

ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

**Triển khai tích hợp các giải pháp nâng cao an toàn
cho phân hệ xác thực danh tính trên dịch vụ Web**

Đặng Phương Nam

nam.dp194335@sis.hust.edu.vn

Ngành Công nghệ thông tin và truyền thông

Giảng viên hướng dẫn: ThS. Bùi Trọng Tùng

Chữ kí GVHD

Khoa:

Kỹ thuật máy tính

Trường:

Công nghệ thông tin và Truyền thông

HÀ NỘI, 06/2023

LỜI CẢM ƠN

Trước tiên, tôi xin được gửi lời cảm ơn đến Thạc sỹ Bùi Trọng Tùng đã tận tâm hướng dẫn và hỗ trợ trong quá trình nghiên cứu và viết đề án này. Sự hướng dẫn và khuyến khích từ thầy đã đóng góp quan trọng vào sự hoàn thiện của đề tài này.

Tôi cũng muốn gửi lời cảm ơn sâu sắc đến bạn bè và người thân của tôi. Sự hỗ trợ và động viên từ phía họ đã là nguồn động lực quan trọng trong quá trình hoàn thiện đề án này.

Cuối cùng, tôi muốn cảm ơn Trường Công nghệ Thông tin và Truyền thông trực thuộc Đại học Bách Khoa Hà Nội và khoa Kỹ thuật Máy tính đã tạo điều kiện và cung cấp nguồn lực để tôi hoàn thành đề án này.

TÓM TẮT NỘI DUNG ĐỒ ÁN

Vấn đề xác thực danh tính trên ứng dụng web là một chủ đề quan trọng và ngày càng được quan tâm. Xác thực danh tính đảm bảo rằng người dùng có quyền truy cập và thực hiện các hoạt động trên ứng dụng web dựa trên danh tính của họ. Điều này đóng vai trò quan trọng trong việc bảo vệ thông tin cá nhân, dữ liệu và đảm bảo an toàn cho người dùng.

Cho đến nay, giải pháp xác thực bằng mật khẩu truyền thống vẫn rất phổ biến, tuy nhiên đang dần trở nên kém bảo mật do các nguy cơ như mật khẩu bị đánh cắp hay tấn công vét cạn để đoán nhận được mật khẩu. Ngoài ra sử dụng mật khẩu cũng rất bất tiện cho người sử dụng, khi độ dài và số lượng mật khẩu cần quản lý tăng lên. Chính vì thế mà nhiều giải pháp xác thực nâng cao ra đời như đăng nhập một lần (Single Sign On) giúp việc xác thực và quản lý trở nên đơn giản, tiện lợi, hay là xác thực nhiều bước (Multifactor Authentication) giúp nâng cao tính an toàn bảo mật cho quá trình xác thực.

Mục tiêu của đề tài là xây dựng module tích hợp các giải pháp xác thực nâng cao giúp giải quyết các vấn đề phổ biến với phương pháp xác thực mật khẩu truyền thống. Module được xây dựng dưới dạng gói phần mềm độc lập giúp dễ dàng tích hợp vào các ứng dụng web. Điều này cho phép phát triển các ứng dụng web đơn giản và nhanh chóng hơn.

Bộ cục đồ án tốt nghiệp này được tổ chức thành 6 chương chính. Chương một giới thiệu bài toán xác thực mật khẩu và sự cần thiết của việc tích hợp các giải pháp xác thực nâng cao cho phân hệ xác thực danh tính trên nền tảng web. Chương hai sẽ trình bày các khảo sát hiện trạng của bài toán và đưa ra các định hướng giải pháp và chức năng. Trong chương 3, tôi sẽ giới thiệu về các công nghệ được sử dụng trong giải pháp nói trên. Chương 4 sẽ đi sâu và chi tiết chức năng của giải pháp sử dụng các loại biểu đồ trực quan khác nhau. Chương 5 sẽ trình bày khái quát về kết quả đạt được cũng như những ca kiểm thử của đồ án. Chương cuối cùng của đồ án điếm qua các thách thức đặc thù trong quá trình phát triển và giải pháp, đóng góp nổi bật của đề tài; trong chương này, tôi cũng sẽ đưa ra đánh giá về kết quả của đề tài và hướng phát triển trong tương lai.

ABSTRACT

Authentication on the Web platform has always been a crucial topic and is garnering more and more attention recently. The act of authenticating itself means to make sure users actually possess the authority and privileges required to perform certain actions on the web application based on their identities. This play an important part to make sure the users' experiences on the application safe and secure.

Up until now, password-based authentication has always been the most dominant entry when it comes to identifying users online. This, however, poses several security threats such as password theft or employment of bruteforce attacks to figure out users' passwords. Apart from all that, passwords themselves are very inconvenient to use and difficult to maintain when the number and complexity of them increases. This leads to the advent of a number of alternatives to address the issues of password authentication such as Single Sign on or Multifactor Authentication, just to name a few. These alternative solutions help tackle different problems and when working together, they massively enhance the secureness and ease of use of the authentication process.

The goal of this thesis is to build a module which integrate various alternative authentication methods to address some of the most common issues with the conventional password-based authentication. The module will be developed as a isolated library packages to provide the developers with the ability to include it in their own web application, which wil greatly boost the speed and reduce the complexity of the development process.

The outline of this graduation thesis consists of 6 main chapters. The fist chapter will introduce the common problems of password-based authentication thus solidify the importance of integrating advanced authentication solutions on the web platform. The second chapter will survey the current state of the problems and propose potential solutions and functionalities. In chapter 3, I will briefly introduce all the technologies used in the chosen solution scheme. Chapter 4 will discuss in great details the aforementioned functionalities and features using various diagram types. In chapter 5, I will go through the results of the development process in brief and display test cases for the main use cases. The final chapter will selectively highlight the unique challenges I faced in development and the remarkable contributions that this thesis have to those challenges; in this chapter, I will also sum up the results of this graduation thesis, about what it had and hadn't achieved and the directions for future developemnt.

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....	1
1.1 Đặt vấn đề.....	1
1.2 Mục tiêu và phạm vi đề tài.....	2
1.3 Phân tích bài toán và định hướng giải pháp	3
1.4 Bố cục đồ án	4
CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH CHỨC NĂNG.....	5
2.1 Khảo sát hiện trạng	5
2.2 Tổng quan chức năng	6
2.2.1 Tác nhân hệ thống	6
2.3 Phân tích chi tiết các ca sử dụng.....	6
2.3.1 Đối với tác nhân Người sử dụng hệ thống	6
2.3.2 Đối với tác nhân Quản trị viên.....	14
CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG.....	15
3.1 Giới thiệu về các công nghệ sử dụng cho giải pháp đăng nhập một lần.....	15
3.1.1 Tổng quan về giao thức OAuth 2.0	15
3.1.2 Tổng quan về Google OAuth 2.0.....	16
3.1.3 Tổng quan về Github OAuth 2.0	17
3.2 Giới thiệu về các công nghệ sử dụng trong giải pháp xác thực đa bước.....	17
3.2.1 Giới thiệu về mật khẩu một lần dựa trên thời gian	17
3.2.2 Google Authenticator	18
3.2.3 Tổng quan về giao thức WebAuthn.....	19
3.3 Giới thiệu về các công nghệ được sử dụng để xây dựng module	19
3.3.1 Giới thiệu về framework NestJS.....	19
3.3.2 Giới thiệu về framework VueJS	21

3.3.3 Giới thiệu về MongoDB	21
3.3.4 Giới thiệu về Redis.....	22
3.3.5 Tổng quan về Restful API	22
CHƯƠNG 4. THIẾT KẾ CHỨC NĂNG HỆ THỐNG.....	23
4.1 Thiết kế kiến trúc.....	23
4.1.1 Lựa chọn kiến trúc phần mềm	23
4.1.2 Thiết kế tổng quan.....	24
4.1.3 Thiết kế chi tiết các gói.....	25
4.2 Thiết kế chức năng.....	29
4.2.1 SSO sử dụng Google OAuth.....	29
4.2.2 Thiết lập và xác thực với TOTP	30
4.2.3 Thiết lập giao thức WebAuthn	31
4.2.4 Xác thực với WebAuthn.....	32
4.3 Thiết kế cơ sở dữ liệu	33
4.3.1 Thiết kế tổng quan.....	33
4.3.2 Đặc tả	33
4.4 Thiết kế giao diện	36
4.4.1 Trang đăng nhập	36
4.4.2 Trang thiết lập TOTP.....	36
4.4.3 Trang xác thực TOTP	37
4.4.4 Trang thiết lập WebAuthn	37
4.4.5 Trang xác thực WebAuthn.....	37
CHƯƠNG 5. CÀI ĐẶT VÀ THỬ NGHIỆM	38
5.1 Xây dựng ứng dụng.....	38
5.1.1 Thư viện và công cụ sử dụng	38
5.1.2 Kết quả đạt được	38

5.2 Kiểm thử.....	39
5.2.1 Môi trường kiểm thử	39
5.2.2 Kiểm thử chức năng SSO sử dụng Google OAuth.....	39
5.2.3 Kiểm thử chức năng SSO sử dụng Github OAuth	40
5.2.4 Kiểm thử chức năng Thiết lập TOTP với Google Authenticator	40
5.2.5 Kiểm thử chức năng Xác thực với Google Authenticator	41
5.2.6 Kiểm thử chức năng Thiết lập giao thức WebAuthn.....	42
5.2.7 Kiểm thử chức năng Xác thực với WebAuthn.....	43
CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	44
6.1 Các giải pháp và đóng góp nổi bật	44
6.1.1 Bài toán xác thực với thiết bị di động	44
6.1.2 Bài toán tách module thành thư viện	47
6.2 Kết luận	50
6.3 Hướng phát triển.....	51
TÀI LIỆU THAM KHẢO.....	52

DANH MỤC HÌNH VẼ

Hình 2.1	Biểu đồ ca sử dụng với tác nhân Người dùng hệ thống	6
Hình 2.2	Biểu đồ ca sử dụng đối với tác nhân Quản trị viên	14
Hình 3.1	Hoạt động của OAuth 2.0	16
Hình 4.1	Thiết kế kiến trúc tổng quan	23
Hình 4.2	Biểu đồ gói tổng quan	24
Hình 4.3	Thiết kế gói controllers	25
Hình 4.4	Thiết kế gói guards	26
Hình 4.5	Thiết kế gói services	26
Hình 4.6	Thiết kế gói schemas	27
Hình 4.7	Thiết kế gói helpers	27
Hình 4.8	Thiết kế gói strategies	28
Hình 4.9	Biểu đồ tuần tự đăng nhập SSO	29
Hình 4.10	Biểu đồ tuần tự thiết lập TOTP và xác thực TOTP	30
Hình 4.11	Biểu đồ tuần tự thiết lập giao thức WebAuthn	31
Hình 4.12	Biểu đồ tuần tự xác thực sử dụng với WebAuthn	32
Hình 4.13	Thiết kế cơ sở dữ liệu	33
Hình 4.14	Mockup đăng nhập	36
Hình 4.15	Mockup trang thiết lập TOTP	36
Hình 4.16	Mockup trang xác thực TOTP	37
Hình 4.17	Mockup đăng ký thiết bị WebAuthn	37
Hình 4.18	Mockup xác thực WebAuthn	37
Hình 6.1	Luồng đăng ký thiết bị của WebAuthn	46
Hình 6.2	Luồng xác thực của WebAuthn	46
Hình 6.3	Đoạn mã sinh tùy chọn đăng ký cho WebAuthn	47
Hình 6.4	Biến môi trường	49
Hình 6.5	Interface cho cài đặt module động	49
Hình 6.6	File JSON chứa cls config	50

DANH MỤC BẢNG BIỂU

Bảng 2.1	Các tác nhân hệ thống	7
Bảng 2.2	Bảng đặc tả ca sử dụng SSO sử dụng Google OAuth	8
Bảng 2.3	Bảng đặc tả ca sử dụng SSO sử dụng Github OAuth	9
Bảng 2.4	Bảng đặc tả ca sử dụng thiết lập TOTP với Google Authenticator	10
Bảng 2.5	Bảng đặc tả ca sử dụng Xác thực với Google Authenticator	11
Bảng 2.6	Bảng đặc tả ca sử dụng thiết lập giao thức WebAuthn	12
Bảng 2.7	Bảng đặc tả ca sử dụng xác thực với giao thức WebAuthn	13
Bảng 2.8	Bảng đặc tả ca sử dụng update cài đặt xác thực	14
Bảng 4.1	Đặc tả Collection OtpInfo	33
Bảng 4.2	Đặc tả Collection User	34
Bảng 4.3	Đặc tả Collection ProviderInfo	35
Bảng 4.4	Đặc tả Collection OtpInfo	35
Bảng 5.1	Danh sách thư viện và công cụ sử dụng	38
Bảng 5.2	Thống kê ứng dụng	38
Bảng 5.3	Môi trường kiểm thử	39
Bảng 5.4	Ca kiểm thử chức năng đăng nhập một lần sử dụng Google OAuth	40
Bảng 5.5	Ca kiểm thử chức năng đăng nhập một lần sử dụng Github OAuth	40
Bảng 5.6	Ca kiểm thử chức năng thiết lập TOTP với Google Authenticator	41
Bảng 5.7	Ca kiểm thử xác thực TOTP với Google Authenticator	41
Bảng 5.8	Ca kiểm thử thiết lập giao thức WebAuthn	42
Bảng 5.9	Ca kiểm thử xác thực giao thức WebAuthn	43

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ	Ý nghĩa
ĐATN	Đồ án tốt nghiệp
API	Giao diện lập trình ứng dụng (Application Programming Interface)
CRUD	Create, read, update ,delete
DI	kỹ thuật lập trình nội xạ phụ thuộc (Dependencies Injection)
FIDO	Hiệp hội công nghiệp mở FIDO được thành lập với sứ mệnh "giảm sự phụ thuộc quá mức của thế giới vào mật khẩu" (Fast Identity Online)
HTML	Ngôn ngữ đánh dấu siêu văn bản (HyperText Markup Language)
JWT	Chuỗi mã hóa nguồn gốc ban đầu là một chuỗi JSON (JSON Web Token)
MFA	Xác thực đa yếu tố (Multifactor Authentication)
REST	Chuyển giao trạng thái đại diện(Representational State Transfer)
SSO	Đăng nhập một lần (Single Sing On)
URI	Định dạng tài nguyên thống nhất (Unified Resource Indentifier)
W3C	Tổ chức tiêu chuẩn quốc tế chính cho World Wide Web (World Wide Web Consortium)

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Hiện nay, phương pháp xác thực phổ biến nhất trên nền tảng web vẫn là sử dụng một bộ tài khoản/mật khẩu truyền thống. Mặc dù phổ biến, phương pháp này vẫn có rất nhiều nhược điểm dẫn đến sự bất tiện và không an toàn cho người dùng. Một trong những vấn đề tiêu biểu nhất là vấn đề về quản lý mật khẩu cho nhiều tài khoản khác nhau. Để đảm bảo tính an toàn bảo mật online, người dùng được khuyến nghị là nên sử dụng mật khẩu phức tạp, khó đoán và có đầy đủ các loại ký tự từ chữ cái viết thường, viết hoa, ký tự số, ký tự đặc biệt,... và có độ dài tương đối. Các mật khẩu phức tạp như thế này tuy an toàn, nhưng lại khó nhớ và khó quản lý khi số lượng tăng cao, dẫn đến rất nhiều người vẫn lựa chọn sử dụng mật khẩu đơn giản, hay sử dụng chung mật khẩu cho nhiều dịch vụ web khác nhau. Trung bình có đến 2/3 dân số Hoa Kỳ sử dụng lại mật khẩu cho nhiều tài khoản trải rộng nhiều trang web khác nhau, và các mật khẩu đơn giản như "123456" hay "password" được sử dụng bởi 23 triệu người dùng toàn cầu [1].

Một trong những giải pháp cho vấn đề quản lý này là mô hình xác thực một lần (Single Sign on). Xác thực một lần là mô hình xác thực cho phép người dùng đăng nhập một lần bằng một thông tin xác thực duy nhất và truy cập vào các dịch vụ web khác nhau mà không cần phải thực hiện xác thực lại. Mô hình xác thực một lần mang lại rất nhiều lợi ích cho người dùng, tiêu biểu nhất có thể kể đến giảm số lượng tài khoản/mật khẩu mà người dùng phải ghi nhớ, giảm số lượng lần phải nhập tài khoản/mật khẩu và tập trung thông tin người dùng lại cho một dịch vụ cung cấp SSO duy nhất. Tuy nhiên mô hình này cũng có các hạn chế về mặt bảo mật, vì khi tập trung xác thực của nhiều dịch vụ lại cho một dịch vụ duy nhất, rủi ro bảo mật sẽ tăng cao khi mật khẩu bị lộ hoặc bị đánh cắp dẫn kẻ tấn công có quyền truy cập vào tất cả các dịch vụ khác nhau phụ thuộc vào mật khẩu đó. Theo thống kê của Viettel Threat Intelligence, trong năm 2022 có 216,848,984 thông tin tài khoản bị đăng nhập bị đánh cắp[2]. Nguyên nhân chủ yếu dẫn đến vấn đề này là việc xác thực mật khẩu của các mô hình đăng nhập một lần vẫn còn sử dụng xác thực mật khẩu một yếu tố.

Để có thể nâng cao tính an toàn bảo mật cho các mô hình đăng nhập một lần, giải pháp ứng dụng mô hình xác thực đa yếu tố là cần thiết. Xác thực đa yếu tố là mô hình yêu cầu nhiều hơn một phương thức xác thực từ các danh mục thông tin độc lập với nhau để xác minh danh tính người dùng khi đăng nhập hoặc thực hiện các giao dịch khác. Yếu tố thứ 2 của mô hình này có thể là yếu tố tài sản như thiết

bị di động, thẻ bảo mật, hoặc yếu tố tri thứ như câu hỏi bí mật,... Mô hình xác thực đa yếu tố này khi được tích hợp có thể giảm thiểu các rủi ro bảo mật vì nó yêu cầu người dùng cung cấp các yếu tố thứ 2, thứ 3 trước khi họ được xác nhận là đáng tin cậy từ đó giảm thiểu các mối nguy hại khi mật khẩu bị lộ hoặc bị trộm, vì các yếu tố xác thực bổ sung như thiết bị hay sinh trắc khác với mật khẩu thông thường, rất khó để bị làm giả hoặc đánh cắp.

