VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Programming Intergration Project (CO3103)

CC01 - Group 07

File Sharing Web Frontend

Advisor(s):	Lê Đình Thuận	
Student(s):	Lê Nguyễn Bảo Minh	ID: 2352742
	Cao Nguyễn Gia Khánh	ID: 2352514
	Mai Trung Kiên	ID: 2352641
	Thi Minh Thức	ID: 2313406

HO CHI MINH CITY, JANUARY 2026

MEMBER LIST

CLASS CC01 - GROUP 07

No.	**Fullname**	**Student ID**	**Work load**	**% done**
1	Lê Nguyễn Bảo Minh	2352742	Auth & TOTP & User Dashboard	100%
2	Cao Nguyễn Gia Khánh	2352514	Upload File	100%
3	Mai Trung Kiên	2352641	Admin System	100%
4	Thi Minh Thức	2313406	Access & Download	100%

Table 0.1: Team members' contributions

Contents

List of Tables

1 Tổng quan dự án

1.1 Vai trò

Lập trình viên Frontend

1.2 Mục tiêu

Mục tiêu chính của dự án là thiết kế và xây dựng một giao diện web phản hồi nhanh (responsive), bảo mật và lấy người dùng làm trung tâm cho hệ thống chia sẻ tệp tin tạm thời. Hệ thống được xây dựng dựa trên repository `file-sharing-fe-web`, đóng vai trò là cầu nối mượt mà giữa người dùng cuối và các logic bảo mật phức tạp được định nghĩa trong *File Sharing System API* ở backend.

Các trách nhiệm chính và mục tiêu triển khai bao gồm:

- **Quy trình Tải lên Bảo mật:** Phát triển giao diện upload động, thực thi các quy tắc validation ngay tại phía client (ví dụ: đảm bảo ngày *availableFrom* luôn trước ngày *availableTo*) và cấu hình quyền riêng tư có điều kiện (ẩn/hiện trường mật khẩu và danh sách email dựa trên trạng thái public/private).
- **Xác thực Nâng cao:** Triển khai bộ tính năng xác thực toàn diện, vượt ra ngoài việc đăng nhập thông thường để hỗ trợ quy trình Mật khẩu Một lần theo Thời gian (TOTP), bao gồm hiển thị mã QR để thiết lập và xác minh hai bước (2FA) khi đăng nhập.
- **Dashboard & Quản lý:** Xây dựng bảng điều khiển trung tâm cho phép người dùng đã đăng nhập theo dõi vòng đời các tệp tin của họ, xem trạng thái thời gian thực (Đang hoạt động, Chờ xử lý, Đã hết hạn) và thống kê chi tiết lượt tải xuống.
- **Xử lý Truy cập Công khai:** Xây dựng trang đích (landing page) tải xuống an toàn, có khả năng diễn giải các mã trạng thái HTTP để phản hồi phù hợp với người dùng, chẳng hạn như xử lý lỗi "Gone" (410) cho file hết hạn, "Locked" (423) cho file chưa đến giờ mở, và các yêu cầu nhập mật khẩu bảo vệ.

2 Technology Stack

Dự án sử dụng các công nghệ sau:

- **Framework cốt lõi:** `Next.js 16` (React 19) - Tận dụng khả năng Server-Side Rendering (SSR) và tối ưu hóa hiệu năng mới nhất.
- **Ngôn ngữ lập trình:** `TypeScript` (chiếm 77.9% source code) - Đảm bảo tính chặt chẽ về kiểu dữ liệu và giảm thiểu lỗi runtime.
- **Styling (Giao diện):** `Tailwind CSS v4` - Sử dụng kết hợp với `clsx` và `tailwind-merge` để xử lý các class động một cách linh hoạt.
- **HTTP Client:** `Axios` - Quản lý các request API, xử lý Interceptors cho việc đính kèm Token và xử lý lỗi toàn cục.
- **Quản lý trạng thái (State Management):** Sử dụng `React Hooks` và `Context API` (tích hợp sẵn trong React/Next.js) để quản lý trạng thái phiên làm việc và dữ liệu người dùng.
- **Tiện ích khác:**
  - `js-cookie`: Quản lý lưu trữ Token bảo mật trong Cookie.
  - `sonner`: Hiển thị thông báo (toast notifications) cho trải nghiệm người dùng mượt mà.
  - `lucide-react`: Bộ icon hiện đại, nhẹ nhàng cho giao diện.

3 Project Specification

3.1 Giới hạn dự án

Phần Frontend của hệ thống File Sharing Web được xây dựng nhằm cung cấp giao diện người dùng hiện đại, dễ sử dụng và bảo mật, cho phép người dùng tải lên, chia sẻ và quản lý các tệp tin tạm thời. Phạm vi của Frontend được xác định rõ ràng trong bối cảnh hệ thống bao gồm cả nhóm Backend (Go) và nhóm Frontend (Web).

3.2 Phạm vi triển khai của Frontend

Frontend chịu trách nhiệm cho toàn bộ các nhiệm vụ liên quan đến giao diện và tương tác người dùng, bao gồm:

- Thiết kế và hiện thực UI/UX cho toàn bộ luồng hoạt động của người dùng: đăng nhập, thiết lập TOTP, tải lên file, truy cập file chia sẻ, quản lý file trong dashboard.
- Thực hiện toàn bộ xử lý logic ở phía client: validation dữ liệu, quản lý trạng thái, thông báo lỗi/thành công, điều hướng người dùng, hiển thị thời gian hiệu lực file.
- Tương tác trực tiếp với Backend bằng REST API để gọi authentication, upload file, truy xuất metadata, tải file xuống, cập nhật hoặc xóa file.
- Xây dựng cấu trúc hệ thống Frontend (routing, service layer, component structure) theo Next.js App Router.
- Đảm bảo hệ thống chạy được trong môi trường production bằng Docker Compose và tuân thủ yêu cầu CI/CD của dự án.

3.3 Hợp tác với Backend

- Frontend sử dụng API mà nhóm Backend cung cấp, không tự định nghĩa logic riêng.
- Giao diện phải tương thích đa nền tảng và hỗ trợ SSR theo kiến trúc Next.js.
- Hệ thống phải được containerized hoàn toàn và deploy bằng GitHub Actions sử dụng self-hosted runners.

4 Functional requirement

4.1 Xác thực & Phân quyền

- **Đăng ký**: Người dùng có thể đăng ký tài khoản mới bằng email, tên đăng nhập và mật khẩu.
- **Đăng nhập**: Người dùng có thể xác thực bằng email và mật khẩu.
- **Xác thực Hai yếu tố (TOTP)**:
  - Người dùng có thể kích hoạt TOTP cho tài khoản của họ (yêu cầu quét mã QR và xác minh).
  - Nếu được kích hoạt, việc đăng nhập yêu cầu mã TOTP hợp lệ.
  - Người dùng có thể vô hiệu hóa TOTP (yêu cầu xác minh mã TOTP hiện tại).
- **Quản lý Mật khẩu**: Người dùng có thể thay đổi mật khẩu. Việc này yêu cầu mật khẩu cũ hoặc mã TOTP (nếu đã kích hoạt).
- **Đăng xuất**: Người dùng có thể đăng xuất khỏi hệ thống một cách an toàn.

4.2 Quản lý Tệp tin (Tải lên)

- **Tải lên Tệp tin**: Người dùng có thể tải lên tệp tin với các ràng buộc sau:
  - **Định dạng hỗ trợ**: pdf, doc, docx, xls, xlsx, ppt, pptx, txt, csv, jpg, jpeg, png, gif, zip, rar, mp4, mp3.
  - **Giới hạn kích thước tệp**: Được thực thi bởi chính sách hệ thống (mặc định 50MB).
- **Cấu hình Bảo mật**:
  - **Bảo vệ bằng Mật khẩu**: Người dùng có thể đặt mật khẩu để truy cập tệp tin (độ dài tối thiểu được thực thi bởi chính sách).
  - **Thời gian Hiệu lực**: Người dùng có thể đặt thời gian "Có hiệu lực từ" (Available From) và "Có hiệu lực đến" (Available To). Hệ thống kiểm tra thời gian bắt đầu phải trước thời gian kết thúc.

  - **Chia sẻ Riêng tư**: Người dùng có thể giới hạn quyền truy cập cho một danh sách các địa chỉ email cụ thể.

- **Logic Hiển thị**: Tệp tin sẽ tự động được đánh dấu là "Riêng tư" nếu bật bảo vệ bằng mật khẩu hoặc chia sẻ với các email cụ thể. Ngược lại, chúng mặc định là "Công khai" (bất kỳ ai có liên kết đều có thể truy cập).

