

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



MẬT MÃ VÀ AN NINH MẠNG (CO3069)

Báo cáo bài tập lớn

Hiện thực Single Sign-On (SSO) trên ứng dụng Web

GVHD: Nguyễn Cao Đạt

SV thực hiện:	Trương Nguyễn Hoàng Anh	2210147
	Trần Tuấn Kha	2211418
	Nguyễn Trường Thái Khang	2211458
	Lâm Hoàng Tân	2213054

Thành phố Hồ Chí Minh, tháng 10 năm 2024

Danh sách thành viên

STT	Họ và tên	MSSV	Nhiệm vụ	Đóng góp
1	Trương Nguyễn Hoàng Anh	2210147	<ul style="list-style-type: none">• Tạo Wirefame• Test chức năng của web• Hiện thực SSO	100%
2	Trần Tuấn Kha	2211418	<ul style="list-style-type: none">• Tạo Host/Local• Tạo IP• Hiện thực SSO	100%
3	Nguyễn Trường Thái Khang	2211458	<ul style="list-style-type: none">• Front-end• Back-end• Hiện thực SSO	100%
4	Lâm Hoàng Tân	2213054	<ul style="list-style-type: none">• Thiết kế UX/UI• Front-end• Hiện thực SSO• L^AT_EX	100%

Mục lục

1	Giới thiệu	1
2	Cơ sở lý thuyết	2
2.1	Single Sign-On	2
2.1.1	Tổng quan	2
2.1.2	Thành phần chính	2
2.1.3	Nguyên lý hoạt động	3
2.1.4	Ưu điểm	4
2.1.5	Rủi ro và thách thức	4
2.2	Các loại giao thức thông dụng	5
2.2.1	OpenID Connect	5
2.2.2	Central Authentication Service	8
2.2.3	Security Assertion Markup Language	10
2.2.4	So sánh các giao thức	13
2.3	Một số mô hình triển khai Single Sign-On	13
3	Phân tích và thiết kế	15
3.1	Phân tích yêu cầu	15
3.1.1	Yêu cầu hệ thống	15
3.1.2	Yêu cầu chức năng	15
3.2	Thiết kế hệ thống	15
3.2.1	Kiến trúc hệ thống	15
3.2.2	Luồng hoạt động	17
3.2.3	Giao diện người dùng	18
4	Hiện thực và đánh giá	21
4.1	Cài đặt và cấu hình SSO	21
4.2	Hiện thực tính năng đăng nhập trên Website	22
4.2.1	Đăng nhập người dùng	22
4.2.2	Đăng xuất người dùng	26
4.2.3	Đăng ký tài khoản	27
4.2.4	Kết quả hiện thực	28
4.3	Đánh giá kết quả	28
5	Kết luận	29
	Tài liệu tham khảo	30
	Phụ lục	31

1 Giới thiệu

Trong bối cảnh số hoá hiện nay, các doanh nghiệp và tổ chức ngày càng sử dụng nhiều ứng dụng, dịch vụ để hỗ trợ cho các hoạt động kinh doanh và quản lý. Tuy nhiên, điều này cũng dẫn đến một vài thách thức lớn, cụ thể là trong việc quản lý tài khoản của người dùng. Việc phải nhớ và sử dụng nhiều tài khoản cho những ứng dụng có thể liên kết với nhau không chỉ gây ra bất tiện, mà nó còn làm tăng nguy cơ bảo mật, đặc biệt là thông tin cá nhân và mật khẩu của người dùng.

Đứng trước tình hình đó, hệ thống Single Sign-On (SSO) được coi là một giải pháp hiệu quả để khắc phục các vấn đề nêu trên. Với hệ thống SSO, người dùng chỉ cần đăng nhập một lần duy nhất để truy cập đồng thời vào các ứng dụng, dịch vụ và được nâng cao trải nghiệm người dùng mà vẫn đảm bảo được bảo mật thông tin cá nhân.

Chính vì vậy, nhóm nghiên cứu quyết định thực hiện đề tài ***Hiện thực Single Sign-On (SSO) trên ứng dụng Web*** cho Bài tập lớn trong chương trình học môn Mật mã và An ninh mạng (CO3069).

Để tiến hành đề tài trên, nhóm tiến hành nghiên cứu các khái niệm và nguyên lý hoạt động của hệ thống Single Sign-On. Đồng thời, nhóm cũng tìm hiểu các hệ thống thông dụng để từ đó đánh giá điểm mạnh cũng như điểm yếu của các giao thức tương ứng. Cuối cùng, với mục đích làm rõ đề tài, nhóm cũng hiện thực một chương trình với tính năng đăng nhập trên Website kết hợp với hệ thống SSO và đưa ra kết quả, giải thích và đánh giá chương trình đã hiện thực.

Ngoài phần **Giới thiệu**, phần **Kết luận**, **Danh mục tài liệu tham khảo** và **Phụ lục**, nội dung chính của đề tài gồm 03 phần.

Phần 2: *Cơ sở lý thuyết*, nêu rõ các khái niệm và nguyên lý liên quan đến hệ thống SSO.

Phần 3: *Phân tích và thiết kế*, nêu lên quy trình thiết kế Website với tính năng đăng nhập dựa trên hệ thống SSO và phân tích các yêu cầu do nhóm nghiên cứu đã đặt ra.

Phần 4: *Hiện thực và đánh giá*, trình bày kết quả do nhóm nghiên cứu đã hiện thực chương trình để từ đó đưa ra giải thích và đánh giá kết quả thu được.