Việc triển khai mô hình xác thực đa yếu tố với mô hình đăng nhập một lần mang lại những lợi ích tuyệt vời. Đăng nhập một lần mang lại yếu tố tiện lợi còn xác thực đa yếu tố thiên về bảo mật. Sự kết hợp giữa Single Sign On và Multifactor Authentication giúp loại bỏ nhu cầu sử dụng nhiều mật khẩu, nâng cao trải nghiệm người dùng đồng thời giúp tăng cường tính an toàn bảo mật trong quá trình sử dụng các dịch vụ web

1.2 Mục tiêu và phạm vi đề tài

Mục tiêu của đề tài là xây dựng một module phần mềm phân hệ xác thực danh tính trên dịch vụ Web trong đó tích hợp 2 mô hình là đăng nhập một lần và xác thực đa yếu tố. Để thực hiện được mục tiêu này, đề án cần thực hiện được các nhiệm vụ sau.

Thứ nhất, tôi cần tìm hiểu về công nghệ Single Sign On. Hoàn thành nhiệm vụ này cho tôi các cơ sở lý thuyết cần thiết và sự hiểu biết về các hệ thống Single Sign On hiện có để có thể đưa ra giải pháp tích hợp tính năng xác thực một lần vào trong module của mình. Sau khi cân nhắc phạm vi của đề tài là một đề tài cá nhân, tôi nhận định rằng định hướng thích hợp nhất nên có những ưu điểm là chi phí rẻ, đơn giản và tiện dụng mà vẫn đảm bảo các tính năng cần thiết của một hệ thống đăng nhập một lần.

Thứ hai, tôi cần tìm hiểu về các mô hình và công nghệ xác thực đa yếu tố. Nhiệm vụ này giúp tôi nắm chắc hơn nền tảng lý thuyết của các hệ thống ứng dụng mô hình xác thực đa yếu tố này và các cách thức để có thể cài đặt nó hoạt động trơn tru trong module xác thực của mình. Bên cạnh đó, nhiệm vụ này cũng sẽ trả lời câu hỏi nên lựa chọn sử dụng yếu tố nào để làm yếu tố xác thực thứ 2 của tôi. Sau khi xem xét đến phạm vi của đề tài, tôi tin rằng giải pháp được lựa chọn nên có các tính chất như: chi phí rẻ, các yếu tố xác thực nâng cao nên là các yếu tố người dùng sẵn có để tiện lợi cho người sử dụng.

Sau khi hoàn thành 2 nhiệm vụ trên và có được kiến thức nền tảng và các định hướng để phát triển các chức năng xác thực nâng cao, nhiệm vụ tiếp theo là xây dựng một module xác thực được tích hợp các giải pháp xác thực nâng cao kể trên dưới dạng một thư viện hoặc một gói phần mềm độc lập. Nhiệm vụ này khi hoàn

thành sẽ giúp biến mô hình xác thực nâng cao này trở nên dễ dàng để tích hợp vào các ứng dụng web hơn.

1.3 Phân tích bài toán và định hướng giải pháp

Trong phần này, tôi sẽ trình bày cụ thể về bài toán và các định hướng giải pháp và các vấn đề của chúng và áp dụng vào mục tiêu và phạm vi đề tài đã phân tích ở phía trên để lựa chọn các giải pháp thích hợp nhất cho từng nhiệm vụ cụ thể.

Đối với nhiệm vụ đầu tiên là tìm hiểu về mô hình xác thực một lần và đưa ra phương hướng để tích hợp mô hình vào trong module xác thực, tôi đã đề ra được 2 định hướng phù hợp với phạm vi đề tài và có tiềm năng trở thành giải pháp nhất: xây dựng mô hình dựa trên các dự án mã nguồn mở có sẵn và sử dụng dịch vụ đang có. Sau khi cân nhắc kỹ ưu nhược điểm của 2 định hướng, tôi nhận định rằng hướng đi thứ nhất có một nhược điểm lớn là sẽ cần phải tốn tài nguyên để triển khai và quản lý cũng như tùy chỉnh để có thể hoạt động ổn định, chính vì thế, để phù hợp hơn với phạm vi và mục tiêu đề tài, tôi quyết định sẽ sử dụng các dịch vụ sẵn có đang được cung cấp bởi bên thứ 3. Cụ thể hơn, sau khi nghiên cứu và so sánh các giải pháp cung cấp dịch vụ này, tôi đã quyết định lựa chọn cài đặt giải pháp Single Sign On với 2 dịch vụ phổ biến là Google OAuth 2.0 và Github OAuth.

Còn về nhiệm vụ thứ 2 là tìm hiểu về mô hình xác thực đa bước. Sau khi hoàn thiện nó, tôi đề ra được 2 lựa chọn để làm yếu tố xác thực nâng cao. Yếu tố tài sản và yếu tố tri thức. Trong 2 lựa chọn này, tôi tin rằng sử dụng yếu tố tài sản sẽ đảm bảo tính an toàn bảo mật tốt hơn yếu tố tri thức, vì khác với yếu tố tri thức có rủi ro bị đánh cắp hay rò rỉ, yếu tố tài sản như thiết bị phần cứng hay sinh trắc rất khó để bị làm giả hoặc đánh cắp. Tuy nhiên, trong các yếu tố tài sản, không phải yếu tố nào cũng phù hợp với phạm vi mà đề tài đề ra. Để đảm bảo tính tiện lợi và chi phí rẻ, tôi tin rằng sử dụng thiết bị di động thông minh làm yếu tố xác thực là lựa chọn thích hợp nhất vì smartphones là một thiết bị rất phổ biến hiện nay. Cụ thể hơn, đề tài sẽ sử dụng 2 giải pháp là TOTP với ứng dụng Google Authenticator và Passkey với tiêu chuẩn WebAuthn trên thiết bị di động. Cả 2 đều là các giải pháp hợp lý với mục tiêu và phạm vi đề tài vì chi phí cài đặt và triển khai thấp, đồng thời vẫn cung cấp được tính an toàn bảo mật của phương pháp xác thực đa yếu tố.

Cuối cùng về khía cạnh công nghệ, đề tài định hướng sẽ sử dụng framework NestJS, là một framework hỗ trợ phát triển ứng dụng web sử dụng ngôn ngữ lập trình phổ biến Typescript và hệ sinh thái của nó, bao gồm cả các thư viện và cài đặt của các giải pháp xác thực nâng cao sử dụng ngôn ngữ lập trình Typescript. NestJS là một framework có tính cấu trúc cao và có các khuôn mẫu phát triển vượt trội giúp cho việc thực thi cài đặt các tính năng đơn giản mà vẫn để lại nhiều chỗ

trông cho mục đích mở rộng trong tương lai, tạo tiền đề cho việc phát triển mô hình thành một module tích hợp các giải pháp xác thực nâng cao đơn giản và hiệu quả.

1.4 Bố cục đề án

Phần còn lại của báo cáo đề án tốt nghiệp này được tổ chức như sau.

Chương 2 trình bày về các khảo sát hiện trạng của bài toán xác thực hiện nay, từ đó đưa ra được các khảo sát về chức năng mà module cần phải có. Chương 2 đồng thời cũng sẽ đi sâu vào các usecase quan trọng và phân tích được các đặc tả usecase.

Chương 3, tôi sẽ giới thiệu về các công nghệ chính được sử dụng trong quá trình phát triển module sản phẩm của đề án. Trong chương này, các công nghệ như NestJS hay VueJS sẽ được giới thiệu, đồng thời các loại giao thức hoặc giải pháp được ứng dụng tích hợp vào module như OAuth hay WebAuthn cũng được mô tả.

Chương 4 là chi tiết chức năng của module. Trong chương này tôi sẽ tập trung vào đặc tả các chức năng chính mà module cung cấp sử dụng các loại biểu đồ như biểu đồ tuần tự, biểu đồ thực thể quan hệ. Đồng thời trong chương này mockup frame cho giao diện cũng được giới thiệu.

Chương 5 trình bày chi tiết về kết quả đạt được sau quá trình phát triển đề tài, môi trường kiểm thử và các ca kiểm thử được áp dụng cho các use case lớn.

Chương 6 là phần đánh giá kết quả đạt được. Trong chương này, tôi điểm qua các bài toán, thách thức đặc thù cũng như các giải pháp, đóng góp nổi bật mà đề tài mang lại. Tôi cũng sẽ điểm qua các khối chức năng lớn của đề tài, đánh giá những kết quả đã đạt được và chưa làm được. Cuối cùng, tôi sẽ đưa ra hướng phát triển trong tương lai của đề tài.

CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH CHỨC NĂNG

2.1 Khảo sát hiện trạng

Hiện nay, nhu cầu sử dụng của người dùng website đối với các phương pháp xác thực thay thế và nâng cao đang tăng trưởng nhanh.

Ví dụ, mức độ phổ biến của Multifactor Authentication đối với người sử dụng web phổ thông tăng mạnh so với quá khứ, từ năm 2017 đến năm 2021 đã tăng khoảng 178%. Đối với Single Sign On, số liệu chỉ ra thị trường cho các sản phẩm cung cấp giải pháp Single Sign On tăng lên đến 1,6 tỉ USD vào năm 2021[3].

Đối với giải pháp xác thực đăng nhập một lần (Single Sign On), Google là một trong những nền tảng cung cấp cơ chế đăng nhập một lần phổ biến nhất hiện nay thông qua máy chủ Google OAuth 2.0. Google OAuth là một dịch vụ mạnh mẽ, được Google sử dụng để phục vụ chức năng đăng nhập một lần với tài khoản của Google. Chỉ bằng một lần đăng nhập, người dùng ngay lập tức có quyền truy cập đến hàng chục các ứng dụng, dịch vụ khác mà google cung cấp như Gmail, Google workspace, Google drive, youtube,... Tất cả các thông tin sử dụng của người dùng sẽ được quản lý tập trung tại một tài khoản Google duy nhất, rất tiện dụng cho người dùng cuối. Google cũng cung cấp các API và cài đặt cần thiết để một ứng dụng bên thứ ba có thể ủy quyền xác thực cho máy chủ cung cấp dịch vụ xác thực của Google, mở ra rất nhiều khả năng cho các ứng dụng tham gia vào mạng lưới đăng nhập một lần của Google.

Github cũng là một nền tảng cung cấp dịch vụ SSO phổ biến khác thông qua Github OAuth. Dịch vụ OAuth của Github cũng cung cấp các tính năng tương tự như với Google, cho phép người dùng ủy quyền xác thực cho Github, và cũng cung cấp các API và cài đặt cho các ứng dụng bên thứ ba có thể đăng ký và sử dụng.

Hai nền tảng nói trên đều cung cấp các giải pháp xác thực đa bước của riêng mình, ví dụ như Google thì có mật khẩu một lần gửi qua tin nhắn, hoặc prompt từ thiết bị di động, còn Github thì có ứng dụng trên thiết bị di động hoặc mật khẩu một lần thông qua tin nhắn. Tuy nhiên, không phải người dùng nào cũng bật các tính năng xác thực đa bước của các nền tảng này. Vì thế, các ứng dụng bên thứ ba sử dụng ủy thác xác thực dùng các nền tảng này sẽ gặp phải vấn đề bảo mật nếu như tài khoản Google hoặc Github của người dùng bị tấn công.

Giải quyết được vấn đề này, các ứng dụng bên thứ ba sử dụng 2 nền tảng nói trên, bên cạnh phụ thuộc vào đa xác thực mà chính nền tảng cung cấp, module cũng cần phải tự mình cài đặt giải pháp xác thực đa bước của riêng mình. Ngoài ra, để

cung cấp sự linh hoạt, tính năng tùy chỉnh như bật tắt cũng là cần thiết.

2.2 Tổng quan chức năng

Dựa vào những khảo sát hiện trạng cũng như các nền tảng có sẵn ở trên, ta xác định được chức năng tổng quan của đề tài như sau:

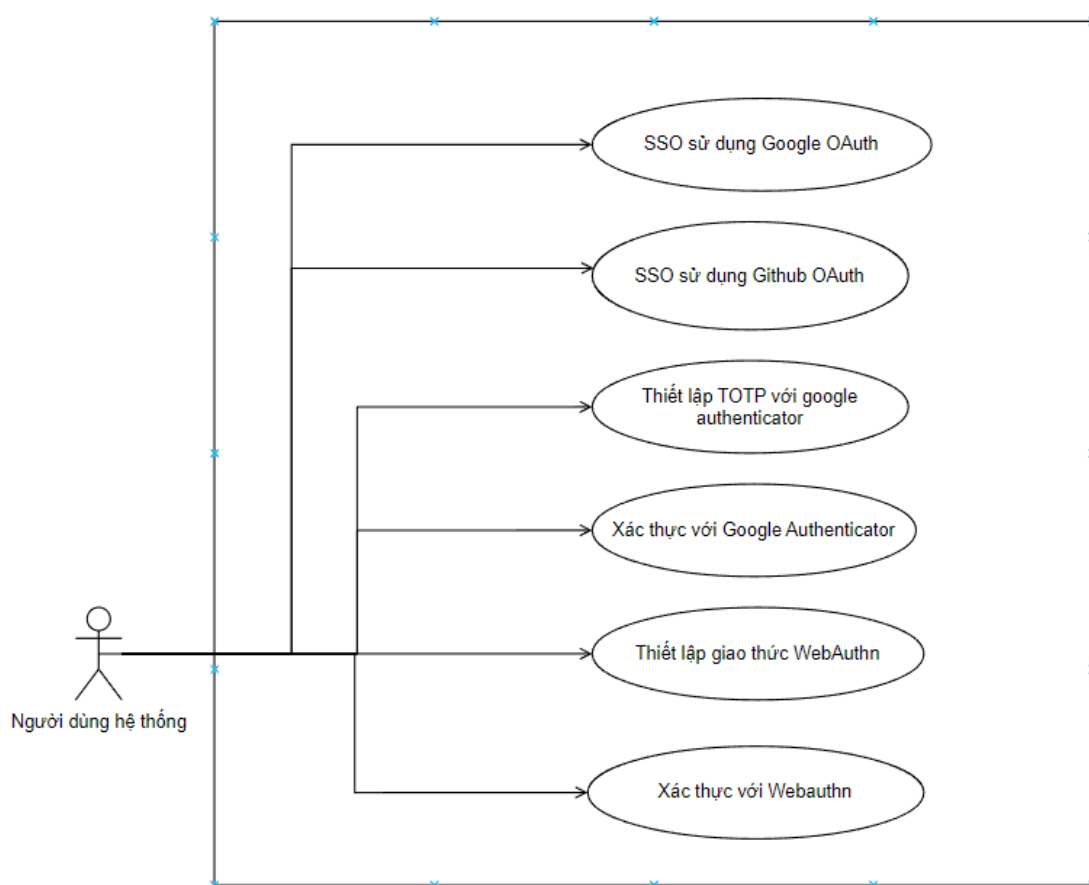
- Xác thực với SSO: cung cấp tính tiện lợi và giảm thiểu khả năng bị tấn công bảo mật do các vấn đề liên quan đến mật khẩu như mật khẩu yếu, quên mật khẩu,...
- Xác thực nhiều bước MFA: tăng cường tính an toàn và bảo mật thông tin cho hệ thống.
- Khả năng cài đặt mức độ bảo mật mong muốn sử dụng API mà không phải khởi động lại hệ thống.

2.2.1 Tác nhân hệ thống

Với những chức năng ở trên, tôi xác định được các actor của hệ thống bao gồm:

2.3 Phân tích chi tiết các ca sử dụng

2.3.1 Đối với tác nhân Người sử dụng hệ thống



Hình 2.1: Biểu đồ ca sử dụng với tác nhân Người dùng hệ thống

Tác nhân	Vai trò	Cả sử dụng chính
Người dùng	Sử dụng module để xác thực danh tính	<ul style="list-style-type: none"> - SSO sử dụng Google OAuth: người dùng đăng nhập một lần thông qua tài khoản Google. - SSO sử dụng Github OAuth: người dùng đăng nhập một lần thông qua tài khoản Github. - Thiết lập TOTP với Google Authenticator: người dùng thiết lập entry trong ứng dụng Google Authenticator với mã QR. - Xác thực với Google Authenticator: người dùng sử dụng mã otp từ ứng dụng Google Authenticator để xác thực đa yếu tố. - Thiết lập giao thức WebAuthn: người dùng sử dụng thiết bị di động của mình để đăng ký thiết bị xác thực với cơ chế Passkey. - Xác thực với WebAuthn: người dùng sử dụng thiết bị xác thực (thiết bị di động) để xác thực đa yếu tố với Passkey.
Quản trị viên	Tùy chỉnh thiết lập của các chức năng xác thực ở mức độ toàn cục	<ul style="list-style-type: none"> - Quản lý cài đặt xác thực: quản trị viên có thể xem các cài đặt xác thực của hệ thống hiện tại và cập nhật các cài đặt đó.

Bảng 2.1: Các tác nhân hệ thống

a, Đặc tả chức năng SSO sử dụng Google OAuth

Dưới đây là bảng đặc tả ca sử dụng "SSO sử dụng Google OAuth".

Mã ca sử dụng	UC-01
Tên ca sử dụng	SSO sử dụng Google OAuth
Mô tả	Người dùng đăng nhập một lần vào hệ thống sử dụng tài khoản Google
Tác nhân	Người dùng hệ thống
Sự kiện kích hoạt	Người dùng muốn đăng nhập vào hệ thống sử dụng tài khoản Google của mình
Tiền điều kiện	- Người dùng phải có tài khoản Google trước đó
Hậu điều kiện	- Người dùng đăng nhập vào hệ thống thành công - Hệ thống lưu lại thông tin của người dùng nếu người dùng chưa từng đăng nhập vào hệ thống với tài khoản Google trước đó
Luồng sự kiện chính	1. Người dùng truy cập trang đăng nhập, click chọn đăng nhập với Google 2. Hệ thống chuyển hướng người dùng sang trang đăng nhập của Google 3. Người dùng nhập thông tin tài khoản và chọn đăng nhập 4. Google xác thực thông tin đăng nhập thành công và cho phép người dùng truy cập dịch vụ 5. Hệ thống lưu lại thông tin của người dùng và tài khoản Google tương ứng
Luồng sự kiện thay thế	4a. Google xác thực thông tin của người dùng thất bại và thông báo lỗi 5a. Hệ thống xác nhận người dùng đã đăng nhập trước đó, không tiến hành lưu lại tài khoản người dùng nữa

Bảng 2.2: Bảng đặc tả ca sử dụng SSO sử dụng Google OAuth

b, Đặc tả chức năng SSO sử dụng Github OAuth

Dưới đây là bảng đặc tả ca sử dụng "SSO sử dụng Github OAuth".

