptTripleDES = (encrypted) => {

  const decipher = crypto.createDecipheriv('des-ede3', tripleDesKey, null);

  let decrypted = decipher.update(encrypted, 'base64', 'utf8');

  decrypted += decipher.final('utf8');

  return decrypted;

};

* Giải mã chuỗi base64 về dữ liệu gốc.

exports.encryptAES = (text) => {

  const iv = crypto.randomBytes(16);

  const cipher = crypto.createCipheriv('aes-256-cbc', aesKey, iv);

  let encrypted = cipher.update(text, 'utf8', 'base64');

  encrypted += cipher.final('base64');

  return iv.toString('base64') + ':' + encrypted;

};

* Mã hóa chuỗi text bằng AES, sinh IV ngẫu nhiên, trả về IV và dữ liệu mã hóa.

exports.decryptAES = (encrypted) => {

  const [ivStr, data] = encrypted.split(':');

  const iv = Buffer.from(ivStr, 'base64');

  const decipher = crypto.createDecipheriv('aes-256-cbc', aesKey, iv);

  let decrypted = decipher.update(data, 'base64', 'utf8');

  decrypted += decipher.final('utf8');

  return decrypted;

};

* Giải mã chuỗi AES: tách IV và dữ liệu, giải mã về chuỗi gốc.

*Giải thích:*

* Các hàm nhận vào dữ liệu và khóa, trả về chuỗi đã mã hóa hoặc giải mã.
* Sử dụng các thuật toán mã hóa đối xứng chuẩn.

## **4.3. Xử lý đăng ký, đăng nhập, phân quyền**

*File: userController.js*

Khi đăng ký: Nhận dữ liệu từ client, mã hóa trường nhạy cảm, lưu vào file.

Khi đăng nhập: Kiểm tra thông tin, giải mã dữ liệu, trả về cho client.

Khi admin xem chi tiết: Yêu cầu nhập mật khẩu admin, giải mã và trả về dữ liệu.

const fs = require('fs');

const path = require('path');

const encryption = require('../services/encryptionService');

const usersFile = path.join(\_\_dirname, '../models/users.json');

const logFile = path.join(\_\_dirname, '../logs/access.log');

if (!fs.existsSync(usersFile)) fs.writeFileSync(usersFile, '[]');

if (!fs.existsSync(logFile)) fs.writeFileSync(logFile, '');

* Import các thư viện để thao tác file, đường dẫn và mã hóa.
* Xác định đường dẫn file lưu thông tin người dùng và file log.
* Nếu file chưa tồn tại thì tạo file mới.

function logAccess(user, action) {

  const time = new Date().toISOString();

  fs.appendFileSync(logFile, `[${time}] ${user}: ${action}\n`);

}

* Ghi lại mọi thao tác của người dùng hoặc admin vào file log, gồm thời gian, tên người thao tác và hành động.

exports.register = (req, res) => {

  const { name, cccd, address, bank, password } = req.body;

  const user = {

    name,

    cccd: encryption.encryptTripleDES(cccd),

    address: encryption.encryptAES(address),

    bank: encryption.encryptAES(bank),

    password

  };

  const users = JSON.parse(fs.readFileSync(usersFile));

  users.push(user);

  fs.writeFileSync(usersFile, JSON.stringify(users, null, 2));

  logAccess(name, 'Đăng ký');

  res.json({ message: 'Đăng ký thành công!' });

};

* Nhận thông tin đăng ký từ client.
* Mã hóa các trường nhạy cảm trước khi lưu.
* Lưu thông tin vào file và ghi log thao tác đăng ký.

exports.login = (req, res) => {

  const { name, password } = req.body;

  const users = JSON.parse(fs.readFileSync(usersFile));

  const user = users.find(u => u.name === name && u.password === password);

  if (!user) return res.status(401).json({ error: 'Sai thông tin đăng nhập' });

  // Giải mã

  const result = {

    name: user.name,

    cccd: encryption.decryptTripleDES(user.cccd),

    address: encryption.decryptAES(user.address),

    bank: encryption.decryptAES(user.bank)

  };

  logAccess(name, 'Đăng nhập xem thông tin cá nhân');

  res.json(result);