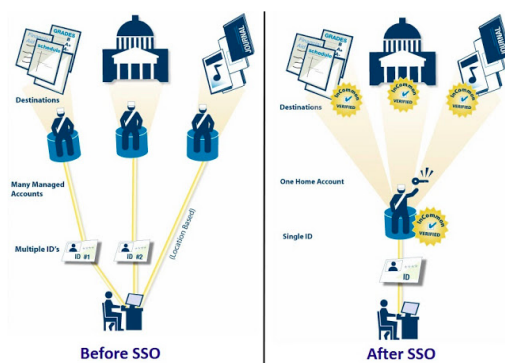
2 Cơ sở lý thuyết

2.1 Single Sign-On

2.1.1 Tổng quan

Single Sign-On (SSO) là một phương thức xác thực cho phép người dùng đăng nhập vào nhiều ứng dụng và dịch vụ khác nhau chỉ với một lần nhập thông tin tài khoản duy nhất. Điều này có nghĩa là người dùng chỉ cần nhập tên đăng nhập và mật khẩu một lần, sau đó có thể truy cập vào các dịch vụ liên quan mà không cần phải đăng nhập lại.

Trước khi có SSO, người dùng đã phải nhập các tài khoản và mật khẩu cho từng ứng dụng mỗi khi họ đăng nhập vào các ứng dụng khác nhau hoặc các hệ thống trong cùng một phiên. Điều này khiến cho người dùng tốn nhiều thời gian, đặc biệt là trong môi trường thông tin đa phương tiện như hiện nay, khi mà người sử dụng phải đăng nhập mỗi khi họ truy cập vào một hệ thống mới từ máy tính của họ. Do vậy, với hệ thống có nhiều website và ứng dụng thì việc sử dụng SSO là rất cần thiết nhằm đem lại nhiều thuận tiện cho người dùng và tăng tính năng bảo mật.



Hình 1: Trước và sau khi áp dụng Single Sign-On

Ví dụ, người dùng sử dụng các dịch vụ của Google như Gmail, Google Drive, Youtube,... khi chưa có SSO thì với mỗi dịch vụ ta phải nhập thông tin để xác thực. Khi áp dụng SSO, người dùng chỉ cần sử dụng một email duy nhất của Google để có thể đăng nhập và khai thác sử dụng tất cả các dịch vụ và ứng dụng của hệ thống. Điều đó thực sự mang lại sự thuận tiện và tiết kiệm thời gian cho người dùng, khi không phải đăng ký bất kỳ tài khoản nào khác nữa.

2.1.2 Thành phần chính

Single Sign-On hoạt động dựa trên sự tương tác giữa tám thành phần chính. Các thành phần này phối hợp với nhau để mang lại trải nghiệm đăng nhập an toàn và hiệu quả cho người dùng.

Người dùng, hay **User**, là cá nhân hoặc thực thể cần truy cập vào nhiều ứng dụng hoặc dịch vụ khác nhau. Họ tương tác với hệ thống SSO thông qua một giao diện đăng nhập duy nhất.

Identity Provider, hay **IdP**, là hệ thống hoặc dịch vụ chịu trách nhiệm xác thực danh tính của người dùng. IdP quản lý các thông tin đăng nhập và tạo ra các Token xác thực để xác nhận danh tính

người dùng cho các ứng dụng khác.

Service Provider, hay **SP**, là các ứng dụng hoặc hệ thống mà người dùng muốn truy cập, chẳng hạn như ứng dụng web, ứng dụng di động, dịch vụ đám mây. SP dựa vào IdP để xác thực danh tính của người dùng. Khi nhận được token xác thực từ IdP, SP sẽ xác minh token và cho phép người dùng truy cập vào dịch vụ.

Authentication Protocol, hay **giao thức xác thực**, là các quy tắc và tiêu chuẩn được sử dụng để trao đổi thông tin xác thực giữa IdP và SP.

Authentication Token là một đoạn dữ liệu mã hóa chứa thông tin về danh tính và quyền truy cập của người dùng, được tạo ra bởi IdP sau khi người dùng đã được xác thực. SP sử dụng token này để xác minh danh tính người dùng mà không cần yêu cầu thông tin đăng nhập lại.

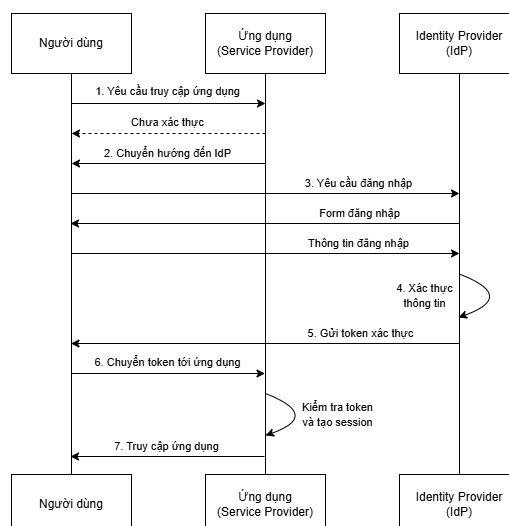
Client là phần mềm mà người dùng tương tác trực tiếp, như trình duyệt web hoặc ứng dụng di động. Client chịu trách nhiệm chuyển tiếp thông tin đăng nhập từ người dùng đến IdP và chuyển tiếp Token từ IdP đến SP.

Session Management System là hệ thống cho phép gửi thông tin về phiên làm việc của người dùng sau khi họ đã đăng nhập, giúp duy trì trạng thái đăng nhập khi chuyển đổi giữa các ứng dụng khác nhau. Hệ thống có thể sử dụng cookies, session tokens hoặc các cơ chế lưu trạng thái khác.

Access Control mô tả các quy tắc và chính sách xác định quyền truy cập của người dùng vào các tài nguyên và dịch vụ. Sau khi xác thực thành công, SP sẽ tiến hành sử dụng thông tin từ IdP để quyết định quyền truy cập của người dùng.

2.1.3 Nguyên lý hoạt động

Nguyên lý hoạt động của SSO thường phụ thuộc vào các giao thức mà doanh nghiệp, công ty muốn sử dụng đến. Một quy trình hoạt động cơ bản của SSO được biểu diễn qua Hình 2.