4.3 Bảng điều khiển Người dùng

- **Xem Hồ sơ**: Người dùng có thể xem thông tin hồ sơ của họ.
- **Thống kê Tệp tin**: Người dùng có thể xem tóm tắt về các tệp tin của họ, bao gồm số lượng tệp Đang hoạt động (Active), Đang chờ (Pending), Đã hết hạn (Expired) và Đã xóa (Deleted).
- **Danh sách Tệp tin**: Người dùng có thể xem danh sách phân trang các tệp tin đã tải lên.
  - **Lọc**: Lọc tệp tin theo trạng thái (ví dụ: Đang hoạt động, Đã hết hạn).
  - **Sắp xếp**: Sắp xếp tệp tin theo ngày tạo hoặc các tiêu chí khác.
- **Thao tác với Tệp tin**: Người dùng có thể xóa các tệp tin đã tải lên của họ.

4.4 Quản trị

- **Quản lý Chính sách Hệ thống**: Quản trị viên có thể xem và cập nhật các chính sách hệ thống toàn cầu ảnh hưởng đến tất cả người dùng:
  - `maxFileSizeMB`: Kích thước tệp tối đa cho phép (MB).
  - `minValidityHours`: Thời gian hiệu lực tối thiểu (giờ).
  - `maxValidityDays`: Thời gian hiệu lực tối đa (ngày).
  - `defaultValidityDays`: Thời gian hiệu lực mặc định (ngày).
  - `requirePasswordMinLength`: Độ dài tối thiểu yêu cầu cho mật khẩu tệp tin.
- **Dọn dẹp Hệ thống**: Quản trị viên có thể kích hoạt quy trình dọn dẹp để xóa vĩnh viễn các tệp tin đã hết hạn hoặc đã xóa mềm khỏi hệ thống.

5 Non-functional Requirements

5.1 Hiệu năng

- **Thời gian tải trang**: Các trang chính (Login, Dashboard, Upload, Download) tải trong vòng ≤ 2 giây trong điều kiện mạng ổn định.
- **Phản hồi giao diện**: Các thao tác UI như mở modal, chuyển trang nội bộ, và hiển thị toast phải phản hồi **gần tức thời** (thường dưới 200–300 ms).

5.2 Tính dễ sử dụng

- **Rõ ràng**: Trạng thái file (Active, Pending, Expired) phải được phân biệt bằng màu sắc, icon nhất quán, trực quan.
- **Nhất quán**: Các toast thông báo (success / warning / error) phải sử dụng chung một style và thời gian hiển thị (mặc định 3-5 giây).

5.3 Khả năng bảo trì

- **Hệ thống mã nguồn rõ ràng**: Mã nguồn được chia theo module (auth, dashboard, upload, admin).
- **Tái sử dụng**: Các UI component phải được gom vào thư mục components và được tái sử dụng thay vì viết lại.
- **Dễ mở rộng**: Hệ thống phải được thiết kế để dễ dàng bổ sung thêm kiểu bảo mật, loại file hoặc chính sách mới.