};

* Kiểm tra thông tin đăng nhập.
* Nếu đúng, giải mã các trường nhạy cảm và trả về cho client.
* Ghi log thao tác đăng nhập.

exports.listUsers = (req, res) => {

  const users = JSON.parse(fs.readFileSync(usersFile));

  // Ẩn thông tin nhạy cảm

  const safeUsers = users.map(u => ({

    name: u.name,

    cccd: '\*\*\*',

    address: '\*\*\*',

    bank: '\*\*\*'

  }));

  res.json(safeUsers);

};

* Trả về danh sách user nhưng ẩn các trường nhạy cảm.

exports.adminDetail = (req, res) => {

  const { name, adminPass } = req.body;

  if (adminPass !== 'admin123') return res.status(403).json({ error: 'Sai mật khẩu admin' });

  const users = JSON.parse(fs.readFileSync(usersFile));

  const user = users.find(u => u.name === name);

  if (!user) return res.status(404).json({ error: 'Không tìm thấy user' });

  const result = {

    name: user.name,

    cccd: encryption.decryptTripleDES(user.cccd),

    address: encryption.decryptAES(user.address),

    bank: encryption.decryptAES(user.bank)

  };

  logAccess('ADMIN', `Xem chi tiết user ${name}`);

  res.json(result);

};

* Chỉ admin nhập đúng mật khẩu mới xem được thông tin chi tiết (đã giải mã) của user.
* Ghi log thao tác của admin.

exports.getLogs = (req, res) => {

  const logs = fs.readFileSync(logFile, 'utf8');

  res.send(logs);

};

* Trả về hết log truy cập cho admin kiểm tra lịch sử thao tác.

## **4.4. Ghi log truy cập**

*File: access.log*

Ghi lại các thao tác đăng ký, đăng nhập, xem chi tiết, ai thao tác, thời gian.