Mã ca sử dụng	UC-02
Tên ca sử dụng	SSO với Github OAuth
Mô tả	Người dùng đăng nhập một lần vào hệ thống sử dụng tài khoản Github
Tác nhân	Người dùng hệ thống
Sự kiện kích hoạt	Người dùng muốn đăng nhập vào hệ thống sử dụng tài khoản Github của mình
Tiền điều kiện	- Người dùng phải có tài khoản Github trước đó
Hậu điều kiện	- Người dùng đăng nhập vào hệ thống thành công - Hệ thống lưu lại thông tin của người dùng nếu người dùng chưa từng đăng nhập vào hệ thống với tài khoản Github trước đó
Luồng sự kiện chính	1. Người dùng truy cập trang đăng nhập, click chọn đăng nhập với Github 2. Hệ thống chuyển hướng người dùng sang trang đăng nhập của Github 3. Người dùng nhập thông tin tài khoản và chọn đăng nhập 4. Github xác thực thông tin đăng nhập thành công và cho phép người dùng truy cập dịch vụ 5. Hệ thống lưu lại thông tin của người dùng và tài khoản Github tương ứng
Luồng sự kiện thay thế	4a. Github xác thực thông tin của người dùng thất bại và thông báo lỗi 5a. Hệ thống xác nhận người dùng đã đăng nhập trước đó, không tiến hành lưu lại tài khoản người dùng nữa

Bảng 2.3: Bảng đặc tả ca sử dụng SSO sử dụng Github OAuth

c, Đặc tả chức năng thiết lập TOTP với Google Authenticator

Dưới đây là bảng đặc tả ca sử dụng "Thiết lập TOTP với Google Authenticator".

Mã ca sử dụng	UC-03
Tên ca sử dụng	Thiết lập TOTP với Google Authenticator
Mô tả	Người dùng thực hiện thiết lập các thông tin cần thiết với ứng dụng Google Authenticator để phục vụ xác thực đa yếu tố
Tác nhân	Người dùng hệ thống.
Sự kiện kích hoạt	Người dùng truy cập tài nguyên được bảo vệ với MFA
Tiền điều kiện	<ul style="list-style-type: none"> - Người dùng phải đăng nhập vào hệ thống trước đó - Tài nguyên hệ thống phải được bảo vệ bởi MFA và yêu cầu người dùng xác thực MFA với TOTP trước khi truy cập - Người dùng phải chưa đăng ký thông tin xác thực TOTP trước đó
Hậu điều kiện	<ul style="list-style-type: none"> - Người dùng thiết lập xác thực đa yếu tố với TOTP thành công - Hệ thống lưu lại thông tin xác thực TOTP của người dùng
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng truy cập vào tài nguyên được bảo vệ bởi MFA khi chưa thực hiện xác thực đa yếu tố 2. Hệ thống chuyển hướng người dùng sang trang thiết lập xác thực đa yếu tố với TOTP 3. Hệ thống tính toán giá trị bí mật (secret) của TOTP và tạo thành một mã QR gửi về cho người dùng 4. Người dùng quét mã QR sử dụng ứng dụng Google Authenticator 5. Google Authenticator xác nhận thông tin thiết lập và bắt đầu sinh mã OTP 6 số theo chu kỳ 6. Người dùng nhập mã 6 số lấy được từ Google Authenticator vào ô mã trên giao diện và chọn verify 7. Hệ thống kiểm tra tính hợp lệ của mã nhận được từ người dùng với giá trị secret được tạo trước đó 8. Hệ thống thông báo thiết lập TOTP thành công 9. Hệ thống lưu thông tin thiết lập TOTP vào cơ sở dữ liệu
Luồng sự kiện thay thế	8a. Mã không hợp lệ, hệ thống thông báo thiết lập TOTP thất bại

Bảng 2.4: Bảng đặc tả ca sử dụng thiết lập TOTP với Google Authenticator

d, Đặc tả chức năng xác thực với Google Authenticator

Dưới đây là bảng đặc tả ca sử dụng "Xác thực với Google Authenticator".

Mã ca sử dụng	UC-04
Tên ca sử dụng	Xác thực với Google Authenticator
Mô tả	Người dùng thực hiện xác thực đa yếu tố sử dụng Google Authenticator
Tác nhân	Người dùng hệ thống
Sự kiện kích hoạt	Người dùng truy cập tài nguyên được bảo vệ với MFA
Tiền điều kiện	<ul style="list-style-type: none"> - Người dùng phải đăng nhập vào hệ thống trước đó - Tài nguyên hệ thống phải được bảo vệ bởi MFA và yêu cầu người dùng xác thực MFA với TOTP trước khi truy cập - Người dùng đã đăng ký thông tin xác thực TOTP trước đó
Hậu điều kiện	- Người dùng xác thực đa yếu tố với TOTP thành công
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng truy cập vào tài nguyên được bảo vệ bởi MFA khi chưa thực hiện xác thực đa yếu tố 2. Hệ thống chuyển hướng người dùng sang trang xác thực đa yếu tố với TOTP 3. Google Authenticator sinh mã OTP 6 số theo chu kỳ 4. Người dùng nhập mã 6 số lấy được từ Google Authenticator vào ô mã trên giao diện và chọn verify 5. Hệ thống kiểm tra tính hợp lệ của mã nhận được từ người dùng với giá trị secret được tạo trước đó 6. Hệ thống thông báo thiết lập TOTP thành công
Luồng sự kiện thay thế	6a. Mã không hợp lệ, hệ thống thông báo thiết lập TOTP thất bại

Bảng 2.5: Bảng đặc tả ca sử dụng Xác thực với Google Authenticator

e, Đặc tả chức năng thiết lập giao thức WebAuthn

Dưới đây là bảng đặc tả ca sử dụng "Thiết lập giao thức WebAuthn".

Mã ca sử dụng	UC-05
Tên ca sử dụng	Thiết lập giao thức WebAuthn
Mô tả	Người dùng thiết lập giao thức WebAuthn để thực hiện xác thực đa bước
Tác nhân	Người dùng hệ thống
Sự kiện kích hoạt	Người dùng truy cập tài nguyên được bảo vệ với MFA
Tiền điều kiện	<ul style="list-style-type: none"> - Người dùng phải đăng nhập vào hệ thống trước đó - Tài nguyên hệ thống phải được bảo vệ bởi MFA và yêu cầu người dùng xác thực MFA với WebAuthn trước khi truy cập - Người dùng chưa đăng ký thiết bị xác thực WebAuthn nào trước đó.
Hậu điều kiện	<ul style="list-style-type: none"> - Người dùng thiết lập xác thực đa yếu tố thành công - Hệ thống lưu lại thông tin xác thực WebAuthn của người dùng
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng truy cập vào tài nguyên được bảo vệ bởi MFA khi chưa thiết lập xác thực đa yếu tố 2. Hệ thống chuyển hướng người dùng sang trang thiết lập WebAuthn 3. Người dùng chọn nút đăng ký thiết bị 4. Hệ thống tính toán ra object chứa tùy chọn cần thiết cho việc đăng ký thiết bị authenticator cho giao thức WebAuthn ứng với người dùng hiện tại 5. Trình duyệt hiển thị các lựa chọn khả thi để thiết lập thiết bị xác thực dựa trên tùy chọn mà hệ thống cung cấp 6. Người dùng lựa chọn thiết bị/phương pháp đăng ký thiết bị xác thực 7. Người dùng thực hiện đăng ký thiết bị dựa trên lựa chọn đã chọn. Thông thường đối với các thiết bị di động, yếu tố sinh trắc hoặc mật khẩu khóa thiết bị sẽ được sử dụng 8. Trình duyệt thông báo đăng ký thiết bị thành công, chuyển tiếp thông tin thiết bị cho hệ thống 9. Hệ thống lưu lại thông tin về thiết bị xác thực được đăng ký
Luồng sự kiện thay thế	<ol style="list-style-type: none"> 5a. Người dùng chọn 'Cancel' 5a1. Trình duyệt thông báo "The operation either timed out or was not allowed" 6a. Trình duyệt thông báo "The operation either timed out or was not allowed" sau khi người dùng không lựa chọn sau khoảng thời gian timeout 6b. Trình duyệt yêu cầu "Turn on Bluetooth?" khi thiết bị của người dùng không bật bluetooth 8a. Trình duyệt thông báo "The operation timed out or was not allowed" khi người dùng thất bại trong việc đăng ký thiết bị đã chọn trong khoảng thời gian timeout

Bảng 2.6: Bảng đặc tả ca sử dụng thiết lập giao thức WebAuthn

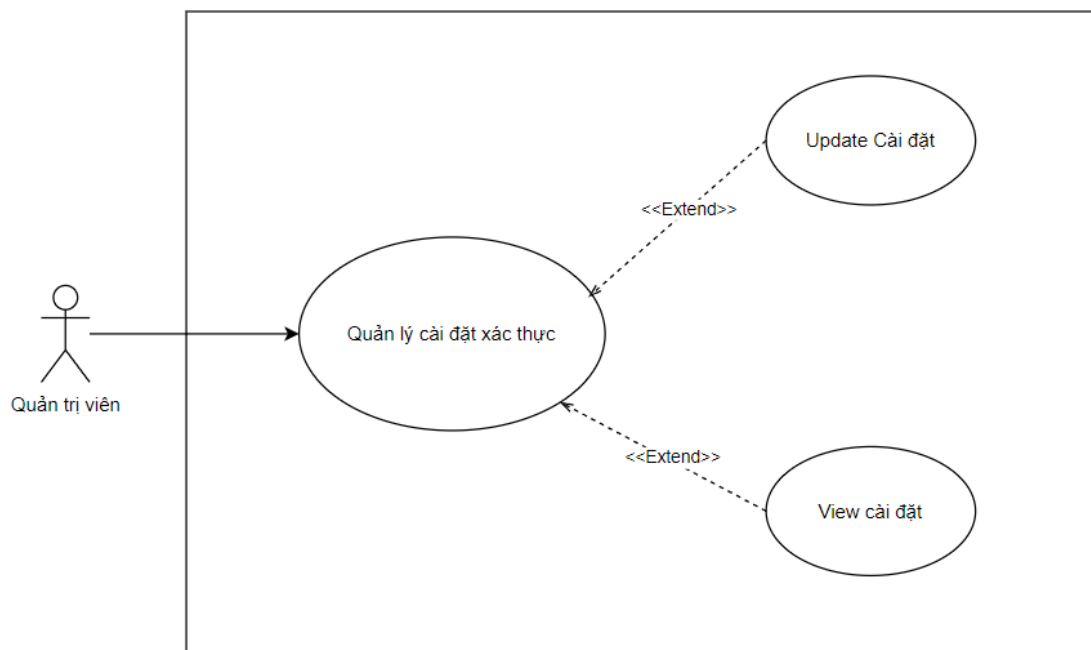
f, Đặc tả chức xác thực với WebAuthn

Dưới đây là bảng đặc tả ca sử dụng "Xác thực với WebAuthn".

Mã ca sử dụng	UC-06
Tên ca sử dụng	Xác thực với WebAuthn
Mô tả	Người dùng thực hiện xác thực đa bước sử dụng WebAuthn
Tác nhân	Người dùng hệ thống
Sự kiện kích hoạt	Người dùng truy cập tài nguyên được bảo vệ với MFA
Tiền điều kiện	<ul style="list-style-type: none"> - Người dùng phải đăng nhập vào hệ thống trước đó - Tài nguyên hệ thống phải được bảo vệ bởi MFA và yêu cầu người dùng xác thực MFA với WebAuthn trước khi truy cập - Người dùng đã đăng ký thành công thiết bị xác thực WebAuthn trước đó
Hậu điều kiện	<ul style="list-style-type: none"> - Người dùng xác thực đa yếu tố thành công
Luồng sự kiện chính	<ol style="list-style-type: none"> 1. Người dùng truy cập vào tài nguyên được bảo vệ bởi MFA khi chưa xác thực MFA 2. Hệ thống chuyển hướng người dùng sang trang xác thực MFA 3. Người dùng chọn nút xác thực 4. Truy vấn từ cơ sở dữ liệu ra thông tin các thiết bị đã đăng ký, từ đó sinh ra tùy chọn xác thực cho người dùng 5. Trình duyệt hiển thị các lựa chọn khả thi để xác thực dựa trên tùy chọn mà hệ thống cung cấp 6. Người dùng lựa chọn thiết bị để xác thực 7. Người dùng thực hiện xác thực trên thiết bị đã chọn 8. Trình duyệt nhận thông tin xác thực từ thiết bị và chuyển tiếp thông tin đến hệ thống 9. Hệ thống kiểm tra tính hợp lệ của thông tin xác thực nhận được từ trình duyệt 10. Hệ thống thông báo xác thực MFA thành công.
Luồng sự kiện thay thế	<ol style="list-style-type: none"> 5a. Người dùng chọn 'Cancel' 5a1. Trình duyệt thông báo "The operation either timed out or was not allowed" 6a. Trình duyệt thông báo "The operation either timed out or was not allowed" sau khi người dùng không lựa chọn sau khoảng thời gian time-out 6b. Trình duyệt yêu cầu "Turn on Bluetooth?" khi thiết bị của người dùng không bật bluetooth 8a. Trình duyệt thông báo "The operation timed out or was not allowed" khi người dùng thất bại trong việc xác thực với thiết bị đã chọn trong khoảng thời gian timeout

Bảng 2.7: Bảng đặc tả ca sử dụng xác thực với giao thức WebAuthn

2.3.2 Đối với tác nhân Quản trị viên



Hình 2.2: Biểu đồ ca sử dụng đối với tác nhân Quản trị viên

a, Đặc tả chức năng Update cài đặt xác thực

Mã ca sử dụng	UC-07
Tên ca sử dụng	Update cài đặt xác thực
Mô tả	Quản trị viên cập nhật cài đặt của các tính năng xác thực
Tác nhân	Quản trị viên
Sự kiện kích hoạt	Quản trị viên có nhu cầu cập nhật cài đặt cho hệ thống
Tiền điều kiện	- Không
Hậu điều kiện	- Quản trị viên cập nhật cài đặt xác thực thành công
Luồng sự kiện chính	1. Quản trị viên truy cập vào trang cài đặt 2. Hệ thống truy vấn các thông tin cài đặt hiện tại và hiển thị form cập nhật cài đặt 3. Quản trị viên cập nhật thông tin các cài đặt theo nhu cầu 4. Quản trị viên chọn lưu để cập nhật dữ liệu lên hệ thống 5. Hệ thống cập nhật lại cài đặt trên file cài đặt
Luồng sự kiện thay thế	5a. Hệ thống báo lỗi, sai lược đồ của form 5b. Hệ thống báo lỗi, cập nhật thất bại

Bảng 2.8: Bảng đặc tả ca sử dụng update cài đặt xác thực

CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

3.1 Giới thiệu về các công nghệ sử dụng cho giải pháp đăng nhập một lần

3.1.1 Tổng quan về giao thức OAuth 2.0

OAuth 2.0 là phiên bản mới nhất của giao thức Open Authorization, thay thế cho OAuth 1.0 từ năm 2012. OAuth 2.0 là tiêu chuẩn cho phép website hoặc ứng dụng được phép thay mặt người dùng truy cập vào tài nguyên được lưu trữ ở ứng dụng khác, cho phép ứng dụng chia sẻ tài nguyên mà không cần xác thực thông tin đăng nhập [4]. Điều này giúp tiết kiệm thời gian và tránh việc nhớ nhiều tài khoản đăng nhập khác nhau.

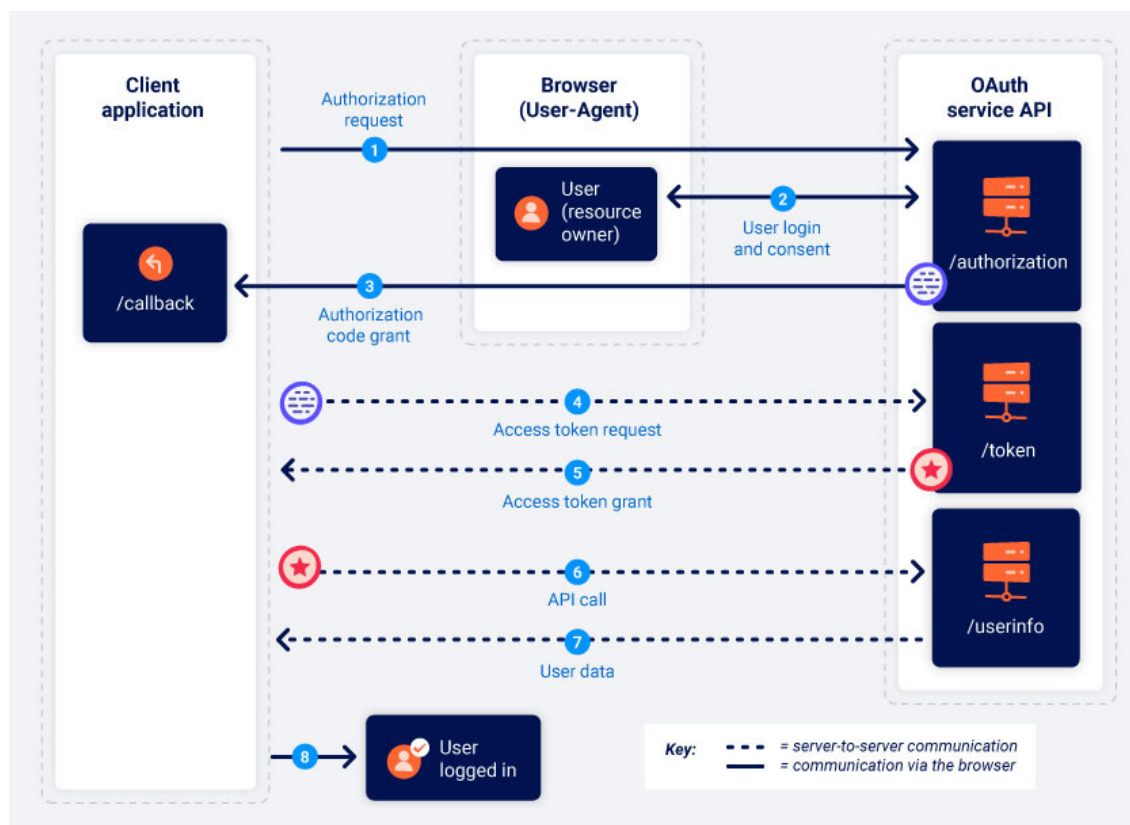
OAuth 2.0 là một giao thức ủy quyền chứ không phải giao thức xác thực, vì vậy có thể xem nó như một phương tiện dùng để cấp quyền truy cập vào tài nguyên. OAuth 2.0 sử dụng mã thông báo truy cập (access token) để cấp quyền truy cập vào tài nguyên cho website và ứng dụng. Mã thông báo truy cập sẽ không được OAuth 2.0 định dạng mà tùy thuộc vào ứng dụng.