6 UI Design & screen flow

6.1 Screen Flow

6.1.1 Public

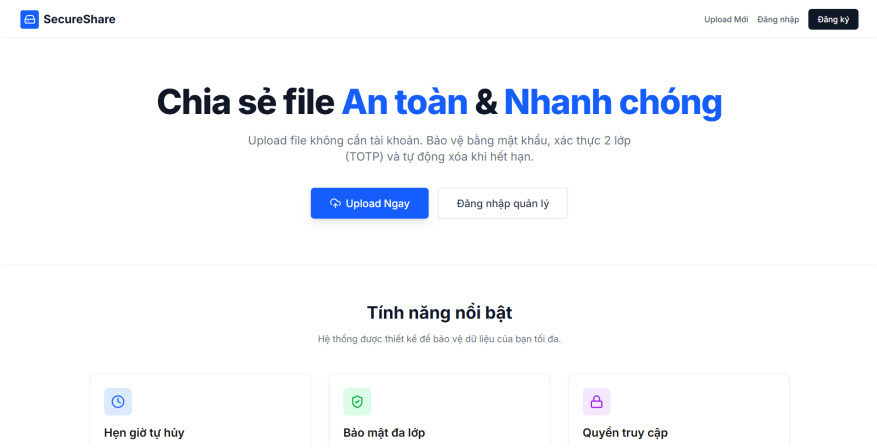

Figure 6.1: Home page

Home Page

- **Route**: `/`
- **File**: `app/page.tsx`

6.1.2 Authentication Screens

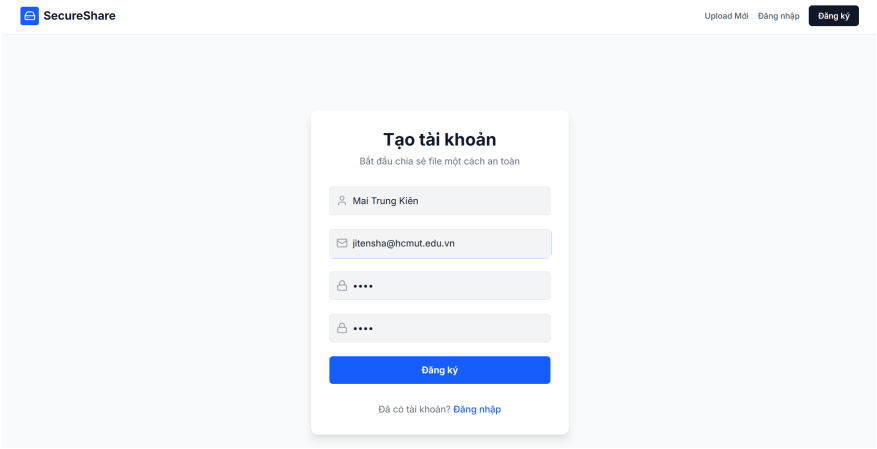

Figure 6.2: Đăng ký

Đăng ký

- **Route**: `/register`
- **File**: `app/(auth)/register/page.tsx`

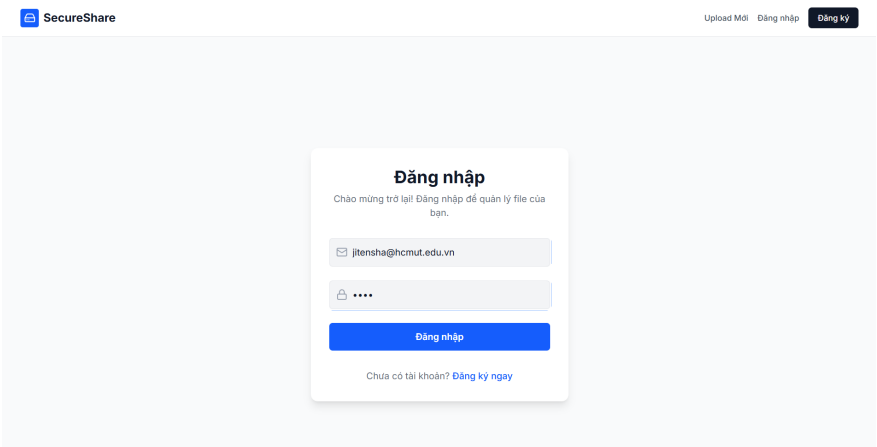

Figure 6.3: Đăng nhập

Đăng nhập

- **Route**: `/login`

- **File**: `app/(auth)/login/page.tsx`

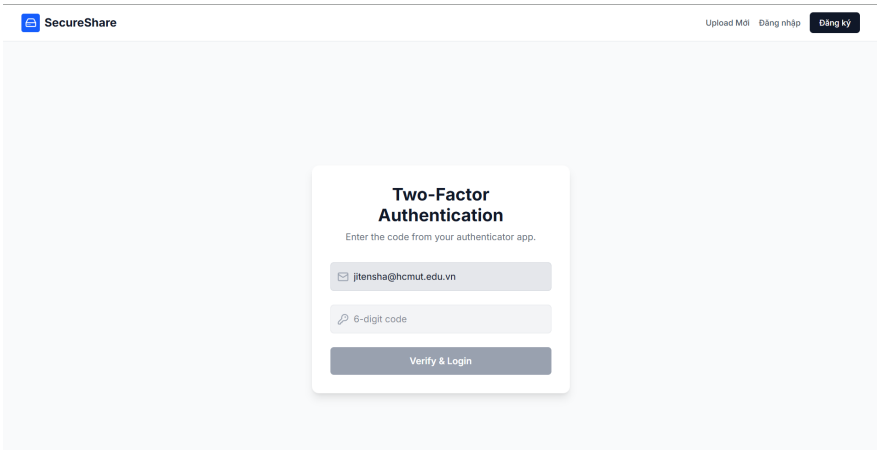

Figure 6.4: Đăng nhập với TOTP

Đăng nhập với TOTP

- **Route**: `/login/totp`
- **File**: `app/(auth)/login/totp/page.tsx`

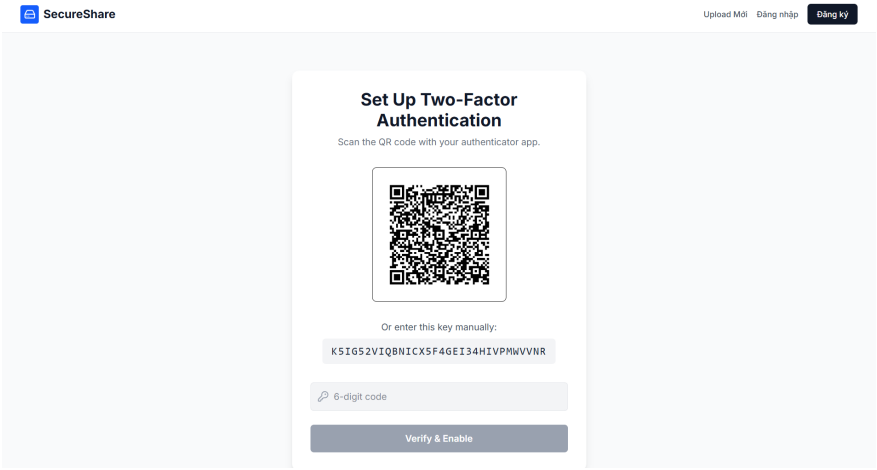

Figure 6.5: Thiết lập TOTP

Thiết lập TOTP

- **Route**: `/totp-setup`

- **File**: `app/(auth)/totp-setup/page.tsx`

6.1.3 User Screens

Bảng điều khiển

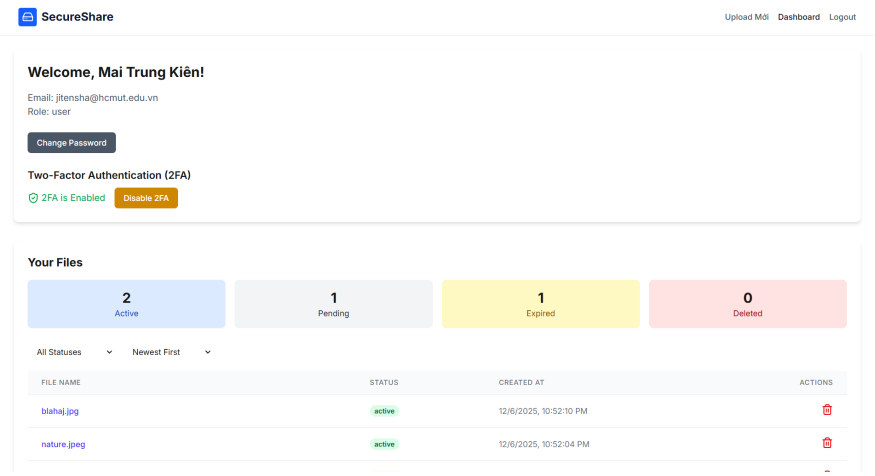

Figure 6.6: Bảng điều khiển

- **Route**: `/dashboard`
- **File**: `app/dashboard/page.tsx`

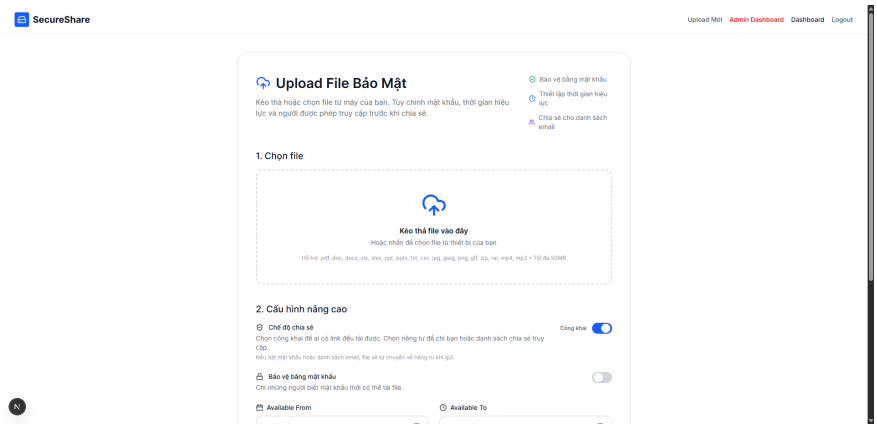

Figure 6.7: Upload

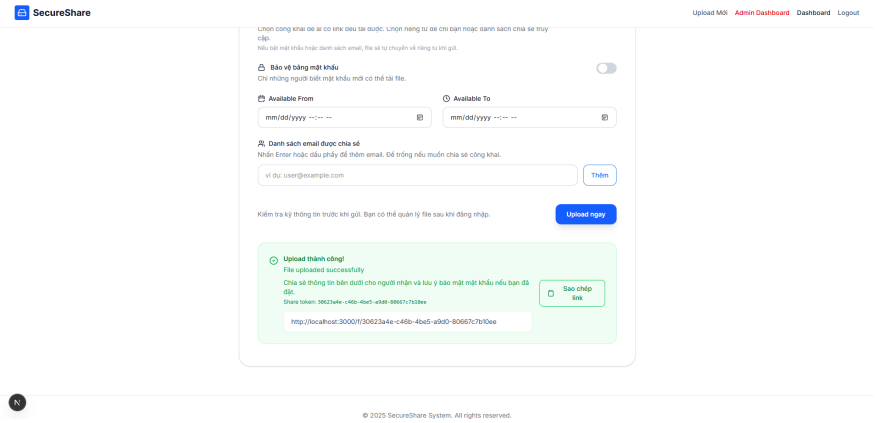

Figure 6.8: Upload thành công

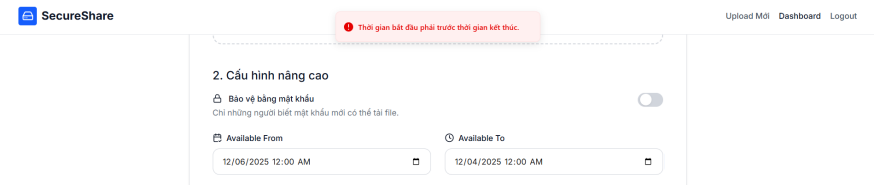

Figure 6.9: Upload thất bại

Upload

- **Route**: `/upload`
- **File**: `app/upload/page.tsx`

6.1.4 Public File Access

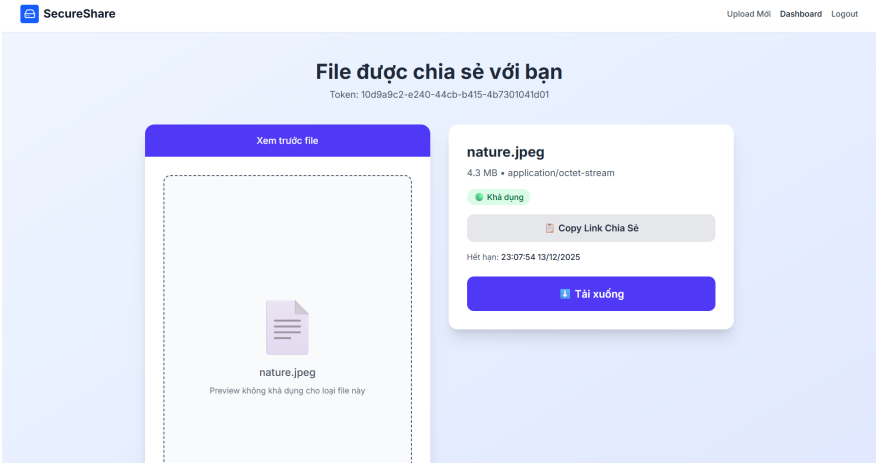

Figure 6.10: Trang truy cập công khai

Trang truy cập công khai

- **Route**: `/f/[token]`
- **File**: `app/(public)/f/[token]/page.tsx`

6.1.5 Admin Screens

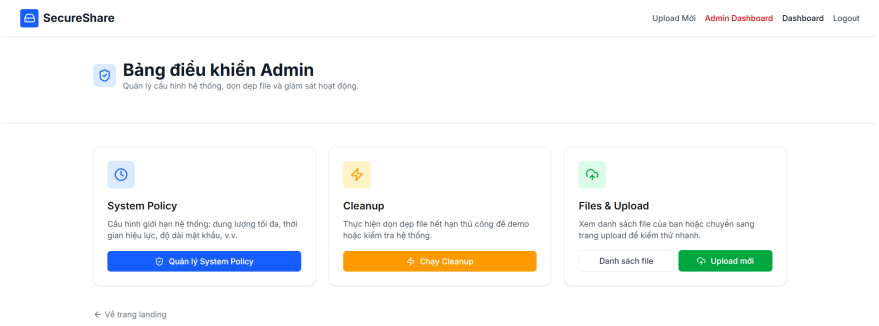

Figure 6.11: Trang chính Admin

Trang chính Admin

- **Route**: `/admin`
- **File**: `app/admin/page.tsx`

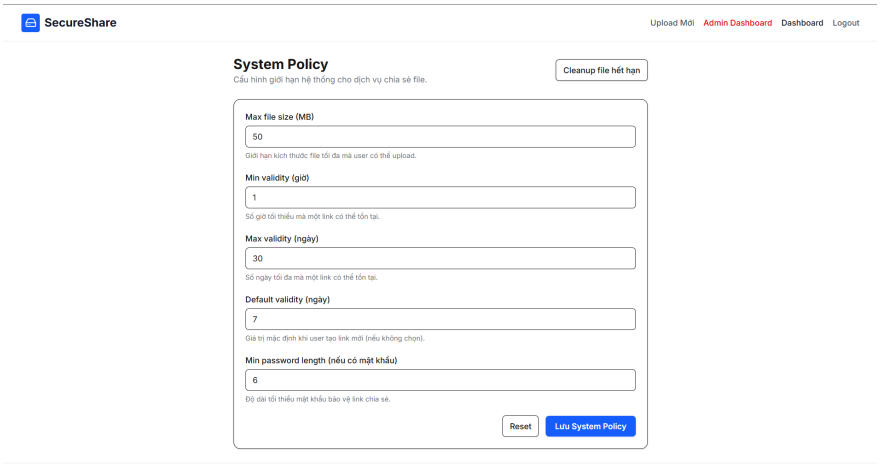

Figure 6.12: Admin Policy

Admin Policy

- **Route**: `/admin/policy`
- **File**: `app/admin/policy/page.tsx`

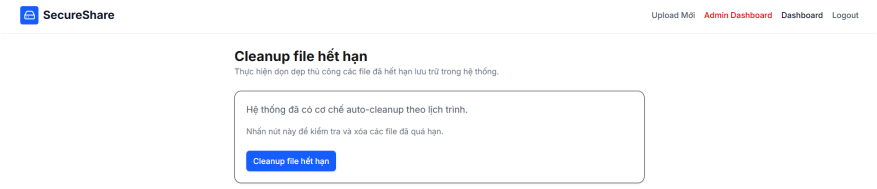

Figure 6.13: Admin Cleanup

Admin Cleanup

- **Route**: `/admin/cleanup`

- **File**: `app/admin/cleanup/page.tsx`

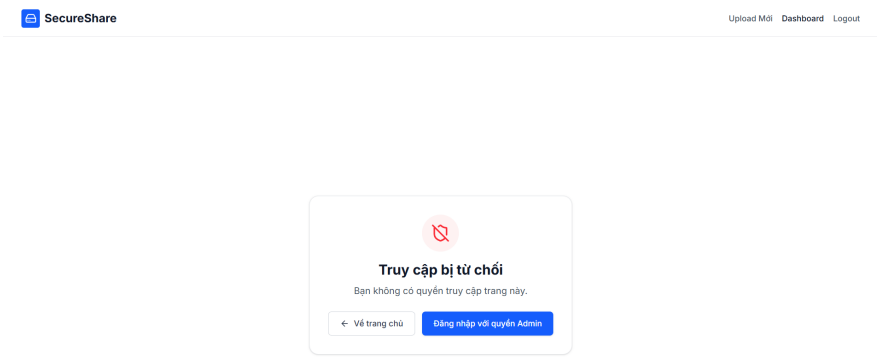

Figure 6.14: Not Admin

Not Admin

- **Route**: `/admin/templates/notadmin`
- **File**: `app/admin/templates/notadmin.tsx`

7 Testing

7.1 Component Testing

7.1.1 /admin/CleanupButton.test.tsx

File kiểm thử này sử dụng cơ chế mocking cho `adminApi` và thư viện thông báo `sonner` để kiểm tra quy trình dọn dẹp hệ thống thông qua nút bấm `CleanupButton`.

- **Xử lý dọn dẹp thành công** (`testCleanupSuccess`):
  Mô phỏng trường hợp API trả về kết quả thành công sau khi gọi.
  Setup: Giả lập `adminApi.cleanupExpiredFiles` trả về object chứa `message: 'Cleanup successful'`.