[2025-06-29T09:23:13.928Z] Phạm Ngọc Sơn: Đăng ký

[2025-06-29T09:23:29.238Z] Phạm Ngọc Sơn: Đăng nhập xem thông tin cá nhân

[2025-06-29T09:23:35.566Z] Phạm Ngọc Sơn: Đăng nhập xem thông tin cá nhân

[2025-06-29T09:24:32.144Z] ADMIN: Xem chi tiết user Phạm Ngọc Sơn

[2025-06-29T09:26:20.348Z] Phạm Ngọc Sơn: Đăng nhập xem thông tin cá nhân

[2025-06-29T09:37:00.413Z] Phạm Ngọc Sơn: Đăng nhập xem thông tin cá nhân

[2025-06-29T09:37:38.189Z] Hoàng Văn An: Đăng ký

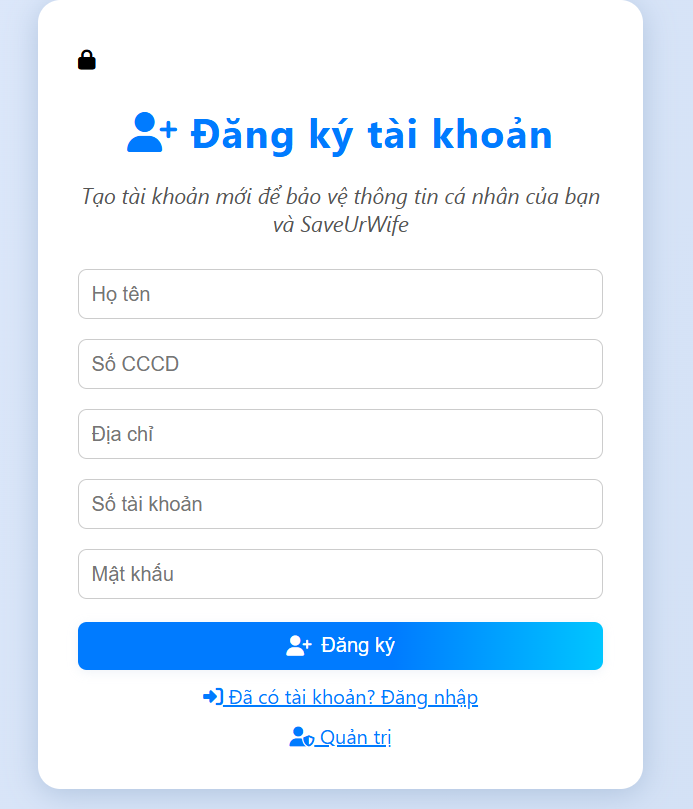
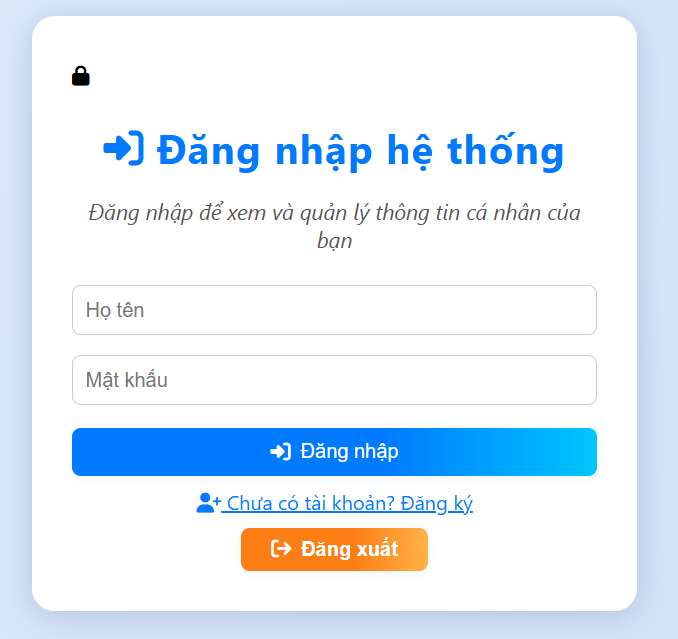
[2025-06-29T09:42:44.958Z] Bùi Thiên Phú: Đăng ký

## **4.5. Giao diện người dùng và quản trị**

*index.html*: Giao diện đăng ký, đăng nhập, xem thông tin cá nhân.

*admin.html*: Giao diện quản trị, xem danh sách người dùng, xem chi tiết, log truy cập.

*main.js*: Xử lý sự kiện giao diện, gọi API, hiển thị kết quả.

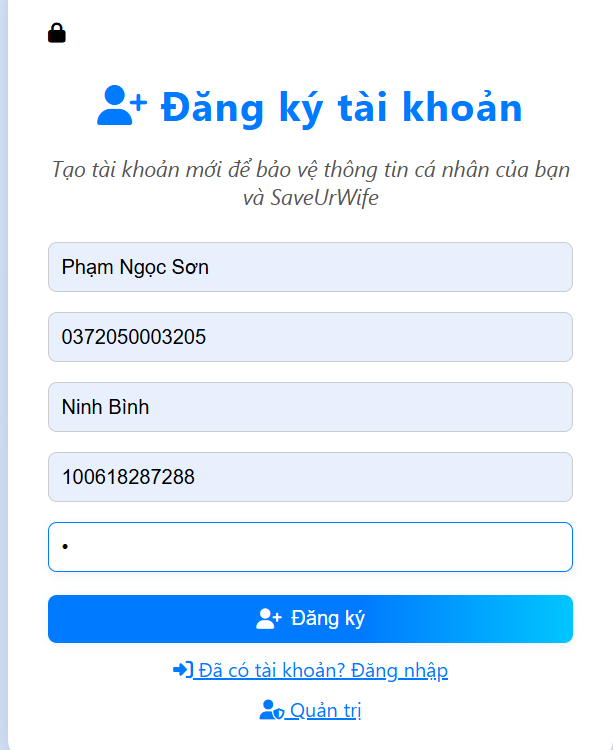
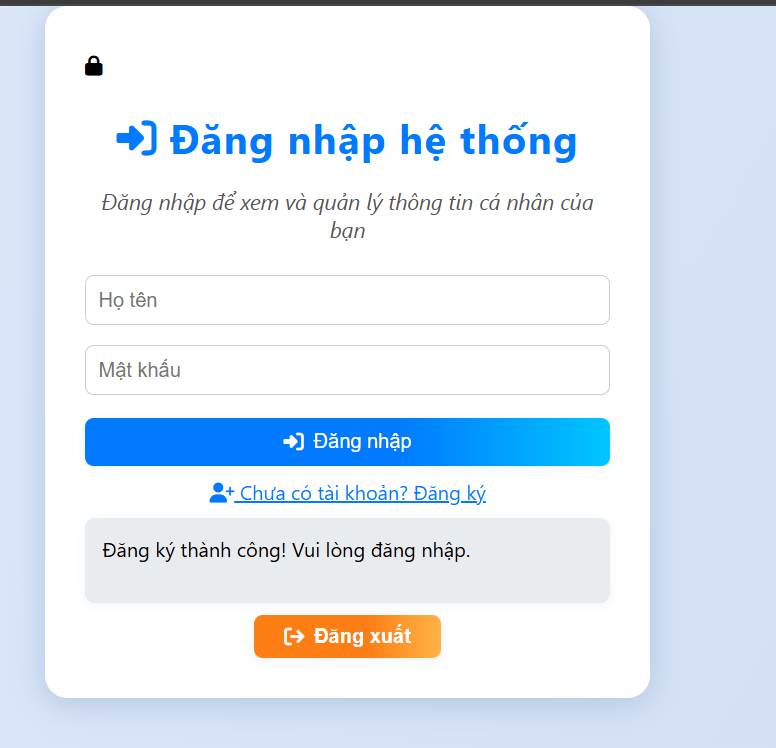
 