Hình 2: Nguyên lý hoạt động cơ bản của SSO

Đầu tiên, người dùng tiến hành truy cập vào một ứng dụng, hoặc một hệ thống yêu cầu xác thực (Service Provider, SP). Lúc này, ứng dụng nhận thấy rằng người dùng chưa được xác thực nên ứng dụng

chuyển hướng người dùng sang trang đăng nhập của Identity Provider (IdP).

Tại trang đăng nhập của IdP, người dùng cần phải nhập tài khoản và mật khẩu của mình. Nếu ứng dụng sử dụng xác thực hai yếu tố, ngoài việc nhập thông tin, ứng dụng cũng có thể yêu cầu người dùng cung cấp mã OTP được gửi hoặc xác nhận thông qua ứng dụng xác thực.

Sau khi xác thực thành công, IdP sẽ tạo ra một Token (thường là một chuỗi ký tự) chứa thông tin của người dùng, bao gồm danh tính và quyền truy cập. Token sẽ được gửi lại qua browser của người dùng hoặc thông qua server, bằng cách chuyển hướng người dùng trở lại ứng dụng và kèm theo Token đã được tạo ra.

Kế tiếp, ứng dụng tiến hành kiểm tra tính hợp lệ của Token vừa nhận được (có thể kiểm tra thông qua thời gian hết hạn, hoặc chữ ký số). Nếu Token hợp lệ, ứng dụng sẽ tạo một phiên làm việc (session) cho người dùng và lưu trữ các thông tin cần thiết.

Sau khi phiên làm việc được thiết lập, người dùng có thể truy cập vào chức năng và dữ liệu trong ứng dụng mà không cần phải đăng nhập lại. Nếu người dùng truy cập vào các ứng dụng khác được liên kết với cùng IdP, họ sẽ không cần phải đăng nhập lại vì đã thiết lập xác thực, đúng với tính chất của hệ thống SSO.

2.1.4 Ưu điểm

Đối với người dùng, SSO mang lại sự thuận tiện bằng cách cho phép họ chỉ cần đăng nhập một lần và sau đó có thể truy cập vào nhiều ứng dụng khác mà không cần phải đăng nhập lại. Thay vì phải nhớ và quản lý nhiều tài khoản và mật khẩu khác nhau cho từng ứng dụng, người dùng chỉ cần duy trì một bộ thông tin đăng nhập duy nhất. Điều đó giúp giảm khả năng người dùng sử dụng mật khẩu yếu hoặc tái sử dụng mật khẩu giữa các ứng dụng khác nhau.

Đối với các tổ chức và doanh nghiệp, việc triển khai SSO giúp tối ưu hóa hiệu suất làm việc bằng cách giảm thời gian và công sức cần thiết để đăng nhập vào các ứng dụng khác nhau, qua đó giúp giảm chi phí vận hành và hỗ trợ kỹ thuật. Đồng thời, việc quản lý tài khoản và truy cập cũng trở nên dễ dàng hơn, đặc biệt trong các tổ chức lớn có hàng ngàn người dùng.

Ngoài ra, SSO cũng có thể tích hợp dễ dàng với các ứng dụng và dịch vụ đa dạng. Hầu hết các hệ thống và ứng dụng hiện đại đều hỗ trợ các giao thức phổ biến như SAML (Security Assertion Markup Language) và OAuth (Open Authorization), giúp việc triển khai SSO trở nên đơn giản và linh hoạt.

2.1.5 Rủi ro và thách thức

Thực tế, SSO không tự cải thiện bảo mật. SSO cho phép người dùng truy cập nhiều hệ thống khác nhau chỉ với một lần đăng nhập duy nhất. Tuy nhiên, nếu tài khoản chính của SSO bị xâm phạm, hacker có thể dễ dàng truy cập vào tất cả các ứng dụng liên kết. Điều này khiến hệ thống dễ bị tổn thương hơn nếu không được bảo mật tốt.

Bên cạnh đó, nếu token của người dùng bị tấn công, hacker cũng có thể giả mạo danh tính của người dùng và truy cập vào các ứng dụng, dịch vụ nhằm gây rối cho người dùng.

Thông thường, để tăng tính bảo mật cho SSO, các tổ chức, doanh nghiệp thường thiết lập các cơ chế bảo mật mạnh mẽ như xác thực hai yếu tố (2FA), giám sát hoạt động bất thường và mã hóa toàn bộ dữ

liệu. Nếu các biện pháp này không được thực hiện nghiêm ngặt, hệ thống dễ dàng trở thành mục tiêu của các cuộc tấn công mạng.

Những rủi ro trên đã đặt ra rất nhiều thách thức lớn cho các doanh nghiệp, tổ chức. Thách thức lớn nhất chính là những tác động khi người dùng bị mất quyền truy cập vào các ứng dụng. Nếu người dùng quên mật khẩu, gặp sự cố không thể truy cập vào tài khoản SSO, hoặc IdP bị lỗi, bị tấn công thì họ sẽ mất quyền truy cập vào tất cả các ứng dụng liên quan. Điều này có thể gây gián đoạn công việc và ảnh hưởng đến năng suất làm việc.

Ngoài ra, việc triển khai SSO yêu cầu đồng bộ và tích hợp giữa nhiều ứng dụng và hệ thống khác nhau. Các tổ chức có thể gặp khó khăn khi phải tương thích các ứng dụng cũ hoặc không hỗ trợ các giao thức hiện đại như OAuth, SAML hay OpenID Connect. Đồng thời, việc triển khai cũng bị phụ thuộc vào các nhà cung cấp. Nếu nhà cung cấp này gặp sự cố hoặc bị tấn công, toàn bộ hệ thống của người dùng cũng sẽ bị ảnh hưởng.