  Action: Click vào nút "Cleanup file hết hạn".
  Assertion:
  - Giao diện chuyển sang trạng thái đang tải: hiển thị văn bản "Dang cleanup...".
  - Hàm API `cleanupExpiredFiles` được thực thi.
  - Thông báo thành công `toast.success` được gọi với nội dung chứa "Cleanup successful".
  - Nút bấm quay trở lại trạng thái văn bản ban đầu ("Cleanup file hết hạn").
- **Xử lý dọn dẹp thất bại** (`testCleanupFailure`):
  Mô phỏng trường hợp API gặp lỗi (ví dụ: lỗi mạng hoặc lỗi server).
  Setup: Giả lập `adminApi.cleanupExpiredFiles` bị từ chối (`rejected`) với lỗi.
  Action: Click vào nút "Cleanup file hết hạn".
  Assertion: Hàm `toast.error` được gọi để thông báo lỗi cho người quản trị.

7.1.2 /admin/AdminShell.test.tsx

File kiểm thử này xác minh khả năng hiển thị của component layout `AdminShell`, đảm bảo tiêu đề, nội dung con, và các slot tùy chọn (mô tả, hành động) được render chính xác thông qua các hàm helper.

- **Render Cơ bản** (`testAdminShellRendering`): Kiểm tra các thành phần cốt lõi của layout.
  Input: Prop `title="Test Title"` và children là một thẻ `div` chứa text "Test Child".
  Assertion: Tiêu đề "Test Title" và nội dung con "Test Child" phải hiện diện trong DOM.

- **Render Mô tả** (`testAdminShellDescription`): Kiểm tra khả năng hiển thị văn bản mô tả phụ đề.
  Input: Prop `description="Test Description"`.
  Assertion: Văn bản "Test Description" phải xuất hiện trong giao diện.

- **Render Hành động** (`testAdminShellActions`): Kiểm tra khu vực chứa các nút tác vụ (action slot) thường nằm ở góc trên.
  Input: Prop `actions` được truyền vào một nút bấm `<button>`.
  Assertion: Phần tử nút bấm với tên "Action Button" phải được tìm thấy trong tài liệu thông qua `getByRole`.

7.1.3 `/admin/NumberField.test.tsx`

File kiểm thử này xác minh hoạt động của component nhập liệu số chuyên dụng `NumberField`, bao gồm việc hiển thị nhãn, mô tả, giá trị và xử lý các sự kiện thay đổi dữ liệu đặc thù.

- **Render Cơ bản** (`testNumberFieldRendering`): Kiểm tra cấu trúc DOM cơ bản.
  Input: Prop `label="Test Label"`.
  Assertion:
  - Nhãn "Test Label" hiển thị.
  - Phần tử nhập liệu số (role `spinbutton`) hiện diện trong DOM.

- **Render Mô tả** (`testNumberFieldDescription`): Kiểm tra hiển thị văn bản hướng dẫn phụ.
  Input: Prop `description="Test Description"`.
  Assertion: Văn bản mô tả phải xuất hiện trong giao diện.

- **Hiển thị Giá trị** (`testNumberFieldValueDisplay`): Kiểm tra binding dữ liệu hai chiều.

Input: Prop `value={10}`.
Assertion: Ô input phải hiển thị giá trị số 10.