OAuth 2.0 định nghĩa 4 vai trò:

1. Chủ sở hữu tài nguyên: là người dùng hoặc hệ thống sở hữu tài nguyên có khả năng cấp quyền truy cập vào hệ thống của họ.
2. Client: là ứng dụng muốn truy cập vào tài nguyên được bảo vệ. Trong trường hợp của đề tài, đó chính là module web đang được phát triển.
3. Máy chủ ủy quyền: là đối tượng quyết định việc cấp quyền truy cập dữ liệu cho client. Máy chủ ủy quyền nhận yêu cầu từ Client sau đó xác minh danh tính người dùng và cấp mã access token để client truy cập ứng dụng.
4. Máy chủ tài nguyên: nơi lưu trữ tài nguyên. Máy chủ này chấp nhận và xác thực mã truy cập từ Client.

Cơ chế hoạt động của giao thức OAuth 2.0 như sau:

- Bước 1: Client gửi một request tới server authentication.
- Bước 2: Sau khi nhận được request khởi tạo trên, trình duyệt của chủ sở hữu tài nguyên được điều hướng tới một trang đăng nhập vào hệ thống OAuth. Sau khi người dùng xác thực thành công tại đây, người dùng sẽ được hỏi có cung cấp các quyền cụ thể cho client hay không.
- Bước 3: Sau khi có được sự đồng ý của người dùng, máy chủ ủy quyền gửi lại cho client một đoạn mã ủy quyền (Authorization code).



Hình 3.1: Hoạt động của OAuth 2.0

- Bước 4: Client yêu cầu access token từ máy chủ ủy quyền sử dụng mã ủy quyền này. Quá trình này xảy ra trên một kênh riêng, không đi qua browser để tăng tính bảo mật.
- Bước 5: Client nhận access token.
- Bước 6: Client sử dụng access token, gửi yêu cầu tới tài nguyên cần thiết để lấy được tài nguyên.
- Bước 7: Lúc này máy chủ tài nguyên dựa vào access token của client mà trả về các dữ liệu tương ứng.

3.1.2 Tổng quan về Google OAuth 2.0

Google cung cấp dịch vụ OAuth 2.0 cho phép các ứng dụng truy cập vào các tài nguyên của Google, chẳng hạn như Gmail, Drive, và Calendar thông qua ủy quyền và xác thực. Dịch vụ OAuth 2.0 của Google là một cách an toàn và hiệu quả để chia sẻ dữ liệu và được khuyến khích sử dụng cho bất kỳ ứng dụng hoặc dịch vụ nào cần truy cập vào tài nguyên của Google. Dịch vụ này sử dụng mã hóa để bảo vệ dữ liệu của người dùng và ngăn chặn việc truy cập trái phép. Dịch vụ OAuth 2.0 của Google cũng rất dễ sử dụng và có thể được tích hợp với các ứng dụng và dịch vụ khác nhau.

Để sử dụng dịch vụ OAuth 2.0 của Google, các ứng dụng cần tạo một tài khoản Google Cloud Platform và cấp quyền cho ứng dụng truy cập vào các tài nguyên mà nó cần. Sau khi đã cấp quyền, ứng dụng có thể sử dụng mã truy cập để truy cập vào các tài nguyên.

Một số lợi ích của Google OAuth 2.0 bao gồm

- An toàn: Dịch vụ OAuth 2.0 của Google sử dụng mã hóa để bảo vệ dữ liệu của người dùng và ngăn chặn việc truy cập trái phép.
- Hiệu quả: Dịch vụ OAuth 2.0 của Google rất dễ sử dụng và có thể được tích hợp với các ứng dụng và dịch vụ khác nhau.
- Linh hoạt: Dịch vụ OAuth 2.0 của Google có thể được sử dụng để truy cập vào nhiều loại tài nguyên khác nhau, chẳng hạn như Gmail, Drive, và Calendar.

3.1.3 Tổng quan về Github OAuth 2.0

Github cung cấp dịch vụ OAuth cho phép các ứng dụng truy cập vào các tài nguyên của Github, chẳng hạn như kho lưu trữ, tổ chức và người dùng. Dịch vụ OAuth của Github là một cách an toàn và hiệu quả để chia sẻ dữ liệu và được khuyến khích sử dụng cho bất kỳ ứng dụng hoặc dịch vụ nào cần truy cập vào tài nguyên của Github.

Để sử dụng dịch vụ OAuth của Github, các ứng dụng cần tạo một ứng dụng OAuth trên Github và cấp quyền cho ứng dụng truy cập vào các tài nguyên mà nó cần. Sau khi đã cấp quyền, ứng dụng có thể sử dụng mã truy cập để truy cập vào các tài nguyên.

Dịch vụ OAuth của Github là một cách an toàn và hiệu quả để chia sẻ dữ liệu. Dịch vụ này sử dụng mã hóa để bảo vệ dữ liệu của người dùng và ngăn chặn việc truy cập trái phép. Dịch vụ OAuth của Github cũng rất dễ sử dụng và có thể được tích hợp với các ứng dụng và dịch vụ khác nhau.

Dịch vụ OAuth của Github cũng cung cấp các lợi ích tương tự như với Google OAuth 2.0.

3.2 Giới thiệu về các công nghệ sử dụng trong giải pháp xác thực đa bước.

3.2.1 Giới thiệu về mật khẩu một lần dựa trên thời gian

a, Tổng quan

Mật khẩu dùng một lần dựa trên thời gian (TOTP) là một thuật toán máy tính tạo mật khẩu dùng một lần (OTP) sử dụng thời gian hiện tại làm nguồn xác định duy nhất. TOTP là nền tảng của tiêu chuẩn xác thực mở OAuth và được sử dụng trong các hệ thống xác thực đa yếu tố.

b, Thuật toán

Trên thực tế, thuật toán của TOTP được dựa trên HOTP (Hash-based one time password), tức là tạo đối xứng các mật khẩu hoặc giá trị mà người dùng có thể đọc được thành 1 dãy liên tục 6-8 chữ số, mỗi một lần xác thực thì sẽ chỉ dùng duy nhất một mật khẩu. Mỗi lần tạo mật khẩu thì sẽ sử dụng duy nhất một giá trị bộ đếm, tuy nhiên, khác với HOTP, TOTP thay thế bộ đếm bằng một giá trị thời gian thực.

Giá trị thời gian thực được tính như sau:

$$T = \frac{T_{current_unix_time} - T_0}{X}$$

Trong đó:

- $T_{current_unix_time}$ là giá trị thời gian hiện tại và được tính theo thời gian unix, nghĩa là tính từ thời điểm của Unix epoch là ngày 01/01/1970 theo UTC - giờ chuẩn quốc tế.
- T_0 là giá trị thời gian ban đầu (thông thường chọn $T_0 = 0$).
- X là bước thời gian, tham số này được coi là yếu tố quyết định thời gian hợp lệ của mật khẩu OTP
- T chính là kết quả tính (đã lấy phần nguyên) và là giá trị cần tìm.

Tiếp đó, TOTP thay thế T với giá trị C trong thuật toán HOTP để tính toán mật khẩu OTP như sau

$$TOTP = HOTP(K, T)$$

Độ dài mật khẩu của thuật toán TOTP cũng tương tự như HOTP và được tính như sau:

$$TOTP_{length} = TOTP(K, T) \bmod 10^d$$

3.2.2 Google Authenticator

Google Authenticator là ứng dụng cho phép tạo thêm một lớp bảo mật cho các tài khoản trực tuyến của bạn bằng cách thêm bước xác minh thứ hai khi bạn đăng nhập. Điều này có nghĩa là, ngoài mật khẩu, bạn cũng sẽ cần nhập mã do ứng dụng Google Authenticator tạo trên điện thoại.

Các tính năng của Google Authenticator gồm có:

- Đồng bộ hóa mã Authenticator với Tài khoản Google, giúp cho người dùng có thể truy cập vào các mã kể cả khi thiết bị di động bị mất.
- Tự động thiết lập tài khoản Authenticator bằng mã QR.

- Hỗ trợ nhiều tài khoản.
- Hỗ trợ cả HOTP lẫn TOTP.
- Chuyển tài khoản giữa các thiết bị bằng mã QR.

3.2.3 Tổng quan về giao thức WebAuthn

WebAuthn, hay Web Authentication API là một tiêu chuẩn kỹ thuật được viết với W3C và FIDO, với sự tham gia của Google, Mozilla, Microsoft, Yubico,... API này cho phép máy chủ đăng ký và xác thực người dùng sử dụng mật mã khóa công khai thay vì sử dụng mật khẩu [5]. Nó là một trong số những giải pháp mới hơn và thiết thực hơn cho bài toán mật khẩu đang ngày càng trở nên hóc búa.

Nó cho phép máy chủ tích hợp với các authenticators mạnh mẽ có sẵn trong các thiết bị như Window Hello hay Apple Touch ID. Thay vì phải lưu trữ mật khẩu, một cặp mã private-public (còn được gọi là credential) được tạo ra cho một website. Mã private sẽ được lưu an toàn tại thiết bị authenticator, còn mã public và credential ID sẽ được gửi đến máy chủ để lưu trữ. Máy chủ sau này có thể sử dụng mã public đó để xác thực và định danh người dùng.

Phương pháp này giảm thiểu vấn đề về ghi nhớ mật khẩu vì người dùng không cần phải nghĩ ra hoặc ghi nhớ mật khẩu phức tạp; đồng thời nó cũng giải quyết bài toán đánh cắp mật khẩu, do dữ liệu được lưu ở máy chủ là một mã public, không hề bí mật và vô dụng nếu không có mã private đi kèm với nó.

Trong đề tài, WebAuthn được ứng dụng để xây dựng tính năng xác thực đa bước từ thiết bị di động.

3.3 Giới thiệu về các công nghệ được sử dụng để xây dựng module

3.3.1 Giới thiệu về framework NestJS

a, Tổng quan

NestJS là một framework để xây dựng dịch vụ web hiệu quả, có khả năng mở rộng cao cho ngôn ngữ Typescript [6].

Các tính năng của NestJS bao gồm:

- Định tuyến: NestJS cung cấp một hệ thống định tuyến linh hoạt và có kiến trúc, cho phép người sử dụng dễ dàng ánh xạ giữa yêu cầu HTTP và các phương thức xử lý phù hợp
- Tuần tự hóa/giải tuần tự hóa: quá trình tuần tự hóa và giải tuần tự hóa được thực hiện tự động bởi NestJS, giúp ánh xạ dữ liệu từ các yêu cầu HTTP sang đối tượng Javascript và ngược lại, chuyển đổi các đối tượng javascript sang JSON để đính kèm vào thông điệp trả lời.

- Kiểm thử dữ liệu đầu vào: NestJS cung cấp một vài cách khác nhau, từ đơn giản đến phức tạp để kiểm tra dữ liệu đầu vào nhận được từ thông điệp yêu cầu HTTP và thông báo lỗi phù hợp.
- Mã trung gian: NestJS hỗ trợ tính năng mã trung gian để thực hiện các xử lý trung gian trước khi yêu cầu đến được xử lý bởi controller. Mã trung gian giúp trong việc xác thực, xử lý lỗi, ghi log, và nhiều tác vụ khác. Mã trung gian trong NestJS cũng được phân loại và đặt tên theo chức năng: guard, interceptor, pipe, middleware và từng loại có vị trí riêng trong lifecycle của một HTTP request.
- Dependency injection: NestJS hỗ trợ nguyên tắc Dependency Injection để quản lý dependencies giữa các thành phần. Điều này giúp giảm sự ràng buộc và tăng tính linh hoạt trong việc quản lý và thay đổi dependencies
- Kiến trúc modules: NestJS sử dụng khái niệm module để tạo thành các phần tử độc lập và tái sử dụng trong ứng dụng. Các module giúp tổ chức code, quản lý dependencies và tạo các cạnh tranh giữa các thành phần khác nhau.

NestJS, ở cốt lõi vẫn sử dụng các HTTP Server framework khác như Express (mặc định) và Fastify, và tập trung vào giải quyết các vấn đề liên quan "kiến trúc" (architecture) của các framework phổ biến, bằng cách cung cấp một kiến trúc sẵn sàng sử dụng giúp thúc đẩy nhanh quá trình và trải nghiệm phát triển cho các kỹ sư phần mềm, đồng thời vẫn đảm bảo được khả năng tái sử dụng, tính kết nối ít phụ thuộc (loose coupling) giữa các thành phần trong hệ thống, và khả năng tự mở rộng theo ý muốn của nhà phát triển.

b, Thành phần kiến trúc

Vì mục tiêu đề tài, phát triển Module xác thực cho framework NestJS, liên quan trực tiếp đến kiến trúc framework, kiến trúc mà NestJS cung cấp sẽ được phân tích trực tiếp tại đây. Kiến trúc của framework NestJS dựa trên kiến trúc Module, được thiết kế để tổ chức và quản lý các thành phần của ứng dụng. Các thành phần của một ứng dụng bao gồm:

- Module: Module là thành phần cơ bản trong NestJS. Nó đại diện cho một phần hoặc chức năng cụ thể của ứng dụng. Module cung cấp cách tổ chức code và đóng gói các thành phần liên quan lại với nhau. Mỗi ứng dụng NestJS bao gồm ít nhất một module gốc (root module) và có thể có nhiều module con. Một module có thể có nhiều controller, provider và các module con hoạt động độc lập với các module khác.
- Controller: Controller xử lý các yêu cầu HTTP đến từ client và chịu trách

nhệm định tuyến các yêu cầu vào các handlers tương ứng. Controller nhận yêu cầu từ client, xử lý nó và trả về câu trả lời. Các actions (hành động) trong controller xác định các endpoints và xử lý logic của ứng dụng.

- **Provider:** Providers là các thành phần cung cấp dịch vụ và dependencies cho ứng dụng. Các provider có thể được inject vào controller, services hoặc middleware bằng cách sử dụng Dependency Injection. Chúng có thể là các thư viện bên thứ ba, các đối tượng tùy chỉnh hoặc các đối tượng đã được xây dựng sẵn trong NestJS.

3.3.2 Giới thiệu về framework VueJS

Vue.js (hay còn gọi là Vue) là một framework JavaScript mã nguồn mở phát triển giao diện người dùng (UI) cho việc xây dựng các ứng dụng web đơn trang (Single-Page Applications) hiện đại. Nó được thiết kế để dễ dàng tích hợp và sử dụng trong dự án web hiện có.

Mặc dù mục tiêu đề tài tập trung vào phát triển module cho framework phía server, VueJS sẽ được sử dụng để phát triển giao diện demo cho các tính năng mà module cung cấp. Các tính năng của VueJS bao gồm:

- **Dễ học và sử dụng:** Vue.js có cú pháp rõ ràng và dễ hiểu, giúp lập trình viên nhanh chóng nắm bắt và sử dụng nó trong dự án.
- **Liên kết dữ liệu tương tác [7]:** Vue.js cung cấp tính năng liên kết dữ liệu hai chiều, cho phép dữ liệu và giao diện người dùng được cập nhật tự động khi có sự thay đổi.
- **Mô hình thành phần [7]:** VueJS khuyến khích phát triển ứng dụng theo mô hình thành phần. Việc chia nhỏ ra các thành phần giúp tăng tính tái sử dụng và tính dễ đọc hiểu của ứng dụng.
- **Hệ sinh thái đa dạng:** VueJS có một hệ sinh thái các thư viện cung cấp chức năng, giao diện giúp cho việc phát triển nhanh chóng và đơn giản.

3.3.3 Giới thiệu về MongoDB

MongoDB được sử dụng làm cơ sở dữ liệu chính lưu trữ thông tin người dùng.

MongoDb là một hệ quản trị cơ sở dữ liệu không quan hệ mã nguồn mở và phát triển bởi MongoDB Inc [8]. Nó cung cấp một cách tiếp cận linh hoạt và mở rộng để lưu trữ và truy xuất dữ liệu không cấu trúc và bán cấu trúc. Nó sở hữu các tính chất hỗ trợ sự linh hoạt và tùy biến cao trong quá trình phát triển như cấu trúc linh hoạt và tính mở rộng, bên cạnh đó vẫn cung cấp đầy đủ các tính chất quan trọng của một hệ quản trị cơ sở dữ liệu như tính nhất quán và bền vững, khả năng truy

vấn linh hoạt và hỗ trợ đa nền tảng,...

3.3.4 Giới thiệu về Redis

Redis là một hệ cơ sở dữ liệu theo kiểu ánh xạ khóa-giá trị được lưu trữ trong bộ nhớ (in memory) [9]. Redis được phát triển để cung cấp một cơ sở dữ liệu nhanh, đơn giản và linh hoạt, với khả năng lưu trữ dữ liệu trong bộ nhớ và hỗ trợ nhiều kiểu dữ liệu khác nhau.

Redis hỗ trợ các cấu trúc dữ liệu đa dạng như strings, hashes, lists, sets, sorted sets và bitmaps. Điều này cho phép lưu trữ và truy xuất dữ liệu một cách hiệu quả trong các ứng dụng có yêu cầu cao về tốc độ và hiệu suất.

Redis được sử dụng trong đề tài như là một cơ sở dữ liệu để lưu những giá trị thách thức tạm thời được sinh ra từ giao thức WebAuthn và lưu trữ các tùy chọn của module xác thực nhằm tăng tốc độ đọc, ghi.