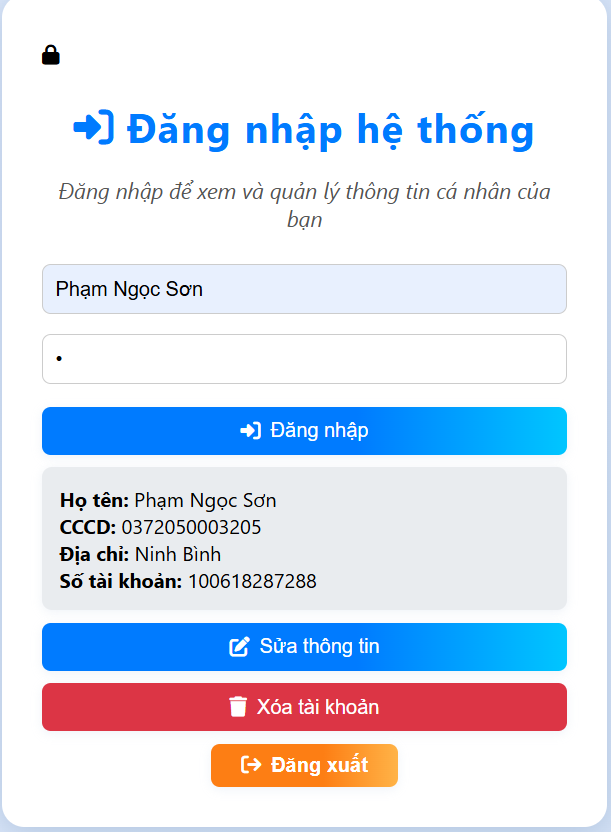
# **CHƯƠNG 5. THỬ NGHIỆM VÀ ĐÁNH GIÁ**

## **5.1. Kịch bản kiểm thử chức năng**

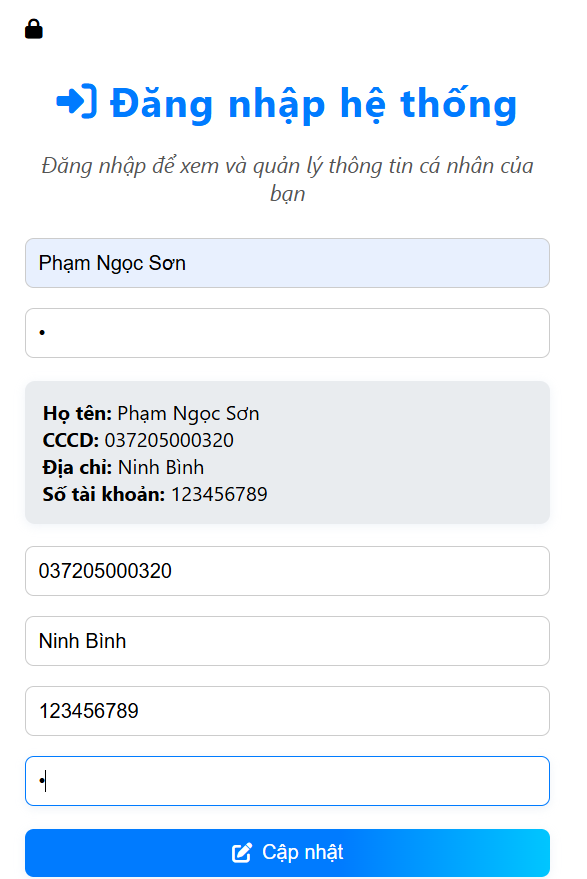
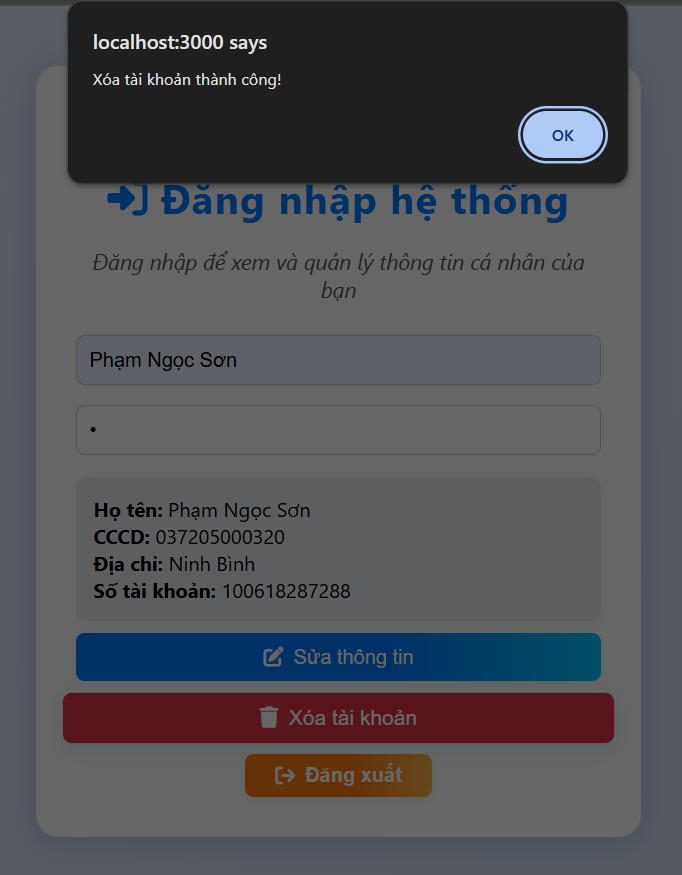
Đăng ký tài khoản mới với dữ liệu hợp lệ.

Đăng nhập với tài khoản đã đăng ký và xem thông tin cá nhân sau khi đăng nhập.