2.2 Các loại giao thức thông dụng

2.2.1 OpenID Connect

a/ Mô tả

OpenID Connect (OIDC) là một tiêu chuẩn xác thực được phát triển bởi OpenID Foundation, dựa trên nền tảng của OAuth2. OIDC là sự kết hợp giữa OAuth2 và một số phần mở rộng, nhằm cung cấp một giải pháp xác thực người dùng đơn giản và an toàn cho các ứng dụng web, di động và API.

Mục đích chính của OIDC là giúp các client xác định danh tính của người dùng thông qua một authentication server mà người dùng tin tưởng. Điều này cho phép người dùng sử dụng tài khoản hiện có (ví dụ, tài khoản Google, Facebook) để đăng nhập vào nhiều ứng dụng khác nhau mà không cần tạo tài khoản riêng cho từng ứng dụng. Quá trình này được gọi là Single Sign-On (SSO).

Trong quá trình xác thực, OIDC cung cấp thông tin về người dùng dưới dạng một JSON Web Token (JWT) gọi là ID token. ID token chứa các thông tin như định danh người dùng, thời gian hết hạn của token và các thông tin khác mà client yêu cầu. JWT là một định dạng token tiêu chuẩn, dễ giải mã và xác minh tính hợp lệ của nó.

OIDC hoạt động trên các luồng xác thực của OAuth2, ví dụ như Authorization Code Flow, Implicit Flow và PKCE. Tuy nhiên, trong khi OAuth2 chỉ cung cấp access token để truy cập tài nguyên của người dùng, OIDC còn cung cấp ID token để xác định danh tính người dùng.

Việc sử dụng OIDC giúp giảm thiểu các rủi ro bảo mật liên quan đến việc xử lý mật khẩu của người dùng, giảm bớt công việc cho các nhà phát triển ứng dụng trong việc xây dựng và duy trì hệ thống xác thực, đồng thời tăng trải nghiệm người dùng khi họ không cần nhớ nhiều mật khẩu cho các ứng dụng khác nhau.

b/ Nguyên lý hoạt động

OpenID Connect được hoạt động dựa trên nền tảng OAuth2, kết hợp thêm một số phần mở rộng để cung cấp chức năng xác thực người dùng. Nhìn chung, cách thức hoạt động của OpenID Connect sẽ trải qua 05 bước.

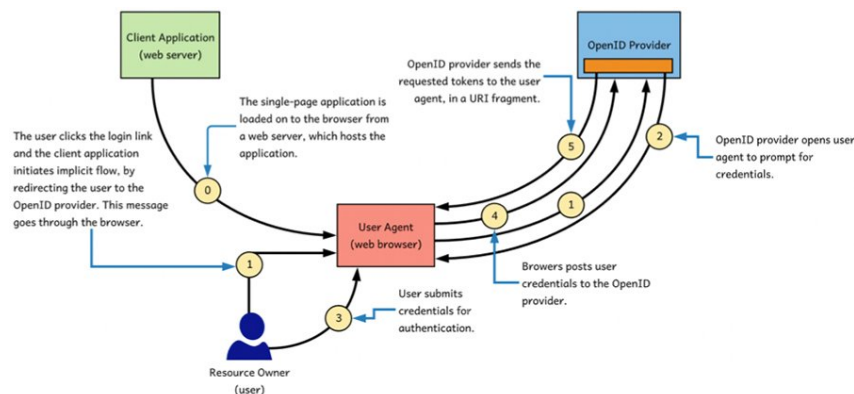
Thứ nhất, yêu cầu xác thực. Khi người dùng muốn đăng nhập vào client, ứng dụng sẽ chuyển hướng người dùng đến authorization server (thường là một OpenID Provider, ví dụ: Google, Facebook). Yêu cầu này sẽ chứa thông tin về client, phạm vi quyền truy cập yêu cầu và một số thông tin khác như chế độ xác thực mong muốn.

Thứ hai, xác thực người dùng. Authorization server sẽ kiểm tra danh tính của người dùng, thông qua việc đăng nhập bằng tài khoản hiện có hoặc xác minh thông qua một phương thức khác. Sau khi xác thực thành công, người dùng sẽ cung cấp sự đồng ý cho việc chia sẻ thông tin cá nhân và quyền truy cập tài nguyên với client.

Thứ ba, trả về mã xác thực và ID token. Sau khi người dùng đồng ý, authorization server sẽ trả về một mã xác thực (authorization code) và ID token (dưới dạng JWT) cho client. Mã xác thực sẽ được sử dụng để đổi lấy access token, trong khi ID token chứa thông tin về danh tính của người dùng.

Thứ tư, đổi mã xác thực thành access token và lấy thông tin người dùng. Client sẽ gửi mã xác thực đến authorization server để đổi lấy access token. Sau đó, client có thể sử dụng access token này để truy cập tài nguyên của người dùng từ resource server (resource server). Ứng dụng cũng có thể giải mã ID token để lấy thông tin người dùng, như định danh, tên, địa chỉ email, ảnh đại diện, và các thông tin khác theo yêu cầu.

Thứ năm, duy trì phiên đăng nhập và làm mới token. Bởi vì access token có thời hạn sử dụng, do đó, client có thể yêu cầu một refresh token từ authorization server để làm mới access token mà không cần người dùng phải đăng nhập lại. Refresh token thường có thời hạn dài hơn access token, cho phép ứng dụng duy trì phiên đăng nhập của người dùng trong thời gian dài hơn.



Hình 3: Mô tả tổng quan nguyên lý hoạt động của OIDC

c/ Cách cài đặt

Để cài đặt giao thức OIDC, nhóm nghiên cứu nhận thấy cần phải thực hiện 05 bước.

Thứ nhất, chuẩn bị môi trường. Để cài đặt, ta cần chuẩn bị hệ thống xác thực người dùng IdP, ứng dụng (có thể là web, mobile,...) và một số thư viện OIDC.