3.3.5 Tổng quan về Restful API

a, Định nghĩa

Restful API là một kiểu kiến trúc và phong cách thiết kế API dựa trên các nguyên tắc của REST[10]. Nó được sử dụng rộng rãi trong phát triển các dịch vụ web để cung cấp giao diện cho các client giao tiếp và tương tác với các tài nguyên trên.

b, Thành phần

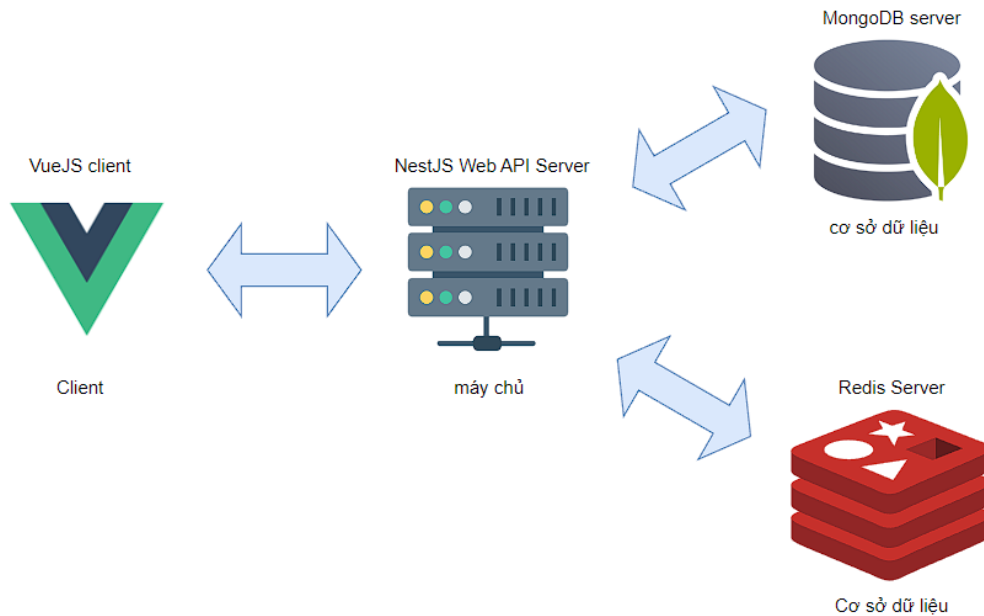
1. API: API là giao diện lập trình ứng dụng, xác định các quy tắc cần phải tuân theo để có thể giao tiếp với một hệ thống khác. Web API được coi là một cổng giúp client có thể tương tác với tài nguyên.
2. REST: hay còn gọi là Chuyển trạng thái đại diện, là một kiến trúc phần mềm quy định phong cách thiết kế và xây dựng các dịch vụ Web. Rest đảm bảo tính nhất quán, khả năng mở rộng và tương tác thông qua giao thức HTTP. REST quy định mỗi tài nguyên được định danh thông qua một URI duy nhất. Client có thể truy cập đến tài nguyên này thông qua cấu trúc đường dẫn và thực hiện hoạt động trên các tài nguyên đó sử dụng các phương thức HTTP tương ứng như GET, PUT, POST, DELETE.

RESTful API còn quy định truyền tải dữ liệu qua định dạng chuẩn như JSON hoặc XML để truyền tải dữ liệu giữa client và server.

CHƯƠNG 4. THIẾT KẾ CHỨC NĂNG HỆ THỐNG

4.1 Thiết kế kiến trúc

4.1.1 Lựa chọn kiến trúc phần mềm



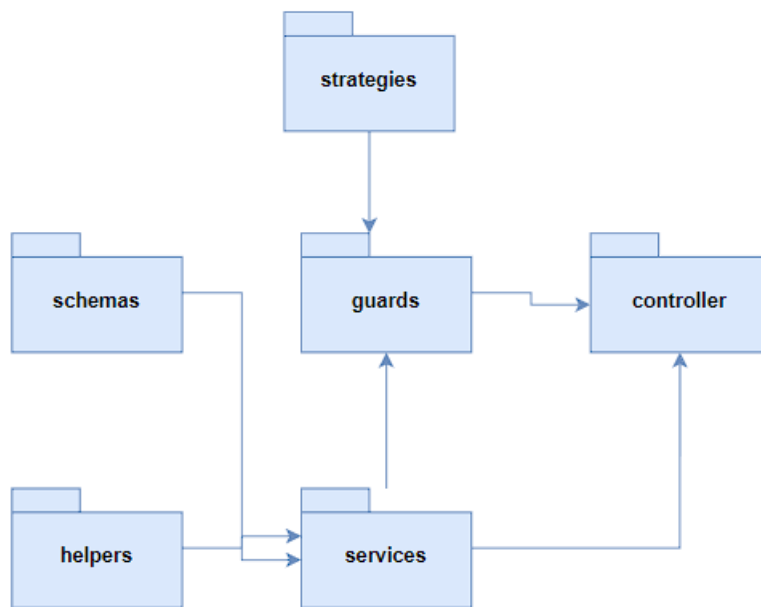
Hình 4.1: Thiết kế kiến trúc tổng quan

Trong đồ án này, tôi lựa chọn kiến trúc client-server cho sản phẩm của mình bởi khả năng mở rộng và sự tách biệt trong việc phát triển các thành phần.

Các thành phần kiến trúc:

- Client: thành phần đảm nhiệm vai trò hiển thị giao diện và xử lý các chức năng liên quan đến trải nghiệm người dùng cũng như các logic tương tác với máy chủ.
- Máy chủ: Tính toán, xử lý logic, đảm nhiệm vai trò giao tiếp với cơ sở dữ liệu và các máy chủ cung cấp dịch vụ OAuth khác.
- Cơ sở dữ liệu: đảm nhiệm vai trò lưu trữ dữ liệu của hệ thống.

4.1.2 Thiết kế tổng quan

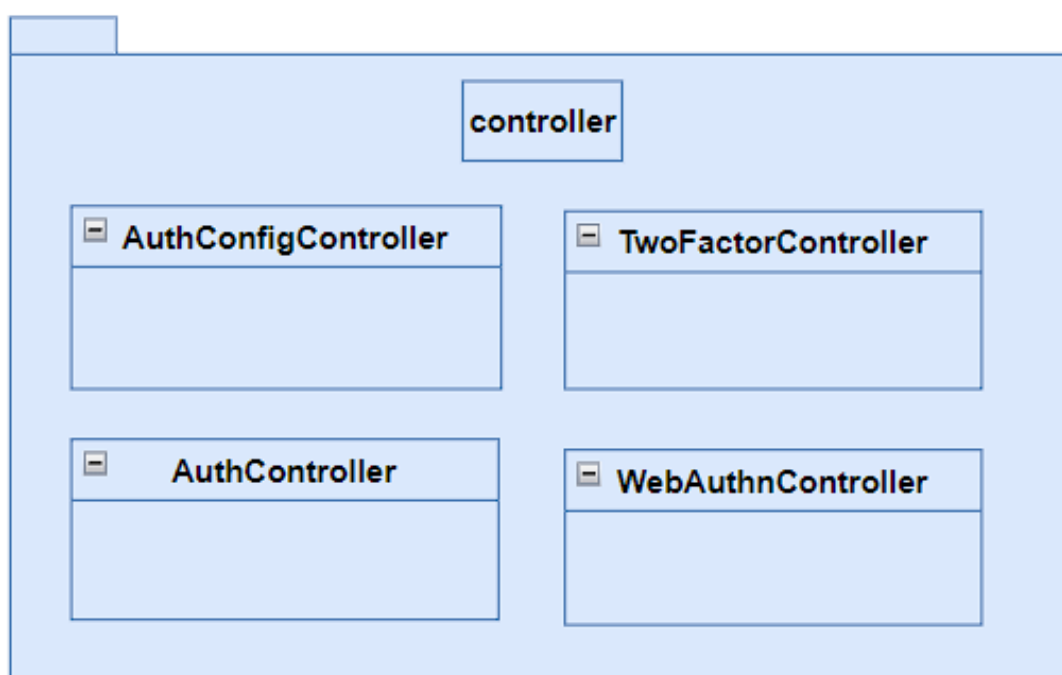


Hình 4.2: Biểu đồ gói tổng quan

Hệ thống được phát triển dựa trên 6 gói chính:

- Gói controllers: Chứa định nghĩa các đoạn mã nhận yêu cầu gửi qua API và trả về kết quả.
- Gói services: Gói khai báo các lớp chứa các logic nghiệp vụ chính.
- Gói schemas: Gói chứa định nghĩa mẫu của các đối tượng trong cơ sở dữ liệu, và cũng là gói cung cấp các phương thức giao tiếp trực tiếp với cơ sở dữ liệu.
- Gói helpers: gói chứa định nghĩa các lớp tiện ích được sử dụng xuyên suốt dự án
- Gói guards: gói chứa định nghĩa các middleware chứa logic kiểm tra các yêu cầu trước khi đến được controllers
- Gói strategies: gói chứa định nghĩa các lớp strategies cho các phương thức xác thực nâng cao, hoạt động dựa trên stratgies pattern.

4.1.3 Thiết kế chi tiết các gói



Hình 4.3: Thiết kế gói controllers

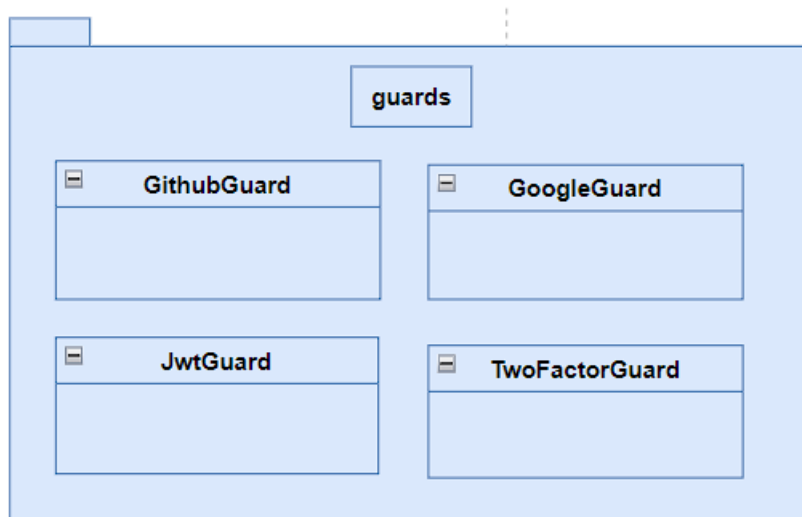
Hình 4.3 là thiết kế chi tiết của gói controller. Trong gói này, ta có các lớp định nghĩa ra các phương thức xử lý được map với các API.

Lớp `AuthConfigController` là lớp chứa phương thức định nghĩa để xử lý các yêu cầu API gửi đến `/auth-config/*`, được sử dụng để thao tác với tài nguyên cài đặt xác thực của hệ thống.

Lớp `TwoFactorController` là lớp chứa phương thức định nghĩa xử lý các yêu cầu API gửi đến `/2fa/*`, phục vụ các tính năng xác thực đa yếu tố.

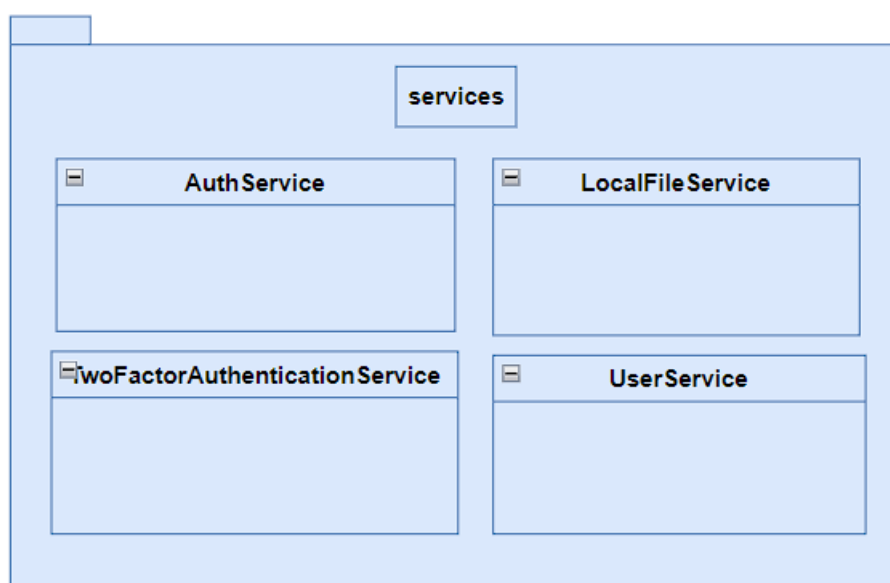
Lớp `AuthController` là lớp định nghĩa các phương thức xử lý yêu cầu API gửi đến `/auth/*`, phục vụ nhiệm vụ xác thực một lần thông qua các dịch vụ có sẵn.

Lớp `WebAuthnController` là lớp định nghĩa các phương thức xử lý yêu cầu API gửi đến `/webauthn/*`, phục vụ các tính năng xác thực với giao thức WebAuthn.



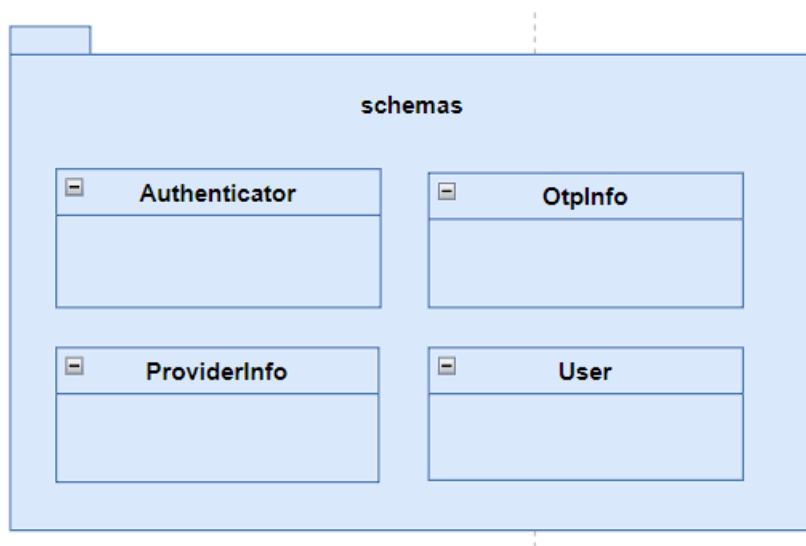
Hình 4.4: Thiết kế gói guards

Hình 4.4 là thiết kế chi tiết gói guards, bao gồm các thành phần sau: GithubGuard được sử dụng để xác thực một lần với Github OAuth. GoogleGuard được sử dụng để xác thực một lần với Google OAuth. JWTGuard kiểm tra tính hợp lệ của JWT cookie trong các API request. TwoFactorGuard kiểm tra người dùng đã được xác thực đa yếu tố hay chưa.



Hình 4.5: Thiết kế gói services

Hình 4.5 là thiết kế chi tiết gói services, bao gồm các lớp sau: AuthService cung cấp các phương thức dịch vụ cho tính năng xác thực. LocalFileService cung cấp phương thức các dịch vụ đọc, ghi file. TwoFactorService cung cấp các phương thức dịch vụ cho tính năng xác thực đa yếu tố. UserService cung cấp các phương thức dịch vụ cho tài nguyên User.



Hình 4.6: Thiết kế gói schemas

Hình 4.6 là thiết kế chi tiết gói schemas, bao gồm các lớp sau: Authenticator là lớp mô hình hóa đối tượng cho collection authenticators trong csdl. OtpInfo là lớp mô hình hóa đối tượng cho collection otpinfos trong csdl. ProviderInfo là lớp mô hình hóa đối tượng cho collection providerinfos trong csdl. User là lớp mô hình hóa đối tượng cho collection users trong csdl.



Hình 4.7: Thiết kế gói helpers

Hình 4.7 là thiết kế chi tiết gói helpers, bao gồm các lớp sau: ClsStoreFactory là lớp chứa phương thức factory để khởi tạo ClsStore cho cơ chế DI của NestJS. GithubStrategyFacotry là lớp chứa phương thức factory để khởi tạo GithubStrategy cho DI của NestJS. GoogleStrategyFactory là lớp chứa phương thức factory để khởi tạo GoogleStategy cho DI của NestJS.



Hình 4.8: Thiết kế gói strategies

Hình 4.8 là thiết kế chi tiết của gói strategies, là gói chứa các lớp định nghĩa các logic xác thực, cài đặt theo giao diện chung dựa trên Strategy pattern.

TwoFactorAuthStrategy là lớp chứa logic cho quá trình xác thực đa yếu tố sử dụng TOTP hoặc WebAuthn.

JwtStrategy là lớp chứa logic cho quá trình trích xuất jwt từ cookies của yêu cầu HTTP và kiểm tra tính hợp lệ của jwt, giải mã nó và chuyển tiếp thông tin giải mã vào ngữ cảnh của yêu cầu.

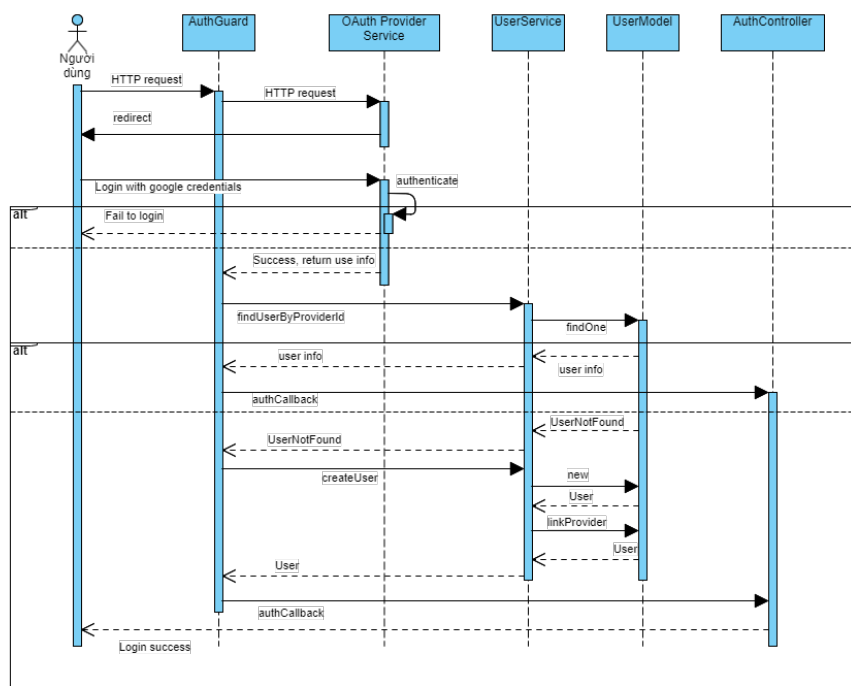
GithubStrategy là lớp chứa logic cho quá trình xác thực một lần sử dụng Github OAuth, bao gồm là các bước kiểm tra và lưu trữ thông tin người dùng và thông tin từ Github provider.