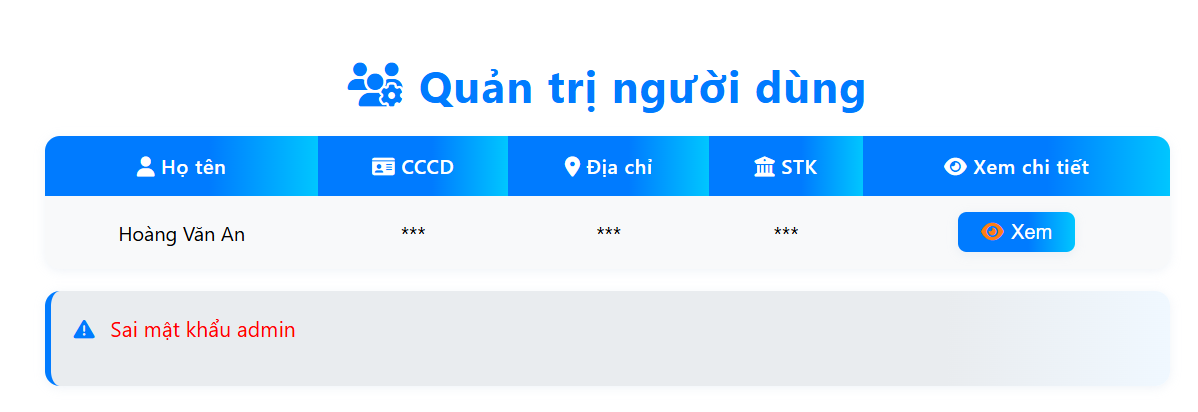


Sửa, xóa thông tin:

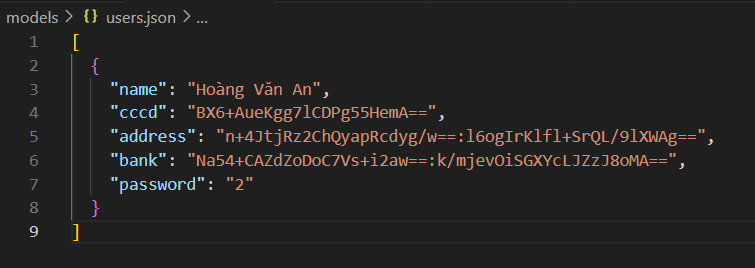
Quản trị viên xem chi tiết thông tin người dùng (nhập đúng/sai mật khẩu admin).





## 5.2. Kịch bản kiểm thử bảo mật

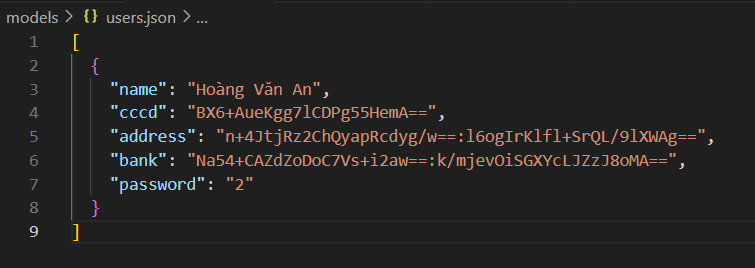
Truy cập file users.json trực tiếp: dữ liệu đã được mã hóa, không đọc được thông tin gốc.



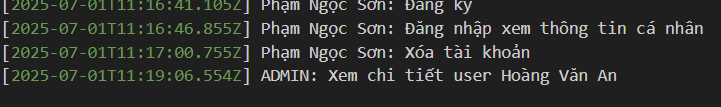
* cccd: Triple DES
* address, bank: AES
* name, password: Không mã hóa

## 5.3. Kết quả thử nghiệm

Dữ liệu trong file users.json đã được mã hóa, không đọc được thông tin gốc.



Log truy cập ghi nhận đầy đủ các thao tác.



## 5.4. Đánh giá hiệu quả bảo mật và hiệu suất

- Dữ liệu nhạy cảm được bảo vệ tốt, không thể đọc trực tiếp từ file.

- Hệ thống xử lý nhanh với dữ liệu nhỏ và vừa.

- Triple DES chậm hơn AES khi xử lý dữ liệu lớn.

- Hệ thống phù hợp cho bài tập lớn, demo, chưa phù hợp cho hệ thống lớn thực tế.

# **CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## **6.1. Kết luận**

Hệ thống đã hoàn thành các chức năng cơ bản về bảo mật thông tin cá nhân, đáp ứng yêu cầu mã hóa dữ liệu nhạy cảm trước khi lưu trữ. Việc sử dụng các thuật toán mã hóa đối xứng như Triple DES và AES giúp đảm bảo rằng chỉ những người có quyền truy cập mới có thể xem được dữ liệu gốc, từ đó nâng cao mức độ an toàn cho hệ thống.