- **Xử lý Thay đổi (`testNumberFieldChange`)**: Kiểm tra logic chuyển đổi sự kiện input thành dữ liệu số.
  Action: Nhập chuỗi '20' vào ô input.
  Assertion: Hàm mock `onChange` được gọi với tham số là số nguyên `20` (không phải chuỗi).

- **Xử lý Input Rỗng (`testNumberFieldEmptyChange`)**: Kiểm tra trường hợp người dùng xóa nội dung trong ô input.
  Action: Xóa giá trị trong ô input (gửi chuỗi rỗng).
  Assertion: Hàm mock `onChange` được gọi với tham số `undefined` (xử lý trạng thái null/empty).

- **Thuộc tính Min (`testNumberFieldMinAttribute`)**: Kiểm tra việc truyền tải các thuộc tính HTML chuẩn.
  Input: Prop `min={5}`.
  Assertion: Phần tử input trong DOM phải có thuộc tính `min="5"`.

7.1.4 `/ui/Alert.test.tsx`

File kiểm thử này xác minh khả năng hiển thị thông báo và việc áp dụng các lớp giao diện (styles) tương ứng với loại thông báo (success/error) của component `Alert`.

- **Render Message (`testAlertRendersMessage`)**:
  Kiểm tra nội dung văn bản được truyền vào component.
  Input: Prop `type="info"` và `message="This is an info"`.
  Assertion: Văn bản "This is an info" phải xuất hiện trong tài liệu.

- **Alert Style Success (`testAlertSuccessStyles`)**:
  Kiểm tra giao diện cho trạng thái thành công.
  Input: Prop `type="success"`.
  Assertion:
  - Container chứa thông báo phải có lớp CSS `bg-green-100` (nền xanh nhạt).

  - Biểu tượng "check mark" phải hiện diện trong giao diện.

- **Alert Style Error** (`testAlertErrorStyles`):
  Kiểm tra giao diện cho trạng thái báo lỗi.
  Input: Prop `type="error"`.
  Assertion:

  - Container chứa thông báo phải có lớp CSS `bg-red-100` (nền đỏ nhạt).
  - Biểu tượng "cross mark" phải hiện diện trong giao diện.

7.1.5 `/ui/Button.test.tsx`

File kiểm thử này đảm bảo component `Button` hiển thị đúng nội dung, áp dụng đúng các lớp CSS theo biến thể (variant), và xử lý trạng thái tải (loading) cũng như vô hiệu hóa (disabled) chính xác.

- **Render Children** (`testButtonRendersChildren`):
  Kiểm tra khả năng hiển thị nội dung con được truyền vào.
  Input: Text con "Click Me".
  Assertion: Văn bản "Click Me" phải xuất hiện trong tài liệu.

- **Variant Classes** (`testButtonVariantClasses`):
  Kiểm tra việc áp dụng lớp CSS cho biến thể giao diện.
  Input: Prop `variant="danger"`.
  Assertion: Nút bấm phải chứa lớp CSS cụ thể `bg-red-600`.

- **Trạng thái Loading** (`testButtonLoadingState`):
  Kiểm tra giao diện và hành vi khi nút đang trong trạng thái xử lý.
  Input: Prop `loading={true}`.
  Assertion:

  - Văn bản hiển thị thay đổi thành "Dang xử lý..." (thay vì nội dung gốc).
  - Nút bấm phải ở trạng thái bị vô hiệu hóa (`disabled`).

- **Sự kiện Click** (`testButtonOnClick`):
  Kiểm tra phản hồi khi người dùng nhấp chuột vào nút ở trạng thái hoạt động.

Action: Click vào nút.
Assertion: Hàm mock `onClick` được gọi chính xác 1 lần.

- **Chặn Click khi Disabled** (`testButtonDisabledOnClick`):
  Kiểm tra logic ngăn chặn tương tác khi nút bị vô hiệu hóa.
  Input: Prop `disabled={true}`.
  Action: Click vào nút.
  Assertion: Hàm mock `onClick` **không** được gọi.

7.1.6 `/ui/Input.test.tsx`

File này kiểm thử các biến thể hiển thị (UI variants) và khả năng tương tác của component nhập liệu cơ bản `Input`.

- **Render Label** (`testInputRendersWithLabel`):
  Kiểm tra khả năng hiển thị nhãn đi kèm.
  Input: Prop `label="Username"`.
  Assertion: Văn bản "Username" phải hiện diện trong DOM để hướng dẫn người dùng.

- **Render Icon** (`testInputRendersWithIcon`):
  Kiểm tra khả năng render slot biểu tượng (icon) bên trong ô input.
  Input: Prop `icon` chứa một thẻ span có `data-testid="icon"`.
  Assertion: Phần tử có test ID `"icon"` phải được tìm thấy trong tài liệu.

- **Hiển thị lỗi** (`testInputDisplaysErrorMessage`):
  Kiểm tra phản hồi giao diện khi dữ liệu không hợp lệ.
  Input: Prop `error="Invalid input"`.
  Assertion:
  - Thông báo lỗi "Invalid input" phải xuất hiện.
  - Ô input phải được áp dụng lớp CSS `border-red-500` (viền đỏ) để cảnh báo thị giác.

- **Tương tác người dùng** (`testInputHandlesChange`):
  Kiểm tra khả năng bắt sự kiện nhập liệu.
  Action: Giả lập người dùng nhập chuỗi `'test'` vào ô input.
  Assertion: Hàm mock `onChange` được truyền vào phải được kích hoạt (called).

7.1.7 /layout/Navbar.test.tsx

File kiểm thử này được cấu trúc lại thành các hàm helper riêng biệt, mỗi hàm chịu trách nhiệm kiểm tra một trạng thái cụ thể của thanh điều hướng thông qua việc giả lập các giá trị trả về của `_isLoggedIn` và `_isAdmin`.

- **Trạng thái Khách** (`testNavBarGuestLinks`):
  Giả lập trạng thái chưa đăng nhập (`_isLoggedIn = false`).
  Assertion:
  - Hiển thị các liên kết công khai: "SecureShare", "Upload Mới", "Đăng nhập", "Đăng ký".
  - Đảm bảo các liên kết bảo mật ("Dashboard", "Logout") không xuất hiện trong DOM.
- **Trạng thái Người dùng** (`testNavBarUserLinks`):
  Giả lập trạng thái đã đăng nhập (`_isLoggedIn = true`) nhưng không phải admin.
  Assertion:
  - Hiển thị liên kết "Dashboard" và "Logout".
  - Ẩn liên kết "Đăng nhập" và "Admin Dashboard".
- **Trạng thái Quản trị viên** (`testNavBarAdminLink`):
  Giả lập người dùng có quyền admin (`_isAdmin = true`).
  Assertion: Hiển thị liên kết "Admin Dashboard" và xác minh thuộc tính `href` trỏ chính xác về `'/admin'`.
- **Chức năng Đăng xuất** (`testLogoutFunctionality`):
  Kiểm tra quy trình xử lý sự kiện click vào nút "Logout".
  Assertion:
  - Hàm API `logout` được gọi.
  - Hệ thống điều hướng về trang đăng nhập thông qua hàm mock `navigateTo('/login')`.

7.1.8 /auth/RegisterForm.test.tsx

File kiểm thử này sử dụng mô hình các hàm helper riêng biệt để kiểm tra từng thành phần của form đăng ký. Test suite khởi tạo các mock function (`mockUpdateField`, `mockHandleSubmit`) và truyền vào component để xác minh hành vi.

- **Render UI** (`testRegisterPageRendering`):
  Kiểm tra component hiển thị đầy đủ 4 trường nhập liệu với placeholder chính xác: "Tên đăng nhập", "Email", "Mật khẩu", "Xác nhận mật khẩu" và sự hiện diện của nút "Đăng ký".
- **Xử lý Input từng trường** (`testRegisterPageUsernameInput`, `testRegisterPageEmailInput`, `testRegisterPagePasswordInput`, `testRegisterPageConfirmPasswordInput`):
  Hệ thống sử dụng các hàm test riêng biệt để đảm bảo binding dữ liệu chính xác cho từng ô input. Khi sự kiện `change` được kích hoạt, hàm `updateField` phải được gọi với key và value tương ứng.

Test function	Input value	Target field
`testRegisterPageUsernameInput`	`'newuser'`	`'username'`
`testRegisterPageEmailInput`	`'newuser@example.com'`	`'email'`
`testRegisterPagePasswordInput`	`'password123'`	`'password'`
`testRegisterPageConfirmPasswordInput`	`'password123'`	`'confirmPassword'`

Table 7.1: Mapping giữa test helper và field được cập nhật trong `RegisterForm`.