GoogleStrategy là lớp chứa logic cho quá trình xác thực một lần sử dụng Google OAuth, bao gồm các bước kiểm tra và lưu trữ thông tin người dùng và thông tin từ Google provider.

AnonymousStrategy là lớp chứa định nghĩa logic xác thực vô danh, nghĩa là cho phép mọi người dùng tiếp tục mà không kiểm tra thông tin xác thực, nó được sử dụng như một chiến lược dự phòng trong các lưu đồ xác thực.

4.2 Thiết kế chức năng

4.2.1 SSO sử dụng Google OAuth



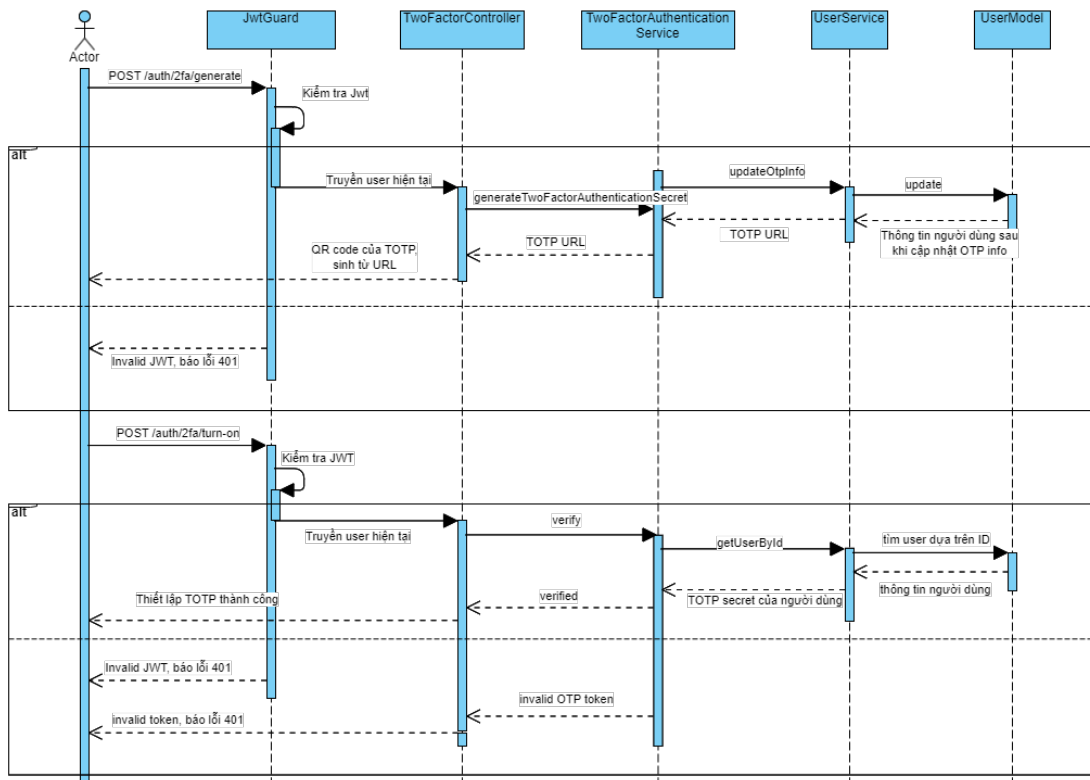
Hình 4.9: Biểu đồ tuần tự đăng nhập SSO

Đối với chức năng xác thực sử dụng SSO, người dùng sẽ bắt đầu chức năng bằng cách gọi API request (thường thông qua nút bấm trên giao diện) đến hệ thống. Request này sẽ được xử lý tại AuthGuard trước khi đến được Controller. Đoạn mã tại Controller chỉ thực thi khi người dùng xác thực thành công.

AuthGuard đóng vai trò xác thực người dùng bằng cách giao tiếp với OAuth Provider của Google hoặc Github. Thông tin đăng nhập từ google được truyền lại vào AuthGuard tại callback URL và được xử lý theo logic sau:

- Nếu người dùng chưa tồn tại trong hệ thống, hệ thống tạo một người dùng mới và gắn liền tài khoản OAuth với người dùng đó.
- Nếu người dùng đã tồn tại trong hệ thống, người dùng được đăng nhập vào hệ thống.

4.2.2 Thiết lập và xác thực với TOTP



Hình 4.10: Biểu đồ tuần tự thiết lập TOTP và xác thực TOTP

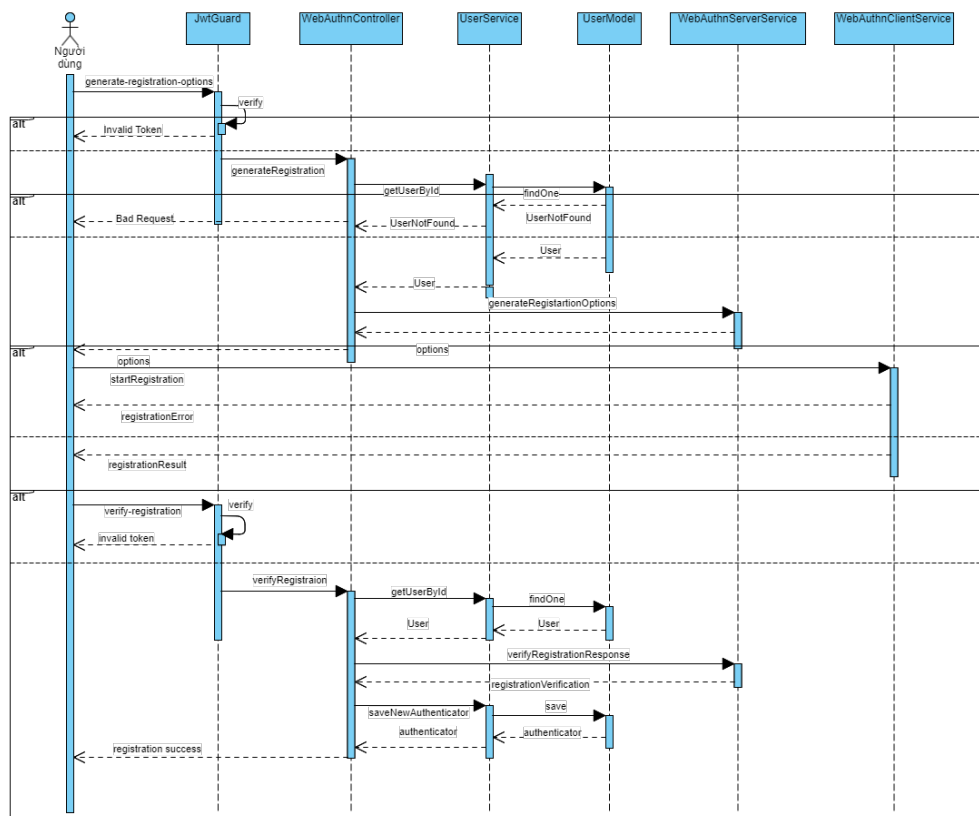
Chức năng thiết lập TOTP sử dụng Google Authenticator trải qua 2 pha: người dùng tạo ra mã QR để quét sử dụng ứng dụng Google Authenticator và người dùng nhập mã otp (mã token 6 số) để xác thực rằng QR là hợp lệ. Pha thứ 2 chính là xác thực một mã OTP hợp lệ.

Khi người dùng muốn thiết lập một mã OTP, hệ thống sẽ phải tạo ra một chuỗi bí mật ứng với TOTP của người dùng. Quá trình này được thực hiện, kèm với phần lưu mã bí mật đó vào CSDL để phục vụ xác thực sau này, bởi TwoFactorAuthenticationService. Sau khi tạo mã bí mật thành công, một URL chứa thông tin xác thực được tạo ra dưới dạng:

`otpauth://TYPE/LABEL?PARAMETERS`

URL này sẽ được sử dụng để tạo ra mã QR trả về, sử dụng cho ứng dụng Google Authenticator.

4.2.3 Thiết lập giao thức WebAuthn

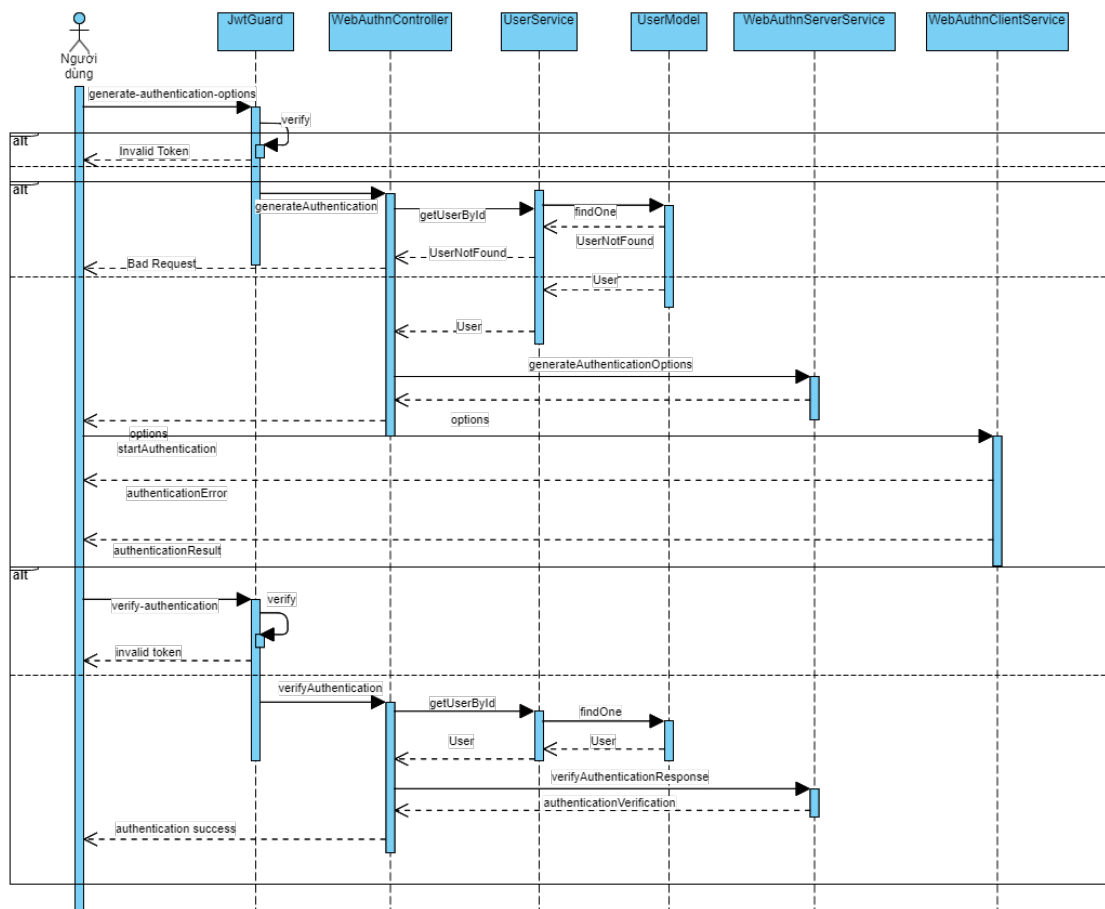


Hình 4.11: Biểu đồ tuần tự thiết lập giao thức WebAuthn

Chức năng thiết lập giao thức Webauthn yêu cầu người dùng phải đăng nhập trước khi bắt đầu. Người dùng gửi yêu cầu đăng ký đến hệ thống, hệ thống sẽ sinh ra một bộ tham số cần thiết được sử dụng cho việc đăng ký, kèm theo một giá trị thử thách được lưu trữ tạm thời trong CSDL redis, và gửi về trình duyệt của người dùng. Trình duyệt của người dùng bắt đầu quá trình đăng nhập bằng cách gọi đến API của WebAuthn có sẵn của trình duyệt. Người dùng thực hiện đăng ký thiết bị authenticator sử dụng thiết bị di động cá nhân của mình, và sau khi quá trình này kết thúc, trình duyệt sẽ nhận được một đối tượng kết quả đăng ký.

Đối tượng này sẽ được chuyển tiếp về phía máy chủ của hệ thống. Hệ thống sẽ tiến hành xác nhận các thông tin có trong đối tượng kết quả này, và kiểm tra nó với giá trị thử thách ở trong redis. Nếu không có vấn đề gì, hệ thống lưu lại giá trị thiết bị đăng ký của người dùng, và thông báo thiết lập thành công.

4.2.4 Xác thực với WebAuthn



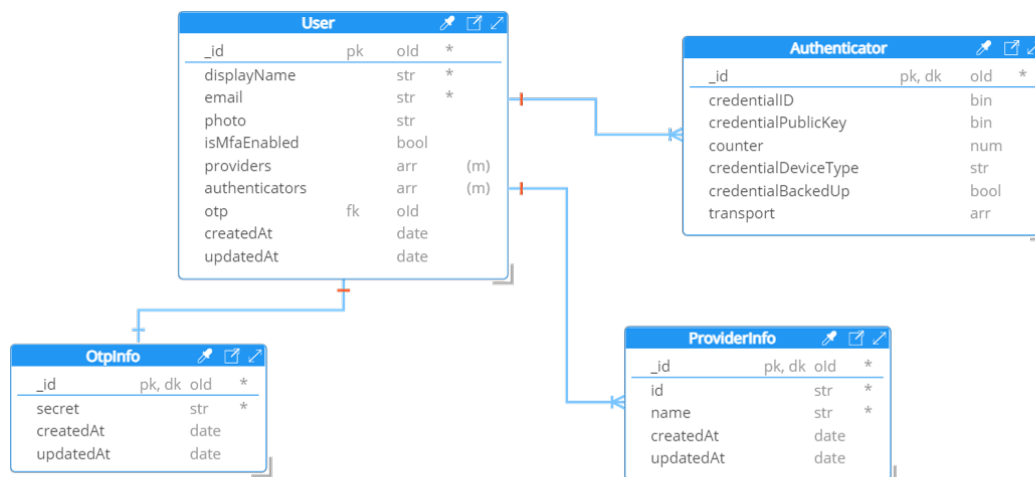
Hình 4.12: Biểu đồ tuần tự xác thực sử dụng với WebAuthn

Chức năng xác thực sử dụng giao thức WebAuthn có khá nhiều nét tương đồng với chức năng thiết lập WebAuthn. Khi người dùng yêu cầu xác thực, hệ thống sẽ tìm kiếm thông tin thiết bị ứng với người dùng, sau đó sử dụng thông tin đó để sinh ra một bộ tham số xác thực kèm theo một trị thách thức tạm thời. Thông tin này sẽ được gửi về trình duyệt của người dùng, và trình duyệt sẽ bắt đầu quá trình xác thực bằng cách sử dụng API WebAuthn của trình duyệt. Người dùng thực hiện xác thực với thiết bị di động cá nhân của mình thành công và trình duyệt sẽ nhận lại được đối tượng kết quả xác thực.

Đối tượng này sẽ được chuyển tiếp về phía máy chủ của hệ thống, và hệ thống sẽ tiến hành xác nhận thông tin và kiểm tra tính hợp lệ của nó với thử thách được tạo ra trước đó. Nếu không có vấn đề gì, hệ thống xác nhận người dùng đăng nhập thành công.

4.3 Thiết kế cơ sở dữ liệu

4.3.1 Thiết kế tổng quan



Hình 4.13: Thiết kế cơ sở dữ liệu

4.3.2 Đặc tả

STT	Tên trường	Kiểu dữ liệu	Bắt buộc	Mô tả
1	_id	string	true	Id tạo tự động bởi MongoDB
2	secret	string	true	Giá trị bí mật gắn với OTP, sử dụng để xác thực code
3	createdAt	date	false	Được khởi tạo tự động khi bản ghi được tạo
4	updatedAt	date	false	Được khởi tạo tự động khi bản ghi được tạo và cập nhật tự động khi bản ghi được cập nhật

Bảng 4.1: Đặc tả Collection OtpInfo

STT	Tên trường	Kiểu dữ liệu	Bắt buộc	Mô tả
1	_id	string	true	Id của người dùng, tạo tự động bởi MongoDB
2	displayName	string	true	Tên hiển thị của người dùng
3	email	string	true	Email của người dùng
4	photo	string	false	URL ảnh đại diện của người dùng
5	isMfaEnabled	boolean	false	Người dùng có bật MFA hay không
6	providers	array	false	Mảng các provider liên kết với tài khoản này. Dữ liệu này trong CSDL được lưu dưới dạng mảng các tham chiếu đến các bản ghi khác trong CSDL
7	authenticators	array	false	Mảng các authenticators liên kết với tài khoản này. Dữ liệu này trong CSDL được lưu dưới dạng mảng các tham chiếu đến các bản ghi khác trong CSDL.
8	otp	OtpInfo	false	Thông tin về OTP gắn liền với tài khoản. Trong CSDL đây là một tham chiếu đến bản ghi khác trong CSDL
9	createdAt	date	false	Được khởi tạo tự động khi bản ghi được tạo
10	updatedAt	date	false	Được khởi tạo tự động khi bản ghi được tạo và cập nhật tự động khi bản ghi được cập nhật

Bảng 4.2: Đặc tả Collection User

STT	Tên trường	Kiểu dữ liệu	Bắt buộc	Mô tả
1	_id	string	true	Id tạo tự động bởi MongoDB
2	name	string	true	Tên của provider
3	id	string	true	Id của user trong csdl của provider đó
4	createdAt	date	false	Được khởi tạo tự động khi bản ghi được tạo
5	updatedAt	date	false	Được khởi tạo tự động khi bản ghi được tạo và cập nhật tự động khi bản ghi được cập nhật

Bảng 4.3: Đặc tả Collection ProviderInfo

STT	Tên trường	Kiểu dữ liệu	Bắt buộc	Mô tả
1	_id	string	true	Id tạo tự động bởi MongoDB
2	credentialPublicKey	binary buffer	false	Khóa công khai của thông tin xác thực
3	credentialID	binary buffer	false	Id của thông tin xác thực đi kèm với khóa công khai trên
4	counter	number	false	giá trị counter lưu ở trên Authenticator, update mỗi khi xác thực, dùng để tránh replay attacks
5	credentialDeviceType	string	false	Thông tin xác thực đơn hay đa thiết bị. Nhận "singleDevice" hoặc "multiDevice"
6	credentialBackedUp	boolean	false	Thông tin xác thực đã được backup hay chưa
7	transports	array	false	Thông tin về các loại thiết bị.
8	createdAt	date	false	Được khởi tạo tự động khi bản ghi được tạo
9	updatedAt	date	false	Được khởi tạo tự động khi bản ghi được tạo và cập nhật tự động khi bản ghi được cập nhật

Bảng 4.4: Đặc tả Collection OtpInfo

4.4 Thiết kế giao diện

Mặc dù module tập trung vào phát triển những tính năng xác thực trên máy chủ, báo cáo sẽ đưa ra một thiết kế giao diện mẫu dùng cho trang web thử nghiệm.