Thứ hai, cấu hình ứng dụng cho client. Việc cấu hình này bao gồm thiết lập các thông tin cần thiết để ứng dụng của người dùng giao tiếp với máy chủ OIDC, bao gồm mã định danh (Client ID), khóa private (Client Key) cho ứng dụng do máy chủ cung cấp.

Thứ ba, triển khai quy trình OIDC. Đầu tiên, máy chủ chuyển hướng người dùng đến trang yêu cầu

xác thực cùng với các tham số như

- `client_id` : Mã định danh của ứng dụng
- `client_key` : Khoá private của ứng dụng
- `redirect_url` : URL callback của ứng dụng

Sau khi người dùng đăng nhập và chấp nhận yêu cầu, máy chủ OIDC sẽ chuyển hướng đến URL callback `redirect_url` kèm theo mã xác thực (Authorization Code) và mã chống giả mạo CSRF. Đồng thời, máy chủ tiến hành trao đổi mã xác thực để lấy Access Token và ID Token dùng để truy cập và thu thập thông tin người dùng dưới dạng JWT.

Thứ tư, tích hợp trong ứng dụng. Máy chủ sẽ sử dụng thông tin từ token thu được để đăng nhập hoặc tạo tài khoản cho người dùng trong hệ thống. Đồng thời, máy chủ cũng lưu thông tin đăng nhập vào session/cookie và kiểm tra quyền hạn xem người dùng có quyền truy cập tài nguyên hay không. Ngoài ra, máy chủ cũng cung cấp Refresh Token để yêu cầu token mới mà không cần đăng nhập lại.

Thứ năm, xử lý bảo mật. Máy chủ phải luôn bảo vệ `client_key` và không chia sẻ nó trong mã nguồn công khai. Ngoài ra, hệ thống cần sử dụng giao thức HTTPS để đảm bảo an toàn khi truyền dữ liệu, cũng như xác thực tính hợp lệ của Token và duy trì phiên làm việc

d/ Đánh giá

OpenID Connect là một trong những giao thức Single Sign-On (SSO) hiện đại, được sử dụng rộng rãi nhờ tính linh hoạt và tích hợp tốt với nhiều ứng dụng.

Xét về điểm mạnh, OIDC mang tính hiện đại và phổ biến, nó cho phép xây dựng trên giao thức OAuth 2.0, mang lại khả năng xác thực hiện đại, bảo mật và hỗ trợ tốt cho các ứng dụng web, ứng dụng di động, và API.

Ngoài ra, OIDC cũng đơn giản hoá việc tích hợp, bằng cách sử dụng JSON Web Tokens (JWT), có nhiều thư viện và SDK hỗ trợ (như Python, Java, Node.js,...), giúp việc triển khai dễ dàng hơn.

OIDC cũng hỗ trợ đa tính năng, như hỗ trợ xác thực dựa trên Token, cho phép tùy chỉnh các quyền truy cập thông qua Scopes và Claims. Nó cũng có khả năng mở rộng, cho phép tích hợp dễ dàng với các hệ thống hiện đại như Kubernetes, API Gateway, và các dịch vụ đám mây.

Không chỉ vậy, OIDC còn có hỗ trợ các cơ chế bảo mật mạnh mẽ và dễ dàng tích hợp với các phương thức xác thực đa yếu tố (MFA).

Xét về điểm yếu, OIDC yêu cầu phức tạp trong việc xây dựng IdP, đồng thời đòi hỏi kiến thức về OAuth2 cũng như các khái niệm liên quan.

Ngoài ra, OIDC khá phụ thuộc vào HTTPS. Giao thức bắt buộc phải sử dụng HTTPS để đảm bảo an toàn dữ liệu, điều này có thể gây khó khăn trong môi trường phát triển hoặc triển khai nội bộ.

OIDC cũng không tương thích với các hệ thống không hỗ trợ giao thức OAuth 2.0 hoặc không thể tích hợp với JWT.

Đặc biệt, OIDC cũng gặp khó khăn trong việc bảo mật Token. Các Token cần được lưu trữ an toàn, nếu bị đánh cắp sẽ dẫn đến các cuộc tấn công giả mạo, gây rắc rối cho hệ thống và người dùng.

2.2.2 Central Authentication Service

a/ Mô tả

Central Authentication Service (CAS) là một giao thức đăng nhập một lần (SSO) cho web được phát triển bởi đại học Yale. Mục đích của nó là cho phép người dùng truy cập nhiều ứng dụng trong khi chỉ cần cung cấp thông tin của họ chỉ một lần, ví dụ như username và password. Nó cũng cho phép các ứng dụng web xác thực người sử dụng mà không cần tiếp cận với các thông tin bảo mật người dùng, chẳng hạn như mật khẩu.

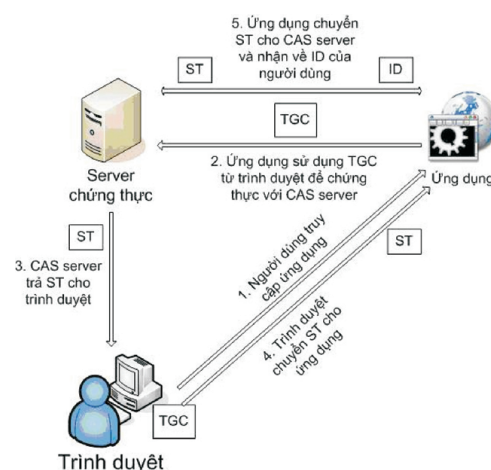
Giao thức CAS bao gồm ít nhất ba bên, bao gồm một trình duyệt của client, các ứng dụng web yêu cầu chứng thực và các máy chủ CAS. Nó cũng có thể liên quan đến một dịch vụ back-end, chẳng hạn như một máy chủ cơ sở dữ liệu, nó không có giao diện HTTPS riêng của mình nhưng giao tiếp với một ứng dụng web.