Quá trình xây dựng hệ thống đã giúp sinh viên hiểu rõ hơn về cách áp dụng các thuật toán mã hóa vào thực tế, nắm vững quy trình bảo vệ dữ liệu cá nhân cũng như rèn luyện kỹ năng lập trình an toàn. Ngoài ra, việc triển khai các chức năng như đăng ký, đăng nhập, sửa, xóa thông tin cá nhân và ghi log truy cập đã góp phần hoàn thiện hệ thống, giúp người dùng chủ động quản lý và bảo vệ dữ liệu của mình.

## **6.2. Hạn chế**

Mặc dù hệ thống đã đáp ứng được các yêu cầu cơ bản về bảo mật, nhưng vẫn còn tồn tại một số hạn chế nhất định:

* Hệ thống hiện tại sử dụng file JSON để lưu trữ dữ liệu, điều này không phù hợp khi mở rộng quy mô hoặc triển khai trên môi trường thực tế với nhiều người dùng.
* Mật khẩu người dùng được lưu dưới dạng plain text, chưa được mã hóa hay hash, tiềm ẩn nguy cơ lộ thông tin nếu file bị truy cập trái phép.
* Hệ thống chưa tích hợp xác thực hai lớp (Two-Factor Authentication), do đó mức độ bảo vệ tài khoản người dùng chưa cao.
* Chưa có cơ chế kiểm tra tính toàn vẹn dữ liệu, nên dữ liệu có thể bị thay đổi mà không phát hiện được.
* Giao diện còn đơn giản, chưa tối ưu cho trải nghiệm người dùng trên nhiều thiết bị.

## **6.3. Đề xuất cải tiến**

Để nâng cao tính bảo mật và khả năng mở rộng của hệ thống trong tương lai, có thể thực hiện một số cải tiến sau:

* **Sử dụng cơ sở dữ liệu chuyên nghiệp** như MySQL hoặc MongoDB thay cho file JSON, giúp quản lý dữ liệu hiệu quả hơn, hỗ trợ truy vấn nhanh và an toàn khi hệ thống có nhiều người dùng.
* **Hash mật khẩu người dùng** trước khi lưu trữ bằng các thuật toán như bcrypt hoặc Argon2, đảm bảo ngay cả khi dữ liệu bị lộ thì mật khẩu vẫn được bảo vệ.
* **Bổ sung xác thực đa yếu tố (Multi-Factor Authentication)** để tăng cường bảo vệ tài khoản, giảm nguy cơ bị tấn công bởi các đối tượng xấu.
* **Kiểm tra tính toàn vẹn dữ liệu** bằng cách sử dụng các thuật toán hash hoặc chữ ký số, giúp phát hiện kịp thời nếu dữ liệu bị thay đổi hoặc giả mạo.
* **Tối ưu giao diện và trải nghiệm người dùng**, thiết kế giao diện thân thiện, dễ sử dụng và tương thích với nhiều thiết bị (máy tính, điện thoại, máy tính bảng).
* **Bổ sung phân quyền người dùng** (user, admin) để kiểm soát tốt hơn các thao tác trên hệ thống.
* **Tích hợp các biện pháp bảo vệ chống tấn công phổ biến** như CSRF, XSS, SQL Injection khi chuyển sang dùng cơ sở dữ liệu thực thụ.

Những cải tiến này sẽ giúp hệ thống trở nên an toàn, chuyên nghiệp và phù hợp hơn với các yêu cầu thực tế trong lĩnh vực bảo mật thông tin cá nhân.

# **TÀI LIỆU THAM KHẢO**

1. Giáo trình *An toàn và Bảo mật thông tin*, Trường Đại học Bách Khoa Hà Nội.
2. Trang chủ Node.js: https://nodejs.org/
3. Tài liệu thư viện mã hóa Crypto của Node.js: https://www.npmjs.com/package/crypto
4. Bài viết về thuật toán AES trên Wikipedia: https://vi.wikipedia.org/wiki/Advanced\_Encryption\_Standard
5. Bài viết về thuật toán Triple DES trên Wikipedia: https://vi.wikipedia.org/wiki/Triple\_DES
6. Một số bài hướng dẫn thực hành mã hóa dữ liệu với Node.js trên các diễn đàn lập trình và Stack Overflow.
7. Tài liệu tham khảo về bảo mật ứng dụng web từ OWASP: https://owasp.org/