4.4.1 Trang đăng nhập



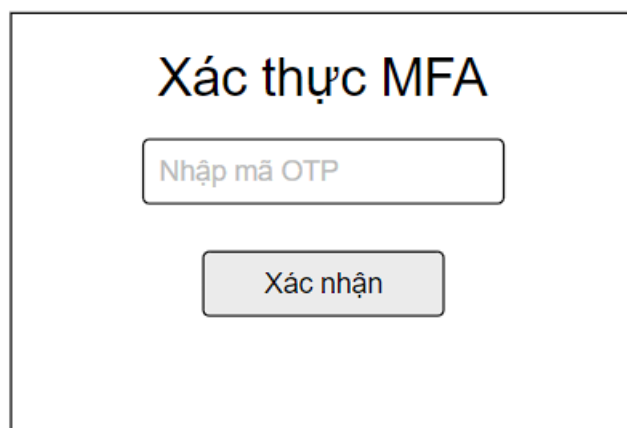
Hình 4.14: Mockup đăng nhập

4.4.2 Trang thiết lập TOTP



Hình 4.15: Mockup trang thiết lập TOTP

4.4.3 Trang xác thực TOTP



The mockup shows a rectangular box with a white background. At the top center, the text "Xác thực MFA" is displayed in a large, bold, black font. Below this, there is a rounded rectangular input field with a light gray border and the placeholder text "Nhập mã OTP" in a light gray font. Underneath the input field is a rounded rectangular button with a light gray background and the text "Xác nhận" in a black font.

Hình 4.16: Mockup trang xác thực TOTP

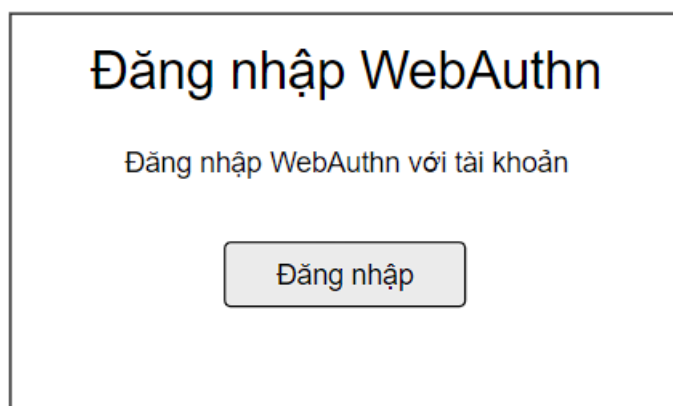
4.4.4 Trang thiết lập WebAuthn



The mockup shows a rectangular box with a white background. At the top center, the text "Đăng ký WebAuthn" is displayed in a large, bold, black font. Below this, there is a paragraph of text in a smaller black font: "Bạn cần phải đăng ký thiết bị cho tài khoản để tiếp tục". Underneath the text is a rounded rectangular button with a light gray background and the text "Đăng ký" in a black font.

Hình 4.17: Mockup đăng ký thiết bị WebAuthn

4.4.5 Trang xác thực WebAuthn



The mockup shows a rectangular box with a white background. At the top center, the text "Đăng nhập WebAuthn" is displayed in a large, bold, black font. Below this, there is a paragraph of text in a smaller black font: "Đăng nhập WebAuthn với tài khoản". Underneath the text is a rounded rectangular button with a light gray background and the text "Đăng nhập" in a black font.

Hình 4.18: Mockup xác thực WebAuthn

CHƯƠNG 5. CÀI ĐẶT VÀ THỬ NGHIỆM

5.1 Xây dựng ứng dụng

5.1.1 Thư viện và công cụ sử dụng

Mục đích	Công cụ	Địa chỉ URL
IDE lập trình	Visual studio code	https://code.visualstudio.com/
Ngôn ngữ lập trình	NodeJS	https://nodejs.org/en
Thư viện xác thực nâng cao	PassportJS	https://www.passportjs.org/
Thư viện cài đặt WebAuthn	SimpleWebAuthn	https://simplewebauthn.dev/
Thư viện cài đặt TOTP	Otplib	https://otplib.yeojz.dev/
Framework phát triển back-end	NestJS	https://nestjs.com/
Framework phát triển front-end	VueJS	https://vuejs.org/

Bảng 5.1: Danh sách thư viện và công cụ sử dụng

5.1.2 Kết quả đạt được

Bằng sự tìm tòi, học hỏi và sự tận tình chỉ bảo của giảng viên hướng dẫn, sau một khoảng thời gian phân tích, thiết kế và xây dựng module xác thực nâng cao đã có các kết quả tích cực. Module đã đáp ứng được toàn bộ các yêu cầu đặt ra trong quá trình phân tích và thiết kế. Các chức năng hoạt động ổn định và cung cấp được các yêu cầu cần thiết của một module xác thực nâng cao.

Cụ thể hơn, dưới đây là thông tin thống kê về ứng dụng

Thông tin	Thống kê
Tổng dung lượng mã nguồn (tính cả thư viện)	314.4 MB
Tổng số file trong module	68
Tổng số collections trong cơ sở dữ liệu	4

Bảng 5.2: Thống kê ứng dụng

5.2 Kiểm thử

5.2.1 Môi trường kiểm thử

Với phạm vi của một đồ án tốt nghiệp cử nhân, để đơn giản hóa quá trình kiểm thử và tăng tính linh hoạt trong các khâu kiểm thử, tôi đã chọn thực hiện kiểm thử trên môi trường local.

Cụ thể môi trường kiểm thử như sau:

Công cụ/môi trường	Phiên bản	Ghi chú
NodeJS	18.12.0	Phiên bản chỉ cần \geq LTS 18.12.0
VueJS	3.2.47	Phiên bản này chỉ áp dụng cho @vue/core package. Các package khác nhau của @vue có thể khác biệt.
NestJS	9.4.3	Phiên bản chỉ cần \geq 9.0.0. Chỉ áp dụng cho @nestjs/core. Các package khác nhau của @nestjs có thể khác biệt
Chrome	Version 114.0.5735.198 (Official Build) (64-bit)	None
OS	Manjaro Linux x86_64	Kernel 5.15.120-1 MANJARO
MongoDB	6.0.6	Đây là phiên bản mongo:latest được pull về từ dockerhub trong quá trình phát triển đồ án
Redis	7.0.10	Đây là phiên bản ứng với image tag redis:alpine được pull về từ dockerhub trong quá trình phát triển đồ án
Google Developer Console	Latest	Sử dụng môi trường test cho ứng dụng bên thứ ba mà google cung cấp

Bảng 5.3: Môi trường kiểm thử

5.2.2 Kiểm thử chức năng SSO sử dụng Google OAuth

STT	Ca kiểm thử	Dữ liệu đầu vào	Kết quả mong đợi	Kết quả thực tế
1	Kiểm thử thông tin đăng nhập không chính xác	Điền giá trị email "testnonexist.xx00xx@gmail.com"	Google thông báo không tìm thấy tài khoản Google của bạn	Đạt
2	Kiểm thử thông tin đăng nhập không chính xác	Điền giá trị email "iwant-totestapis@gmail.com" và điền mật khẩu "123123123"	Google thông báo thông tin đăng nhập sai	Đạt
3	Kiểm thử thông tin đăng nhập chính xác	Điền thông tin đăng nhập chính xác "iwant-totestapi@gmail.com" và mật khẩu "user_vcfv8274"	Google thông báo đăng nhập thành công, chuyển người dùng sang màn consent	Đạt

Bảng 5.4: Ca kiểm thử chức năng đăng nhập một lần sử dụng Google OAuth

5.2.3 Kiểm thử chức năng SSO sử dụng Github OAuth

STT	Ca kiểm thử	Dữ liệu đầu vào	Kết quả mong đợi	Kết quả thực tế
1	Kiểm thử thông tin đăng nhập không chính xác	Điền tên người dùng/email không tồn tại "nonexistent-githubuser"	Github thông báo thông tin đăng nhập không hợp lệ	Đạt
2	Kiểm thử thông tin đăng nhập không chính xác	Tên tài khoản "iwant-totestapis@gmail.com" và mật khẩu "123123"	Github thông báo thông tin đăng nhập không hợp lệ	Đạt
3	Kiểm thử thông tin đăng nhập chính xác	Điền tên tài khoản "iwant-totestapis@gmail.com" và mật khẩu "throwaway1213"	Github thông báo đăng nhập thành công, chuyển người dùng sang màn consent	Đạt

Bảng 5.5: Ca kiểm thử chức năng đăng nhập một lần sử dụng Github OAuth

5.2.4 Kiểm thử chức năng Thiết lập TOTP với Google Authenticator

STT	Ca kiểm thử	Dữ liệu đầu vào	Kết quả mong đợi	Kết quả thực tế
1	Kiểm thử người dùng chưa đăng nhập	Người dùng chưa đăng nhập nhưng yêu cầu thiết lập TOTP	Thông báo lỗi 401, Unauthorized	Đạt
2	Kiểm thử input bắt buộc	Người dùng không điền giá trị mã otp 6 số nhưng vẫn click chọn "confirm"	Thông báo lỗi, trường mã OTP không được để trống	Đạt
3	Kiểm thử mã otp sai	Người dùng điền mã OTP bất kỳ không hợp lệ như "000000" hoặc điền mã OTP hết hạn	Thông báo lỗi, mã xác thực không hợp lệ	Đạt
4	Kiểm thử mã otp đúng	Người dùng điền đúng mã OTP từ ứng dụng Google Authenticator	Thông báo thành công	Đạt

Bảng 5.6: Ca kiểm thử chức năng thiết lập TOTP với Google Authenticator

5.2.5 Kiểm thử chức năng Xác thực với Google Authenticator

STT	Ca kiểm thử	Dữ liệu đầu vào	Kết quả mong đợi	Kết quả thực tế
1	Kiểm thử người dùng chưa đăng nhập	Người dùng chưa đăng nhập nhưng muốn xác thực OTP	Thông báo lỗi 401, Unauthorized	Đạt
2	Kiểm thử người dùng chưa thiết lập TOTP	Người dùng chưa thiết lập TOTP nhưng vẫn yêu cầu xác thực	Thông báo lỗi 401, Unauthorized	Đạt
3	Kiểm thử input bắt buộc	Người dùng không điền giá trị mã otp 6 số nhưng vẫn click chọn "confirm"	Thông báo lỗi, trường mã OTP không được để trống	Đạt
4	Kiểm thử mã otp sai	Người dùng điền mã otp hết hạn hoặc sai mã OTP	Thông báo lỗi, mã xác thực không hợp lệ	Đạt
5	Kiểm thử mã OTP chính xác	Người dùng điền mã OTP sinh ra từ Google Authenticator chính xác	Thông báo thành công	Đạt

Bảng 5.7: Ca kiểm thử xác thực TOTP với Google Authenticator

5.2.6 Kiểm thử chức năng Thiết lập giao thức WebAuthn

STT	Ca kiểm thử	Dữ liệu đầu vào	Kết quả mong đợi	Kết quả thực tế
1	Kiểm thử người dùng chưa đăng nhập	Người dùng chưa đăng nhập nhưng muốn thiết lập giao thức WebAuthn	Thông báo lỗi 401, Unauthorized	Đạt
2	Kiểm thử người dùng không lựa chọn thiết bị khi đăng ký trong khoảng thời gian quy định	Người dùng chọn đăng ký, nhưng không lựa chọn phương pháp đăng ký thiết bị trong thời gian quy định	Trình duyệt hiện prompt "the request timed out", trang web báo lỗi "The request either timed out or was not allowed"	Đạt
3	Kiểm thử người dùng lựa chọn cancel khi được prompt đăng ký thiết bị	Người dùng chọn đăng ký và lựa chọn cancel khi được prompt đăng ký thiết bị	Trình duyệt tắt prompt, trang web báo lỗi "The request either timed out or was not allowed"	Đạt
4	Kiểm thử người dùng thất bại trong việc hoàn tất đăng ký	Người dùng chọn đăng ký nhưng không hoàn thiện được quá trình đăng ký thiết bị trong thời gian quy định	Trình duyệt prompt "The request timed out", trang web báo lỗi "The request either timed out or was not allowed"	Đạt
5	Kiểm thử người dùng chưa bật bluetooth khi đăng ký thiết bị	Người dùng chọn đăng ký nhưng thiết bị chưa bật bluetooth	Trình duyệt prompt "Turn on bluetooth"	Đạt
6	Kiểm thử người dùng đăng ký thiết bị thành công	Người dùng đăng ký thiết bị thành công	Thông báo "Sucessfully registered."	Đạt

Bảng 5.8: Ca kiểm thử thiết lập giao thức WebAuthn

5.2.7 Kiểm thử chức năng Xác thực với WebAuthn

STT	Ca kiểm thử	Dữ liệu đầu vào	Kết quả mong đợi	Kết quả thực tế
1	Kiểm thử người dùng chưa đăng nhập	Người dùng chưa đăng nhập nhưng muốn xác thực với giao thức WebAuthn,	Thông báo lỗi 401, Unauthorized	Đạt
2	Kiểm thử người dùng chưa đăng ký thiết bị	Người dùng chưa đăng ký thiết bị nhưng vẫn muốn xác thực	Thông báo lỗi 400, Bad Request	Đạt
3	Kiểm thử người dùng không lựa chọn thiết bị trong khoảng thời gian quy định	Người dùng chọn xác thực, nhưng không lựa chọn phương pháp đăng ký thiết bị trong thời gian quy định	Trình duyệt hiện prompt "the request timed out", trang web báo lỗi "The request either timed out or was not allowed"	Đạt
4	Kiểm thử người dùng lựa chọn cancel khi được prompt chọn thiết bị xác thực	Người dùng chọn xác thực và lựa chọn cancel khi được prompt chọn thiết bị xác thực	Trình duyệt tắt prompt, trang web báo lỗi "The request either timed out or was not allowed"	Đạt
5	Kiểm thử người dùng thất bại trong việc hoàn tất xác thực	Người dùng chọn xác thực nhưng không hoàn thiện được quá trình xác thực trong thời gian quy định	Trình duyệt prompt "The request timed out", trang web báo lỗi "The request either timed out or was not allowed"	Đạt
6	Kiểm thử người dùng chưa bật bluetooth khi xác thực	Người dùng chọn xác thực nhưng thiết bị chưa bật bluetooth	Trình duyệt prompt "Turn on bluetooth"	Đạt
7	Kiểm thử người dùng xác thực thành công	Người dùng xác thực thành công	Thông báo "Sucessfully registered."	Đạt

Bảng 5.9: Ca kiểm thử xác thực giao thức WebAuthn

CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Các giải pháp và đóng góp nổi bật

Trong phần này, tôi xin được trình bày các bài toán cũng như các giải pháp sáng tạo và đóng góp đáng chú ý của đề án nhằm đáp ứng và vượt qua những thách thức kỹ thuật và mang lại giá trị thực tiễn cho cộng đồng người dùng và ngành công nghiệp.

6.1.1 Bài toán xác thực với thiết bị di động

a, Đặt vấn đề

Xác thực với thiết bị di động là một phương thức xác thực đa bước khá phổ biến, với những ví dụ điển hình về các dịch vụ sử dụng nó bao gồm có Google với chức năng prompt trên thiết bị di động, Github với ứng dụng Github mobile hoặc Shopee với ứng dụng trên thiết bị di động. Xác thực đa bước với thiết bị di động có những lợi ích rõ ràng do sự phổ biến của các thiết bị di động hiện nay, khi mà bất cứ ai cũng sở hữu một thiết bị di động trong người mọi lúc mọi nơi.

Vấn đề lớn đối với việc xác thực trên thiết bị di động là không có sự thống nhất trong giao thức xác thực, nghĩa là một dịch vụ, mỗi ứng dụng lại đưa ra các giải pháp của riêng mình, từ đó nảy sinh ra các vấn đề độc lập và các bài toán không có cách giải quyết tiêu chuẩn.

Bên cạnh đó, một vấn đề khác nảy sinh là sự cần thiết phải xây dựng một ứng dụng di động độc lập. Đối với các nền tảng lớn, việc xây dựng ứng dụng di động là hợp lý và thiết yếu, tuy nhiên đối với các ứng dụng web đang trong quá trình phát triển về quy mô thì rõ ràng việc xây dựng thêm một ứng dụng di động là một sự đánh đổi tài nguyên, do đó, có rất nhiều ứng dụng web ở mức độ vừa và nhỏ lựa chọn chấp nhận bỏ qua phương pháp xác thực đa bước sử dụng thiết bị di động mà chọn các phương pháp khác thay thế khác.

b, Giải pháp

Đối mặt với bài toán xác thực đa bước sử dụng thiết bị di động, tôi đã quyết định ứng dụng một phương pháp xác thực mới, đó là xác thực sử dụng passkey.

Passkey là mật khẩu là chứng chỉ kỹ thuật số, là một phương pháp xác thực cho các ứng dụng web. Nó được phát triển bởi và định nghĩa bởi rất nhiều tổ chức, trong đó có thể kể đến Apple, Google, Microsoft,.. Và được đưa vào áp dụng và khuyến khích bởi FIDO (Fast Identity Online) và W3C (World Wide Web Consortium).

WebAuthn là tiêu chuẩn xác thực được định nghĩa bởi W3C như là một thành

phần cốt lõi cho việc xác thực người dùng sử dụng passkey trên nền tảng web[5].

Tôi chọn sử dụng Passkeys và WebAuthn để giải quyết bài toán xác thực đa bước này là vì 2 lý do chính. Thứ nhất, việc xây dựng một ứng dụng di động chỉ phục vụ cho mục đích xác thực là một sự lãng phí tài nguyên. Phần lớn các ứng dụng di động được phát triển với mục đích phục vụ nhu cầu của người dùng và doanh nghiệp, với tính năng nâng cao bảo mật được thêm vào như là một điểm cộng.