- **Xử lý Submit** (`testRegisterPageFormSubmission`):
  Mô phỏng hành động người dùng nhấn nút "Đăng ký" (submit form).
  Assertion: Hàm mock `handleSubmit` phải được gọi, xác nhận sự kiện đã được truyền đi để xử lý logic đăng ký.

7.1.9 /auth/LoginForm.test.tsx

Tương tự như quy trình đăng ký, file kiểm thử này khởi tạo `describe` suite và thiết lập các biến giả lập (`formData`, `updateField`, `handleSubmit`) để kiểm tra hành vi của component `LoginForm`.

- **Render UI** (`testLoginPageRendering`):
  Kiểm tra component `LoginForm` render chính xác với dữ liệu đầu vào.
  Assertion: Các trường input có placeholder "Email", "Mật khẩu" và nút "Đăng nhập" phải hiện diện trong DOM.

- **Xử lý Input** (`testLoginPageEmailInput`, `testLoginPagePasswordInput`):
  Kiểm tra sự tương tác giữa giao diện nhập liệu và hàm cập nhật trạng thái `updateField`.
  Action: Giả lập người dùng nhập liệu vào trường Email ('test@example.com') và Mật khẩu ('password123').
  Assertion: Hàm mock `updateField` phải được gọi chính xác với các cặp key-value tương ứng (`'email'`, `'test@example.com'`) và (`'password'`, `'password123'`).

- **Xử lý Submit** (`testLoginPageSubmit`):
  Kiểm tra quy trình gửi biểu mẫu đăng nhập.
  Action: Kích hoạt sự kiện submit trên form (thông qua nút Đăng nhập).
  Assertion: Hàm mock `handleSubmit` phải được gọi để chuyển tiếp sự kiện xử lý đăng nhập lên component cha.

7.1.10 `/auth/LoginTotpForm.test.tsx`

File kiểm thử này sử dụng các hàm helper để xác minh giao diện và logic tương tác của component nhập mã xác thực hai yếu tố `LoginTotpForm`.

- **Render Form** (`testLoginTotpRendering`): Kiểm tra các thành phần UI cơ bản được hiển thị chính xác dựa trên props đầu vào.
  Input: Mock props chứa email 'test@example.com'.
  Assertion:
  - Tiêu đề "Two-Factor Authentication" và dòng hướng dẫn "Enter the code from your authenticator app." phải xuất hiện.
  - Email 'test@example.com' được hiển thị trong ô input (dạng display value).
  - Trường nhập mã có placeholder "6-digit code" và nút bấm "Verify & Login" hiện diện trong DOM.

- **Xử lý Input** (`testLoginTotpCodeInput`): Kiểm tra binding dữ liệu khi người dùng nhập mã xác thực.

Action: Nhập chuỗi '123456' vào ô input.
Assertion: Hàm mock `updateField` được gọi với tham số `'code'` và giá trị `'123456'`.

- **Xử lý Submit** (`testLoginTotpSubmission`): Kiểm tra khả năng gửi biểu mẫu khi mã đã được nhập đầy đủ.
  Setup: Props đầu vào có `formData.code = '123456'`.
  Assertion:
  - Nút "Verify & Login" ở trạng thái hoạt động (không bị disabled).
  - Hàm mock `handleSubmit` được gọi khi kích hoạt sự kiện submit form.
- **Trạng thái Nút** (`testLoginTotpButtonState`): Kiểm tra logic vô hiệu hóa (disable) nút submit trong các trường hợp cụ thể.
  Assertion:
  - *Trường hợp thiếu mã*: Khi `code` rỗng, nút phải bị vô hiệu hóa.
  - *Trường hợp đang xử lý*: Khi prop `verifying = true` (dù đã có mã), nút vẫn phải bị vô hiệu hóa để ngăn gửi trùng lặp.

7.1.11 `/auth/TotpSetupForm.test.tsx`

File kiểm thử này sử dụng các hàm helper để bao quát mọi trạng thái giao diện của form thiết lập TOTP, từ lúc tải dữ liệu, hiển thị mã QR, đến xử lý lỗi và nhập liệu.

- **Trạng thái Loading** (`testTotpSetupLoading`): Kiểm tra giao diện khi đang khởi tạo dữ liệu bảo mật.
  Input: Prop `loading={true}`.
  Assertion: Văn bản "Loading Security Setup..." phải hiện diện trong DOM.
- **Trạng thái Lỗi** (`testTotpSetupError`): Kiểm tra phản hồi khi gặp sự cố (ví dụ: không lấy được mã QR).
  Input: Prop `error="Something went wrong"`.
  Assertion:
  - Hiển thị tiêu đề "Error" và nội dung lỗi "Something went wrong".
  - Hiển thị liên kết điều hướng "Go back to home".

- **Render Form** (`testTotpSetupRendering`): Kiểm tra hiển thị chuẩn khi dữ liệu sẵn sàng.
  Input: Props chứa QR code giả lập và secret key "TEST_SECRET".
  Assertion:
  - Tiêu đề "Set Up Two-Factor Authentication".
  - Hình ảnh QR với thuộc tính alt "TOTP QR Code".
  - Mã bí mật dạng văn bản "TEST_SECRET".
  - Ô nhập liệu với placeholder "6-digit code".
- **Xử lý Input** (`testTotpSetupCodeInput`): Kiểm tra binding dữ liệu cho mã xác thực.
  Action: Nhập chuỗi '123456'.
  Assertion: Hàm mock `updateField` được gọi với tham số `'totpCode'` và giá trị `'123456'`.
- **Xử lý Submit** (`testTotpSetupSubmission`): Kiểm tra khả năng gửi form.
  Setup: `formData` đã chứa mã hợp lệ.
  Assertion:
  - Nút "Verify & Enable" không bị disabled.
  - Hàm mock `handleSubmit` được gọi khi submit.
- **Trạng thái Nút** (`testTotpSetupButtonState`): Kiểm tra logic vô hiệu hóa nút bấm.
  Assertion: Nút "Verify & Enable" phải bị disabled trong 2 trường hợp:
  - Mã xác thực chưa được nhập đầy đủ.
  - Prop `verifying={true}` (đang trong quá trình xác minh).

7.2 Page Testing

7.2.1 `admin.test.tsx`

File kiểm thử này tích hợp kiểm tra logic của hai trang quản trị chính: `AdminPolicyPage` và `AdminCleanupPage`. Các component phức tạp (như `AdminShell`, `NumberField`) được giả lập (mock) để cô lập việc kiểm thử vào luồng dữ liệu và logic trang.

- **Admin Policy Page**: Sử dụng các hàm helper để kiểm tra vòng đời của trang chính sách.
  - **Trạng thái Loading** (`testPolicyLoading`): Kiểm tra phản hồi giao diện khi dữ liệu chưa tải xong.
    Setup: Giả lập `adminApi.getPolicy` trả về một Promise đang chờ (pending).
    Assertion: Văn bản thông báo "Đang tải System Policy..." phải hiện diện trong DOM.
  - **Render Dữ liệu** (`testPolicyFormRendering`): Kiểm tra việc hiển thị dữ liệu cấu hình sau khi tải thành công.
    Setup: Giả lập API trả về đối tượng `mockPolicy` (maxFileSizeMB=50, maxValidityDays=7).
    Assertion:
    * Tiêu đề "System Policy" xuất hiện.
    * Các trường nhập liệu hiển thị đúng giá trị từ API (ví dụ: "Max file size" hiển thị 50).
  - **Cập nhật Chính sách** (`testPolicyUpdate`): Kiểm tra tương tác người dùng khi chỉnh sửa và lưu cấu hình.
    Action: Thay đổi giá trị "Max file size" thành 100 và nhấn nút "Lưu System Policy".
    Assertion:
    * Hàm `adminApi.updatePolicy` được gọi với dữ liệu đã cập nhật (`maxFileSizeMB: 100`).
    * Thông báo thành công `toast.success('Updated')` được kích hoạt.
- **Admin Cleanup Page**: Kiểm tra cấu trúc tĩnh của trang dọn dẹp hệ thống.
  Assertion:
  - Tiêu đề "Cleanup file hết hạn" và dòng mô tả về cơ chế tự động phải xuất hiện.
  - Component con `CleanupButton` (đã được mock) phải có mặt trong giao diện.