Khi client truy cập vào một ứng dụng mong muốn để xác thực với nó, ứng dụng chuyển hướng nó đến CAS xác nhận tính xác thực của client, thường là bằng cách kiểm tra tên người dùng và mật khẩu đối với một cơ sở dữ liệu chẳng hạn như MySQL. Nếu xác thực thành công, CAS trả client về ứng dụng trước đó thông qua một server ticket (ST). Ứng dụng này sau đó xác nhận ticket bằng cách liên hệ CAS trên một kết nối an toàn và cung cấp dịch vụ nhận dạng riêng của mình và ticket.

b/ Nguyên lý hoạt động

Đối với trường hợp người dùng truy cập vào ứng dụng khi đã chứng thực với CAS, đầu tiên người dùng truy xuất ứng dụng thông qua trình duyệt. Sau đó, ứng dụng lấy TGC (Ticket Granting Cookie, là cookies của HTTPS đặt bởi CAS trên sự khởi tạo phiên làm việc của cơ chế SSO) từ trình duyệt và chuyển nó cho CAS thông qua giao thức HTTPS. Nếu TGC này là hợp lệ, CAS trả về một Service Ticket (ST) cho trình duyệt và trình duyệt truyền ST vừa nhận được cho ứng dụng.

Kế tiếp, ứng dụng sử dụng ST nhận được từ trình duyệt và sau đó chuyển nó cho CAS. CAS trả về ID của người dùng cho ứng dụng, mục đích là để thông báo với ứng dụng là người dùng này đã được chứng thực bởi CAS. Cuối cùng, ứng dụng đăng nhập cho người dùng và bắt đầu phục vụ người dùng. Hình ảnh tổng quan cho trường hợp này được biểu diễn qua Hình 4.

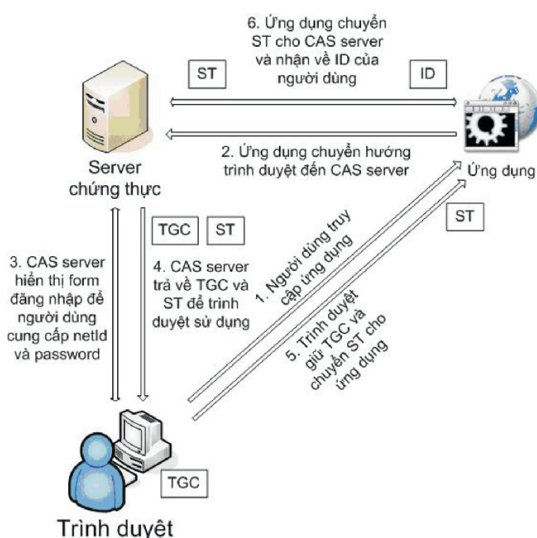


Hình 4: Người dùng truy cập vào ứng dụng khi đã chứng thực với CAS

Đối với trường hợp người dùng truy cập vào ứng dụng mà chưa chứng thực với CAS, khi người dùng truy xuất ứng dụng thông qua trình duyệt, bởi vì chưa nhận được TGC nên ứng dụng sẽ chuyển hướng người dùng cho CAS.

Người dùng cần phải cung cấp tên truy cập và mật khẩu của mình thông qua khung đăng nhập để CAS xác thực. Thông tin thu được sẽ truyền đi thông qua giao thức HTTPS. Sau khi xác thực thành công, CAS sẽ trả về cho trình duyệt đồng thời cả TGC và ST.

Ngoài ra, trình duyệt sẽ giữ lại TGC để sử dụng cho các ứng dụng khác (nếu có) và truyền ST cho ứng dụng nêu trên. Đồng thời, ứng dụng chuyển ST cho CAS và nhận về ID của người dùng. Cuối cùng, ứng dụng đăng nhập cho người dùng và bắt đầu phục vụ người dùng. Hình ảnh tổng quan cho trường hợp này được biểu diễn qua Hình 5.



Hình 5: Người dùng truy cập vào ứng dụng khi chưa chứng thực với CAS

c/ Cách cài đặt

Đầu tiên, CAS cần hoạt động trên môi trường Java. Do đó, để cài đặt CAS, ta cần phải cài đặt môi trường Java trước.

Kế tiếp, ta cần phải triển khai CAS Server. CAS Server có thể được triển khai dưới dạng một ứng dụng WAR. Ta cần phải tải tệp WAR của CAS Server và triển khai nó trên một máy chủ ứng dụng.

Sau khi cài đặt, ta cần phải cấu hình cho CAS Server để nó hoạt động theo nhu cầu của người dùng bằng cách xác định các tham số như tên miền và URL của CAS Server hoặc cấu hình hệ thống xác thực.

Ngoài ra, ta cũng cần phải cài đặt máy chủ trước khi triển khai CAS Server. CAS Server cần được triển khai trên một máy chủ ứng dụng như Apache Tomcat, Jetty hoặc bất kỳ máy chủ Java EE nào.

Khi cấu hình và triển khai CAS Server xong, ta cần kiểm tra bằng cách truy cập vào URL của CAS Server. Nếu đúng, màn hình sẽ chuyển sang giao diện đăng nhập của CAS.

Bên cạnh CAS Server, ta cũng cần phải cài đặt và cấu hình cho CAS Client. Các ứng dụng cần kết nối với CAS Server sẽ hoạt động như các client. Ta có thể thực hiện một số thay đổi cấu hình của ứng dụng hoặc tích hợp thêm nhiều tính năng cho client, có thể bao gồm tích hợp các phương thức xác thực khác nhau, cài đặt các chính sách bảo mật và tùy chỉnh giao diện đăng nhập.

Cuối cùng, khi tất cả các cấu hình hoàn tất, ta nên tiến hành kiểm tra tính năng đăng nhập và giám sát hệ thống để đảm bảo rằng CAS đang hoạt động ổn định và bảo mật.

d/ Đánh giá

Xét về điểm mạnh, CAS cung cấp một giải pháp đăng nhập đơn giản và hiệu quả, giúp người dùng chỉ cần đăng nhập một lần để truy cập vào nhiều ứng dụng khác nhau. Điều này làm tăng trải nghiệm người dùng và giảm thiểu việc quản lý tài khoản và mật khẩu cho từng ứng dụng.