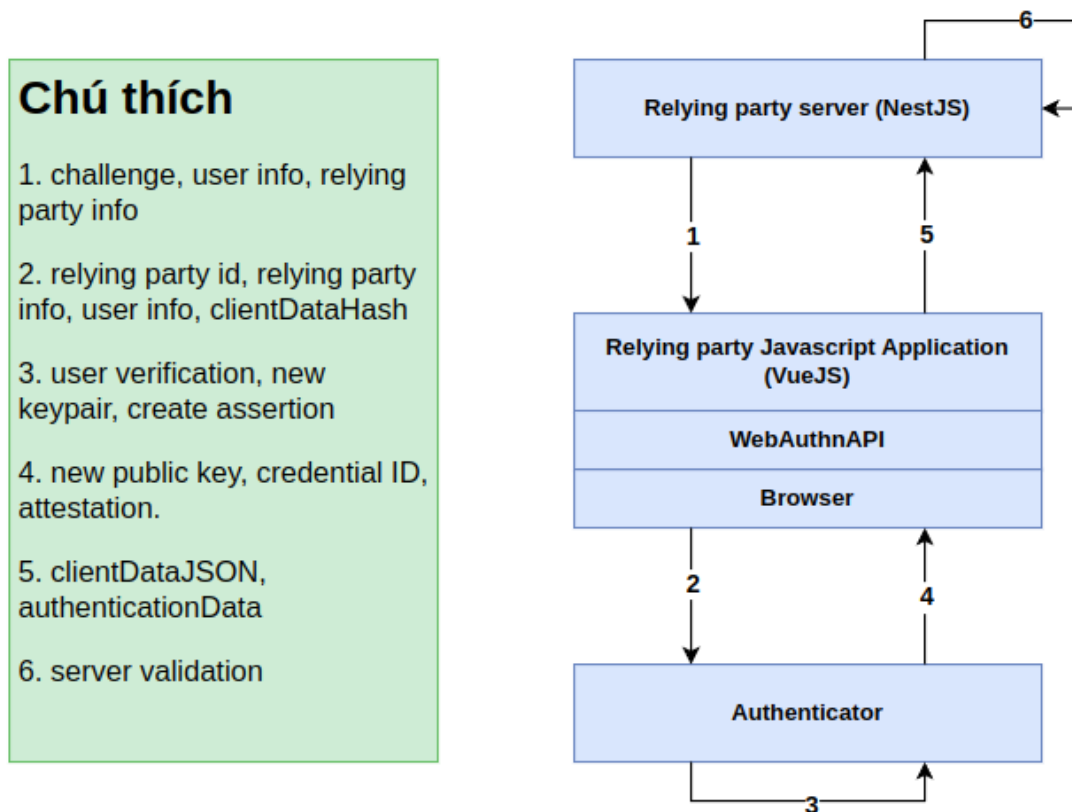
Thứ hai, WebAuthn có thể trực tiếp hoạt động trơn tru trên rất nhiều nền tảng và thiết bị hiện đại mà không cần phải có cài đặt hay triển khai bổ sung; điều này là điểm cộng lớn vì nó vừa mang lại tính tiện dụng cho người dùng cuối, vừa giúp tiết kiệm tài nguyên cho các đội ngũ phát triển. Ngoài ra, việc sử dụng một tiêu chuẩn phổ biến được khuyến nghị bởi các tổ chức lớn sẽ đảm bảo rằng ứng dụng của bạn được hậu thuẫn bởi một cộng đồng các nhà phát triển hùng hậu. Các lỗ hổng bảo mật đi kèm với tiêu chuẩn sẽ cùng được nghiên cứu và giải quyết bởi cả một cộng đồng, và các tính năng mới sẽ được phát triển và tích hợp vào trong tiêu chuẩn mà không yêu cầu bạn phải dành tài nguyên ra để nghiên cứu.

c, Kết quả đạt được

Đề tài đã thành công trong việc áp dụng phương pháp xác thực sử dụng Passkey với WebAuthn như là một giải pháp xác thực đa bước.

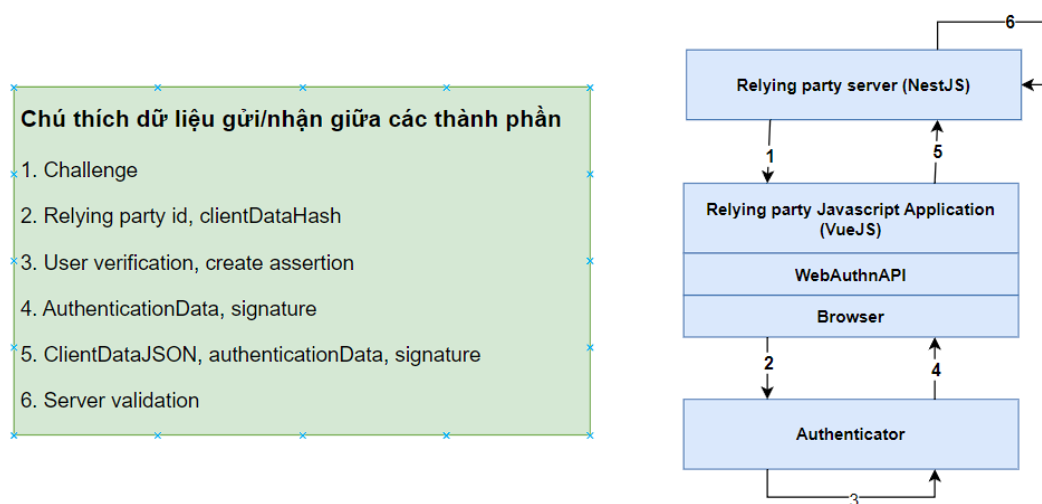
Để đơn giản hóa quá trình cài đặt WebAuthn, tôi đã ứng dụng thư viện SimpleWebAuthn, một thư viện đóng gói các API của WebAuthn thành một API đơn giản hơn, và đã thành công trong việc cài đặt luồng hoạt động của WebAuthn với framework NestJS và CSDL MongoDB.

Cụ thể hơn, luồng hoạt động đăng ký thiết bị của WebAuthn là như sau:



Hình 6.1: Luồng đăng ký thiết bị của WebAuthn

Luồng xác thực của WebAuthn như sau: Module xác thực đóng vai trò như là



Hình 6.2: Luồng xác thực của WebAuthn

relying party server, đã cài đặt thành công các chức năng cần thiết để hoàn thành vai trò của mình dưới dạng API. Ví dụ, tại bước 1, server tạo ra một object chứa các thông tin cần thiết như user info, relying party info và challenge và trả về cho Client tại API `/webauthn/generate-registration-options`. Đoạn mã thực

thi nhiệm vụ này có thể được tìm thấy tại:

`libs/auth/src/controllers/webauthn.controller.ts`

```

72     const options = generateRegistrationOptions({
73       rpID: this.config.rpID,
74       rpName: this.config.rpName,
75       userID: user._id.toString(),
76       userName: user.email,
77       userDisplayName: user.displayName,
78       attestationType: 'none',
79       authenticatorSelection: {
80         residentKey: 'discouraged',
81       },
82       excludeCredentials: userAuthenticators.map((authenticator) => ({
83         id: Buffer.from(authenticator.credentialID),
84         type: 'public-key',
85         transports: authenticator.transports as AuthenticatorTransport[],
86       })),
87     });
88     this.redisService
89       .getClient()
90       .set(`challenge_${user._id.toString()}`, options.challenge, 'EX', 600);
91     return options;

```

Hình 6.3: Đoạn mã sinh tùy chọn đăng ký cho WebAuthn

6.1.2 Bài toán tách module thành thư viện

a, Đặt vấn đề

Module được xây dựng đã tích hợp thành công các mục tiêu xác thực nâng cao được đề ra. Tuy nhiên, trên thực tế, để một module có tính ứng dụng cao, nó phải có khả năng được tái sử dụng vào nhiều ứng dụng web khác nhau và được tùy chỉnh tùy theo nhu cầu của các nhà phát triển.

Có thể nói, bài toán bóc tách thư viện từ module sẽ được chia thành 2 bài toán nhỏ hơn là bài toán đóng gói và bài toán tùy chỉnh cho module.

b, Giải pháp

Đối với bài toán con đóng gói, giải pháp đơn giản mà tương thích nhất với framework NestJS chính là sử dụng NestJS library, một tính năng được phát triển để hỗ trợ cho các kỹ sư phần mềm bóc tách logic trong mã nguồn của họ thành các module riêng biệt gọi là thư viện, có thể được sử dụng cho nhiều dự án khác nhau. NestJS cung cấp các công cụ cũng như câu lệnh CLI cho phép sinh ra một thư viện nhanh chóng.

```
$ nest generate library auth
```

Đối với bài toán tùy chỉnh còn lại, tôi đã chọn giải pháp là Dynamic Module - Module động, một tính năng khác mà NestJS cung cấp cho phép các nhà phát triển định nghĩa các tùy chỉnh dưới dạng Object khi khai báo Module. Cơ chế này cho phép người phát triển module định nghĩa các tùy chỉnh mong muốn, và người sử

dụng module tùy chỉnh hoạt động của module nó trong mã nguồn của mình. Bên cạnh module động, tôi còn sử dụng cơ chế lưu trữ nội bộ liên tục, một cho chế cho phép truyền một ngữ cảnh qua các callback. Cơ chế này cho phép đảm bảo các tùy chỉnh được duy trì xuyên suốt vòng đời của một yêu cầu HTTP, đảm bảo khả năng tùy chỉnh của người dùng.

c, Kết quả đạt được

Module đã thành công trong việc áp dụng các giải pháp kể trên. Kết quả đạt được của việc sử dụng NestJS library là tôi đã tạo được ra một thư viện hoàn chỉnh có thể được sử dụng cho nhiều dự án khác nhau. Tuy nhiên, do bản thân thư viện sử dụng khá nhiều thư viện phụ thuộc, việc setup trong một project trắng khá phức tạp.

Để đơn giản hóa quá trình tích hợp thư viện, tôi đã chuyển đổi module kết quả thành một github template tích hợp sẵn các file cài đặt và ví dụ cần thiết để sử dụng. Mã nguồn chính của module có thể được tìm thấy tại `libs/auth/src`.

Đối với giải pháp sử dụng module động, tôi đã thành công trong việc bóc tách các giá trị tĩnh thành các tùy chỉnh cho phép người dùng thay đổi tùy ý. Cơ chế lưu trữ nội bộ liên tục cũng được tích hợp thành công sử dụng thư viện NestJS CLS, một thư viện cho phép khởi tạo duy trì ngữ cảnh xuyên suốt vòng đời của một yêu cầu, cung cấp tính chất tùy biến và linh động cần có cho người dùng.

Tóm lại, người dùng có thể tùy chỉnh cách thức hoạt động và các giá trị của module thông qua 3 nguồn chính sau:

1. Biến môi trường (env) được sử dụng cho các giá trị tĩnh, chỉ khởi tạo 1 lần khi ứng dụng bắt đầu.
2. Cài đặt module động, cũng được khởi tạo một lần nhưng có thể là các logic phức tạp hơn do nằm trực tiếp trong code.
3. File json chứa tùy chọn của cls, được đọc và nạp thành ngữ cảnh tại đầu mỗi yêu cầu, cho phép tùy chỉnh các hoạt động của các phương pháp xác thực nâng cao như bật/tắt, tùy chỉnh scope của OAuth,... Giá trị của file JSON có thể được edit sử dụng API và yêu cầu phải được điền trước khi chạy server.

```
1 REDIS_HOST=localhost
2 REDIS_PORT=6379
3 REDIS_PASSWORD=eYVX7EwVmmxKPCDmwMtyKVge8oLd2t81
4 REDIS_AUTH_CONFIG_KEY=AUTH_CONFIG
5
6 MONGO_URI=mongodb://namdp:123123@localhost:27017/test?authSource=admin
7
8 POSTGRES_PASSWORD=FLbzSqazTR4Pxt4h
9 POSTGRES_HOST=db.vuslbjcakzpgzzrdzww0.supabase.co
10 PORT=5432
11 POSTGRES_DB=postgres
12 POSTGRES_USER=postgres
13
14 JWT_SECRET=VERYSECRET
15 TWO_FACTOR_AUTHENTICATOR_APP_NAME=Nest-OAUTHMFA-plugin
```

Hình 6.4: Biến môi trường

```
55 export interface IAuthModuleOptions {
56   serveStatic: {
57     rootPath: string;
58     exclude: string[];
59   };
60   cls: {
61     configFilePath: string;
62   };
63   env: {
64     prefix: string;
65   };
66   routes: {
67     redirect: {
68       successAuthenticatedWithProvider:
69         | string
70         | ((user: UserDocument) => string);
71     otpAuthenticate: string | ((user: UserDocument) => string);
72     otpSetup: string | ((user: UserDocument) => string);
73     webAuthnRegister: string | ((user: UserDocument) => string);
74     webAuthnAuthenticate: string | ((user: UserDocument) => string);
75   };
76 };
77 }
```

Hình 6.5: Interface cho cài đặt module động

```

1  {
2    "willAuthenticate": true,
3    "mfaEnforce": true,
4    "mfaType": "otp",
5    "webAuthnConfig": {
6      "rpName": "Webauthn test",
7      "rpID": "localhost",
8      "origin": "http://localhost:5173"
9    },
10   "githubProviderOptions": {
11     "active": true,
12     "clientSecret": "164689da5dda4f4d6cf8c441ea24c56b83cf37f6",
13     "clientId": "1827e2b0a178598e180f",
14     "callbackURL": "http://localhost:5173/api/auth/github/callback",
15     "scope": ""
16   },
17   "googleProviderOptions": {
18     "active": true,
19     "clientSecret": "GOCSPX-0EZusdnNNpGzRMsr7xNRPMWnOPu",
20     "clientId": "527158108169-ocok1u080nt9thrum0trnu2g011cakb8.apps.googleusercontent.com",
21     "callbackURL": "http://localhost:5173/api/auth/google/callback",
22     "scope": "openid email profile"
23   }
24 }

```

Hình 6.6: File JSON chứa cls config

6.2 Kết luận

Đề tài đã thành công xây dựng một module xác thực nâng cao cho framework NestJS, tích hợp các giải pháp xác thực nâng cao và cung cấp khả năng tùy biến linh hoạt. Ngoài ra đề tài cũng thành công trong việc bóc tách module thành một thư viện độc lập cho phép tái sử dụng module trong nhiều dự án khác nhau.

Trong suốt quá trình làm đồ án tốt nghiệp, tôi tự nhận định mình đã hoàn thành các khối nhiệm vụ được đề ra cho đề tài từ quá trình phân tích, thiết kế. Tôi cũng đã học hỏi được nhiều kiến thức, kinh nghiệm từ phạm trù kỹ thuật cho đến kỹ năng phân tích, lập luận hay kỹ năng quản lý tiến độ dự án.

Khối chức năng lớn thứ nhất của đồ án là xác thực một lần đã hoàn thiện với tính năng xác thực một lần sử dụng 2 nền tảng cung cấp dịch vụ OAuth lớn là Google và Github.

Khối chức năng lớn thứ 2 là xây dựng giải pháp xác thực đa bước cũng hoạt động, nhờ vào ứng dụng công nghệ TOTP với Google Authenticator, và tiêu chuẩn WebAuthn như là một lựa chọn xác thực đa bước bổ sung.

Cuối cùng là khả năng tùy biến, cho phép người dùng linh hoạt áp dụng những tùy chọn, cài đặt của riêng mình để điều khiển cách thực hoạt động của 2 khối chức năng trên ngay trong thời gian chạy mà không cần phải khởi động lại dịch vụ web, cho phép một trải nghiệm tức thời và gần như không có downtime.

6.3 Hướng phát triển

Về mặt chức năng, module vẫn còn nhiều hạng mục có thể cải thiện và bổ sung để giúp tăng tính ứng dụng hơn nữa. Thứ nhất, module có thể tích hợp thêm các nền tảng xác thực một lần phổ biến khác như twitter, facebook, microsoft... hoặc cho phép người dùng thêm một dịch vụ cung cấp OAuth bất kỳ với các tham số tùy chỉnh. Điều này sẽ giúp tăng phạm vi bao quát của module, hỗ trợ nhiều nền tảng hơn và nhiều phân khúc người dùng hơn.

Thứ hai, module có thể bổ sung thêm một giao diện cài đặt cho Admin sẵn sàng ngay bên trong module, và cung cấp khả năng tùy chỉnh cho giao diện đó. Cuối cùng, module có thể hỗ trợ thêm nhiều CSDL hơn, cho phép người dùng trực tiếp tùy chỉnh. Hiện nay, MongoDB được lựa chọn là CSDL chính vì tính linh hoạt của nó, tuy nhiên, PostgreSQL, MySQL đều là các lựa chọn phổ biến và nhiều người dùng có thể không mong muốn sử dụng một hệ CSDL khác chỉ để phục vụ phần xác thực.

TÀI LIỆU THAM KHẢO

- [1] B. Krstic, *Impressive password statistics to know in 2023*, 2023. [Online]. Available: <https://webtribunal.net/blog/password-stats/>.
- [2] Viettel Threat Intelligence, *Báo cáo tình hình nguy cơ an toàn thông tin năm 2022*, 2022. [Online]. Available: <https://blog.viettelcybersecurity.com/bao-cao-tinh-hinh-nguy-co-attn-nam-2022-viettel-threat-intelligence/>.
- [3] G. Mondaca, *Two-factor authentication statistics: First line of defense*, 2022. [Online]. Available: <https://eftsure.com/statistics/two-factor-authentication-statistics/>.
- [4] *The OAuth 2.0 Authorization Framework*. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc6749>.
- [5] W3C, *Web authentication: An api for accessing public key credentials level 2*, 2021. [Online]. Available: <https://www.w3.org/TR/2021/REC-webauthn-20210408/>.
- [6] *Nestjs official documentation*. [Online]. Available: <https://docs.nestjs.com/>.
- [7] *VueJS Glossary*. [Online]. Available: <https://vuejs.org/glossary/>.
- [8] *MongoDB official documentation*. [Online]. Available: <https://www.mongodb.com/docs/manual/>.
- [9] *Redis official Documentation*. [Online]. Available: <https://redis.io/docs/>.
- [10] *REST Architecture*. [Online]. Available: <https://restfulapi.net/>.

PHỤ LỤC