7.2.2 `dashboard.test.tsx`

File kiểm thử này tập trung vào logic hiển thị của trang Bảng điều khiển (Dashboard). Các hàm helper được sử dụng để kiểm tra luồng dữ liệu từ API và phản ứng của giao diện

đối với các trạng thái xác thực khác nhau.

- **Hiển thị dữ liệu Dashboard** (`testDashboardRendering`): Kiểm tra trạng thái tải và hiển thị thông tin khi API trả về dữ liệu hợp lệ.
  - *Trạng thái chờ*: Văn bản "Loading your dashboard..." phải xuất hiện ngay khi trang khởi tạo.
  - *Hiển thị dữ liệu*: Sau khi API phản hồi, xác minh sự hiện diện của:
    * Lời chào: "Welcome, testuser!".
    * Thông tin cá nhân: "Email: test@example.com".
    * Danh sách tệp tin: Tên file "test.txt", trạng thái "Active" và các chỉ số thống kê (giá trị "1").
- **Điều hướng khi chưa xác thực** (`testUnauthorizedRedirect`): Kiểm tra cơ chế bảo vệ route khi phiên làm việc không hợp lệ.
  Setup: Giả lập API `getUserProfile` trả về lỗi "Unauthorized".
  Assertion: Hàm mock `router.push` được gọi với tham số `'/login'`, buộc người dùng phải đăng nhập lại.
- **Xử lý lỗi tải dữ liệu** (`testFetchError`): Kiểm tra thông báo lỗi hiển thị trên UI khi gặp sự cố mạng hoặc lỗi khác.
  Setup: Giả lập API trả về lỗi "Network error".
  Assertion: Giao diện hiển thị thông báo văn bản cụ thể: "Failed to fetch dashboard data.".

7.2.3 `download.test.tsx`

File kiểm thử này bao quát toàn bộ vòng đời của trang tải xuống công khai, sử dụng các hàm helper riêng biệt để kiểm tra các trạng thái file khác nhau (Active, Expired, Pending, Password-protected).

- **Trạng thái tải trang** (`testDownloadLoading`): Kiểm tra giao diện chờ khi đang lấy thông tin file.
  Assertion: Văn bản "Đang tải..." phải xuất hiện ngay khi truy cập.
- **Thông tin file** (`testDownloadInfo`): Kiểm tra hiển thị metadata khi tải thành công file hợp lệ.

Setup: Giả lập file "test.pdf", kích thước 1024 B.
Assertion:

- Tên file "test.pdf" và thông tin chi tiết "1024 B • application/pdf" được hiển thị.
- Nhãn trạng thái "Khả dụng" xuất hiện.

- **Bảo vệ mật khẩu** (`testPasswordProtectedDownload`): Kiểm tra luồng tải file có mật khẩu.
  - *Setup*: File "secret.txt" có cờ `hasPassword: true`.
  - *Trường hợp thiếu mật khẩu*: Nhấn "Tải xuống" mà không nhập liệu → Hiển thị Alert "Vui lòng nhập mật khẩu.".
  - *Trường hợp đúng mật khẩu*: Nhập "password123" và nhấn tải → Gọi hàm `downloadFile` với token và mật khẩu chính xác → Hiển thị Alert "Đã bắt đầu tải file thành công!".

- **File hết hạn** (`testExpiredFile`): Kiểm tra giao diện khi truy cập file đã quá hạn.
  Setup: File có trạng thái `'expired'`.
  Assertion:
  - Hiển thị nhãn "Hết hạn" và thông báo "File đã hết hạn.".
  - Nút "Tải xuống" bị ẩn khỏi giao diện.

- **File chưa mở** (`testPendingFile`): Kiểm tra giao diện khi truy cập file chưa đến giờ kích hoạt.
  Setup: File có trạng thái `'pending'` và thời gian mở trong tương lai.
  Assertion: Hiển thị nhãn "Chưa mở" và thông báo "Chưa đến thời gian mở khóa".

- **Xem trước file** (`testFilePreview`): Kiểm tra tính năng xem trước (preview) cho các định dạng hỗ trợ.
  Action: Nhấn nút "Tải Preview".
  Assertion:
  - Hàm `loadFilePreview` được gọi.
  - Thẻ hình ảnh (`img`) hiển thị với `src` là blob URL trả về từ API và `alt="image.png"`.

7.2.4 `login-totp.test.tsx`

File kiểm thử này xác minh quy trình xác thực hai yếu tố (2FA), sử dụng các hàm helper để kiểm tra từ việc điều hướng phiên làm việc đến xử lý xác thực thành công và thất bại.

- **Chuyển hướng khi phiên không hợp lệ** (`testInvalidSessionRedirect`): Kiểm tra cơ chế bảo vệ khi người dùng truy cập trang mà không có Challenge ID hợp lệ.
  Setup: Giả lập `getLoginChallengeId` trả về `null`.
  Assertion:
  - Hiển thị thông báo lỗi "Invalid session. Please login again.".
  - Thực hiện điều hướng về trang đăng nhập `/login` ngay lập tức.
- **Render Giao diện** (`testFormRendering`): Kiểm tra hiển thị form khi phiên làm việc hợp lệ.
  Setup: Giả lập có Challenge ID và email trong tham số tìm kiếm.
  Assertion:
  - Tiêu đề "Two-Factor Authentication" xuất hiện.
  - Trường Email được điền sẵn giá trị từ URL (`test@example.com`).
  - Trường nhập mã 6 số ("6-digit code") hiện diện.
- **Xác thực Thành công** (`testSuccessfulVerification`): Kiểm tra luồng xác minh mã TOTP hợp lệ.
  Action: Nhập mã "123456" và nhấn nút "Verify & Login".
  Assertion:
  - Hàm API `loginTotp` được gọi với đúng `cid` và `code`.
  - Điều hướng người dùng đến `/dashboard`.
  - Hiển thị thông báo thành công "Đăng nhập thành công!".
- **Xác thực Thất bại** (`testFailedVerification`): Kiểm tra xử lý lỗi khi mã TOTP không chính xác.
  Setup: Giả lập API từ chối với lỗi "Invalid code".
  Action: Nhập mã và submit form.
  Assertion: Hiển thị thông báo lỗi tương ứng từ API trả về.

7.2.5 `login.test.tsx`

File kiểm thử này bao quát các kịch bản đăng nhập chính, bao gồm đăng nhập thành công, yêu cầu xác thực TOTP, và xử lý lỗi đăng nhập.

- **Đăng nhập thành công** (`testSuccessfulLogin`): Kiểm tra luồng đăng nhập chuẩn khi thông tin xác thực chính xác.
  Setup: Giả lập API `login` trả về token truy cập và thông tin người dùng hợp lệ.
  Action: Nhập Email "test@example.com", Mật khẩu "password" và nhấn nút "Đăng nhập".
  Assertion:
  - Hàm `login` được gọi với đúng payload (email, password).
  - Hệ thống điều hướng đến `/dashboard` thông qua hàm `navigateTo`.
  - Hiển thị thông báo `toast.success` với nội dung "Đăng nhập thành công!".
- **Yêu cầu TOTP** (`testTotpRequirement`): Kiểm tra luồng xử lý khi tài khoản đã bật xác thực hai yếu tố.
  Setup: Giả lập API trả về cờ `requireTOTP: true` và Challenge ID (`cid`).
  Action: Thực hiện thao tác đăng nhập như bình thường.
  Assertion: Hệ thống phát hiện yêu cầu TOTP và chuyển hướng người dùng đến trang nhập mã xác thực (`/login/totp`) thông qua `router.push`.
- **Đăng nhập thất bại** (`testLoginFailure`): Kiểm tra phản hồi giao diện khi thông tin đăng nhập sai hoặc lỗi hệ thống.
  Setup: Giả lập API `login` trả về lỗi (rejected promise).
  Action: Nhập thông tin và submit form.
  Assertion: Hàm `toast.error` được gọi với thông báo lỗi cụ thể từ API ("Login failed").

7.2.6 `register.test.tsx`

File kiểm thử này xác minh toàn bộ quy trình đăng ký tài khoản, từ việc xử lý thành công, kiểm tra tính hợp lệ của mật khẩu ngay tại phía client, đến việc xử lý lỗi từ server trả về.

- **Đăng ký thành công** (`testSuccessfulRegistration`): Kiểm tra luồng người dùng tạo tài khoản hợp lệ.