CAS cũng sử dụng các giao thức bảo mật mạnh mẽ như HTTPS để đảm bảo rằng thông tin đăng nhập của người dùng được mã hóa khi truyền tải qua mạng. Đồng thời, CAS còn hỗ trợ xác thực đa yếu tố và các phương thức xác thực khác giúp tăng cường bảo mật.

Ngoài ra, CAS có thể dễ dàng tích hợp vào các hệ thống hiện có, đặc biệt là các ứng dụng web, và hỗ trợ nhiều nền tảng khác nhau. Hoặc hỗ trợ tích hợp với nhiều ứng dụng khác nhau từ các ứng dụng Java (Spring Security) đến các hệ thống bên ngoài, hỗ trợ nhiều giao thức như SAML, OAuth, OpenID để dễ dàng tích hợp với các hệ thống xác thực bên ngoài.

CAS còn cho phép quản lý tất cả các tài khoản người dùng và chính sách bảo mật từ một điểm trung tâm, giúp giảm chi phí và thời gian quản lý người dùng trong môi trường có nhiều ứng dụng.

Xét về điểm yếu, CAS rất phụ thuộc vào CAS Server. Nếu CAS Server gặp sự cố, tất cả các ứng dụng sử dụng CAS sẽ không thể xác thực người dùng. Điều này tạo ra điểm yếu duy nhất trong hệ thống khi CAS Server bị lỗi hoặc ngừng hoạt động.

Bên cạnh đó, việc triển khai và cấu hình CAS Server có thể phức tạp, đặc biệt là khi liên quan đến việc cấu hình bảo mật, kết nối với cơ sở dữ liệu hoặc dịch vụ xác thực LDAP. Các tổ chức không có đủ kinh nghiệm có thể gặp khó khăn trong việc thiết lập và bảo trì hệ thống.

CAS cũng gặp khó khăn trong vấn đề tương thích với một số ứng dụng. Một số ứng dụng cũ hoặc ứng dụng không được thiết kế để sử dụng CAS có thể gặp khó khăn khi tích hợp với hệ thống này. Việc tùy chỉnh hoặc phát triển các giải pháp thay thế sẽ làm tốn thời gian và chi phí rất nhiều.

Mặc dù CAS cung cấp bảo mật mạnh mẽ, nhưng nếu không được cấu hình đúng, nó có thể trở thành mục tiêu tấn công. Các lỗ hổng bảo mật như tấn công giả mạo hoặc khai thác điểm yếu trong xác thực có thể xảy ra nếu không duy trì các biện pháp bảo mật chặt chẽ.

2.2.3 Security Assertion Markup Language

a/ Mô tả

SAML (Security Assertion Markup Language) là một tiêu chuẩn XML dựa trên giao thức được phát triển bởi OASIS (Organization for the Advancement of Structured Information Standards) để trao đổi thông tin xác thực và ủy quyền giữa các bên. SAML giúp đơn giản hóa quá trình đăng nhập cho người dùng khi họ truy cập vào nhiều ứng dụng và dịch vụ web khác nhau.

SAML cho phép người dùng đăng nhập một lần và sử dụng quyền truy cập đó để truy cập vào nhiều ứng dụng khác nhau mà không cần phải đăng nhập lại. Nó giúp giảm bớt khó khăn và thời gian đăng nhập cho người dùng, đồng thời tăng cường bảo mật và giảm thiểu rủi ro liên quan đến việc quản lý nhiều tài khoản và mật khẩu.

Cấu trúc chính của SAML bao gồm hai thành phần, là Identity Provider (IdP) và Service Provider (SP). Identity Provider là nơi xác thực danh tính của người dùng và cung cấp thông tin xác thực cho Service Provider. Service Provider là ứng dụng hoặc dịch vụ mà người dùng muốn truy cập, yêu cầu xác thực từ Identity Provider để đảm bảo người dùng có quyền truy cập hợp lệ.

b/ Nguyên lý hoạt động

SAML hoạt động dựa trên việc trao đổi các thông điệp XML giữa Identity Provider (IdP) và Service Provider (SP) để xác thực người dùng và truyền dữ liệu ủy quyền. Cách thức hoạt động của SAML bao gồm 05 bước.

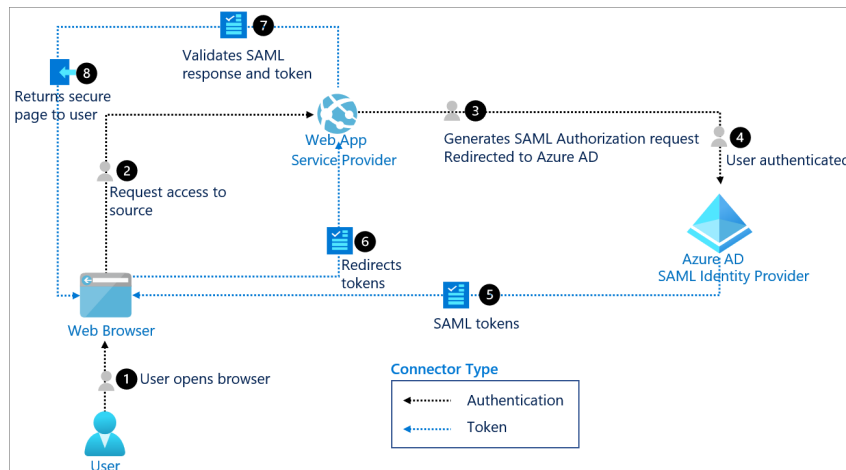
Thứ nhất, yêu cầu truy cập. Khi người dùng truy cập vào một ứng dụng hoặc dịch vụ (SP), họ sẽ được chuyển hướng đến IdP để xác thực danh tính.