Setup: Giả lập API `register` trả về kết quả thành công.
Action: Nhập đầy đủ thông tin vào các trường "Tên đăng nhập", "Email", "Mật khẩu", "Xác nhận mật khẩu" (khớp nhau) và nhấn nút "Đăng ký".
Assertion:

- Hàm API `register` được gọi với object chứa thông tin người dùng chính xác.
- Hệ thống điều hướng người dùng về trang đăng nhập (`'/login'`) thông qua `router.push`.
- Hiển thị thông báo thành công `toast.success`.

- **Lỗi mật khẩu không khớp** (`testPasswordMismatch`): Kiểm tra logic validation phía client.
Action: Nhập "Mật khẩu" là 'password' nhưng "Xác nhận mật khẩu" là 'different' và submit form.
Assertion:
  - Hàm API `register` **không** được gọi (ngăn chặn request không hợp lệ).
  - Hiển thị thông báo lỗi cụ thể: "Mật khẩu và xác nhận mật khẩu không khớp.".

- **Đăng ký thất bại** (`testRegistrationFailure`): Kiểm tra phản hồi giao diện khi server gặp sự cố.
Setup: Giả lập API `register` trả về lỗi (rejected).
Action: Submit form với dữ liệu hợp lệ.
Assertion: Hiển thị thông báo lỗi `toast.error` chứa nội dung lỗi từ server trả về ("Registration failed").

7.2.7 `upload.test.tsx`

File kiểm thử này sử dụng các hàm helper để bao quát toàn bộ quy trình tải lên tệp tin, từ việc render giao diện, chọn file, kiểm tra tính hợp lệ (validation) đến xử lý phản hồi từ API.

- **Render Form** (`testUploadFormRendering`): Kiểm tra cấu trúc ban đầu của trang upload.
Assertion: Tiêu đề "Upload File Bảo Mật" và hướng dẫn bước "1. Chọn file" phải hiện diện trong DOM.

- **Chọn File (`testFileSelection`)**: Kiểm tra tương tác người dùng khi tải file từ máy tính.
  Action: Kích hoạt sự kiện `change` trên input file ẩn với một file giả lập (`mockFile`).
  Assertion: Tên file "test.txt" phải hiển thị trên giao diện sau khi chọn.

- **Validation: Thiếu File (`testMissingFileValidation`)**: Kiểm tra ràng buộc bắt buộc chọn file.
  Action: Nhấn nút "Upload ngay" mà không chọn file.
  Assertion: Hiển thị thông báo lỗi `toast.error` với nội dung "Vui lòng chọn một file để upload.".

- **Validation: Mật khẩu ngắn (`testPasswordValidation`)**: Kiểm tra ràng buộc độ dài mật khẩu khi tính năng bảo vệ được bật.
  Action:
  - Bật toggle "Bảo vệ bằng mật khẩu".
  - Nhập mật khẩu ngắn ("123") vào ô input.
  - Nhấn nút "Upload ngay".

  Assertion: Hiển thị thông báo lỗi "Mật khẩu phải có tối thiểu 6 ký tự.".

- **Upload Thành công (`testSuccessfulUpload`)**: Kiểm tra luồng tải lên hợp lệ.
  Setup: Giả lập API `uploadFile` trả về kết quả thành công kèm token chia sẻ.
  Action: Chọn file và nhấn upload.
  Assertion:
  - Gọi API `uploadFile`.
  - Hiển thị thông báo `toast.success("Upload thành công!")`.
  - Giao diện hiển thị kết quả thành công và token chia sẻ ("token123").

- **Upload Thất bại (`testUploadError`)**: Kiểm tra xử lý lỗi mạng hoặc lỗi server.
  Setup: Giả lập API trả về lỗi ("Network Error").
  Action: Thực hiện upload.
  Assertion: Hiển thị thông báo lỗi tương ứng từ API.

8 CI/CD Pipeline & Deployment

8.1 Proposed CI/CD Pipeline (Quy trình đề xuất)

Do cấu hình CI/CD chưa được tích hợp sẵn trong mã nguồn hiện tại, nhóm đề xuất quy trình Tích hợp và Triển khai liên tục (CI/CD) như sau để tự động hóa việc kiểm thử và đóng gói Frontend:

1. **Source Code Management**: Clone repository từ GitHub về môi trường runner.
2. **Dependency Resolution**: Chạy lệnh `npm install` để cài đặt các thư viện phụ thuộc.
3. **Automated Testing**: Thực thi bộ kiểm thử bằng lệnh `npm test`. Quy trình sẽ dừng lại ngay lập tức nếu có bất kỳ test case nào thất bại.
4. **Artifact Building**: Nếu kiểm thử thành công, hệ thống chuyển sang máy build (build machine) để đóng gói **Frontend** thành Docker image.
5. **Publish Artifact**: Docker image (Frontend) hoàn chỉnh được đẩy (publish) lên GitHub Container Registry (GHCR).

8.2 Deployment Instructions (Docker)

Artifact Frontend đã được đóng gói sẵn sàng trên GHCR, tuy nhiên người vận hành cần thực hiện đăng nhập trước khi hệ thống có thể kéo (pull) image về.

1. Cấu hình GitHub Personal Access Token (PAT)

Để có quyền truy cập vào GHCR:

- Truy cập GitHub: **Settings** → **Developer settings** → **Personal access tokens** → **Tokens (classic)**.
- Tạo token mới với quyền (scope): `read:packages`.
- *Lưu ý quan trọng*: Nếu tài khoản thuộc tổ chức sử dụng SSO (SAML), bắt buộc phải ủy quyền (Authorize) cho token thông qua nút "Configure SSO".

2. Đăng nhập Docker Client

Chạy lệnh sau trong terminal để xác thực (thay thế `YOUR_USERNAME` bằng GitHub username và nhập PAT vừa tạo làm mật khẩu):

```
docker login ghcr.io -u YOUR_USERNAME
```

3. Khởi chạy hệ thống

Di chuyển vào thư mục dự án `file-sharing-fe-web`, thêm `.env` như mô tả trong `example.env` nhằm chỉnh URL của backend và chạy lệnh dưới đây. Cấu hình này sẽ **tải (pull) image Frontend** từ GHCR, đồng thời **build cục bộ** các service còn lại (Backend và Nginx):

```
docker compose -f docker-compose.prod.yaml up --build
```

9 Conclusion

Dự án đã hoàn thành các mục tiêu cốt lõi: xây dựng giao diện hỗ trợ SSR, luồng xác thực (login, TOTP), upload tệp an toàn, dashboard theo dõi trạng thái, và trang truy cập công khai. Hệ thống đã có bộ kiểm thử cho các component và trang quan trọng, giúp kiểm soát chất lượng khi mở rộng.

9.1 Khó khăn / hạn chế chưa giải quyết

- Chưa có cơ chế OTP riêng cho từng lần tải xuống (mới dừng ở bảo vệ mật khẩu và email whitelist).
- Xử lý lỗi mạng/chậm chưa có retry/backoff và thông điệp vi mô (microcopy) chi tiết cho từng trường hợp.
- Progress upload mới ở mức cơ bản; chưa hỗ trợ chunked upload, resume cho tệp lớn hoặc mạng chập chờn.
- UX đa thiết bị chưa được kiểm thử đầy đủ trên mobile cũ/trình duyệt ít phổ biến; accessibility (ARIA, focus state) mới ở mức tối thiểu.
- Hiệu năng: chưa tối ưu code-splitting sâu theo route/component, chưa prefetch thông minh cho các luồng phổ biến.

9.2 Hướng phát triển tương lai

- **OTP cho file:** Thêm lớp xác thực OTP một lần cho mỗi lượt tải, kèm rate-limit và TTL ngắn; cân nhắc email/SMS nếu backend hỗ trợ.
- **Upload nâng cao:** Chunked upload + resume, hiển thị tốc độ và ước tính thời gian; retry/backoff và phát hiện mạng yếu.
- **Bảo mật & quyền riêng tư:** Chính sách chia sẻ chi tiết hơn (role-based, link scope, giới hạn lượt tải, geo/IP allowlist); cảnh báo khi file sắp hết hạn.
- **UX & Accessibility:** Bổ sung trạng thái tải/lỗi rõ ràng, thông điệp vi mô cho từng lỗi HTTP; hoàn thiện ARIA/focus-trap và kiểm thử trên thiết bị cũ.

- **Hiệu năng:** Tối ưu code-splitting sâu theo route, prefetch thông minh, giảm chi phí hydration; rà soát hình ảnh/icon để giảm kích thước bundle.

Phần mở rộng này giúp frontend sẵn sàng cho các tình huống bảo mật cao hơn, trải nghiệm ổn định hơn và dễ bảo trì khi hệ thống phát triển.