Thứ hai, xác thực danh tính. IdP sẽ kiểm tra danh tính của người dùng thông qua tài khoản và mật khẩu, xác nhận hai yếu tố hoặc các phương thức xác thực khác. Nếu xác thực thành công, IdP sẽ tạo ra một thông điệp SAML Assertion chứa thông tin xác thực và ủy quyền của người dùng.

Thứ ba, truyền thông điệp SAML Assertion. IdP sẽ gửi thông điệp SAML Assertion đến SP thông qua trình duyệt của người dùng. Thông điệp này sẽ được ký bằng chữ ký số của IdP để đảm bảo tính bảo mật và toàn vẹn của dữ liệu.

Thứ tư, xử lý SAML Assertion. SP sẽ kiểm tra chữ ký số của thông điệp SAML Assertion để đảm bảo rằng nó đến từ IdP đáng tin cậy. Sau đó, SP sẽ trích xuất thông tin xác thực và ủy quyền từ SAML Assertion để cấp quyền truy cập cho người dùng.

Thứ năm, cấp quyền truy cập. Người dùng sẽ được cấp quyền truy cập vào ứng dụng hoặc dịch vụ (SP) dựa trên thông tin xác thực và ủy quyền từ SAML Assertion.



Hình 6: Mô tả luồng xác thực người dùng của SAML

c/ Cách cài đặt

Đầu tiên, ta cần cài đặt các phần mềm hỗ trợ SAML và thư viện SAMP cho SP khi triển khai IdP, SP cho hệ thống của mình.

Kế tiếp, ta cần triển khai việc cài đặt IdP bằng cách cài đặt Shibboleth IdP và thiết lập SSL và

certificate cho IdP để bảo mật giao tiếp.

Đối với Keycloak, sau khi cài đặt, ta thiết lập SAML Identity Provider và tạo một Service Provider trong Keycloak.

Sau khi cài đặt IdP, ta cũng cần triển khai SP. Ta cần đăng ký ứng dụng SP với IdP và cung cấp các thông tin, sau đó cấu hình IdP để chấp nhận yêu cầu SAML từ SP để tạo ra metadata cho SP.

Để cấu hình SAML metadata, đối với metadata của IdP, IdP sẽ cung cấp một tệp metadata (hoặc URL) mà SP có thể tải về để cấu hình. Còn đối với metadata của SP, SP cũng sẽ tạo ra metadata cho IdP để nhận diện và cấu hình kết nối. Tệp này chứa các thông tin như URL của IdP, certificate và các tham số yêu cầu.

Sau khi cài đặt và cấu hình xong, ta cần kiểm tra quy trình từ SP đến IdP. Đảm bảo rằng người dùng có thể đăng nhập qua IdP và dịch vụ SP nhận diện thông tin người dùng từ SAML assertions.

Cuối cùng, ta cần đảm bảo rằng các kết nối SAML đều được mã hóa và sử dụng chứng chỉ SSL để bảo mật thông tin truyền tải. Đồng thời, ta cũng cần cấu hình các quyền truy cập, nhóm người dùng và các thông tin trong SAML assertions để kiểm soát quyền truy cập cho từng dịch vụ.

d/ Đánh giá

Xét về điểm mạnh, SAML sử dụng mã hóa mạnh mẽ để bảo vệ dữ liệu và thông tin nhạy cảm, như tên người dùng và thông tin phân quyền, trong các thông điệp SAML. Nó cũng sử dụng chứng chỉ SSL/TLS để bảo mật việc truyền tải dữ liệu giữa IdP và SP, đồng thời cung cấp các cơ chế bảo vệ chống lại các cuộc tấn công.

SAML còn là một chuẩn mở và hỗ trợ rộng rãi trong nhiều hệ thống và công cụ nhận dạng và cho phép quản lý danh sách người dùng và phân quyền từ một điểm duy nhất, giúp đơn giản hóa quá trình quản lý người dùng và giảm thiểu sai sót.

SAML thường được sử dụng trong các môi trường doanh nghiệp lớn, nơi có nhiều ứng dụng và dịch vụ cần một cơ chế xác thực tập trung.

Xét về điểm yếu, SAML có thể yêu cầu cấu hình chi tiết và phức tạp, đặc biệt là khi triển khai cho các tổ chức lớn hoặc khi tích hợp nhiều ứng dụng với các nhà cung cấp IdP khác nhau. Việc cấu hình metadata và đảm bảo tính chính xác trong các tệp cấu hình có thể là một thử thách.

SAML có thể gặp khó khăn trong việc mở rộng với các dịch vụ web có nhu cầu truy cập cao vì nó yêu cầu giao tiếp giữa IdP và SP qua các xác nhận và các thông điệp SAML. Đặc biệt, việc mã hóa và giải mã các thông điệp có thể làm tăng độ trễ trong quá trình xác thực người dùng.

Không chỉ vậy, SAML còn yêu cầu tổ chức phải duy trì và quản lý một IdP đáng tin cậy. Nếu IdP gặp sự cố, người dùng sẽ không thể đăng nhập vào các ứng dụng SP. Điều này yêu cầu có các giải pháp dự phòng và kiểm tra thường xuyên để đảm bảo tính sẵn sàng.

Mặc dù SAML rất mạnh mẽ trong môi trường doanh nghiệp và với các ứng dụng lớn, tuy nhiên nó lại ít phổ biến đối với các ứng dụng di động hoặc các dịch vụ đám mây quy mô nhỏ. Các giao thức khác như OAuth2 và OpenID Connect đang dần thay thế SAML trong những trường hợp này.

2.2.4 So sánh các giao thức

Mỗi giao thức có những đặc điểm và ưu, nhược điểm riêng phù hợp với các nhu cầu và môi trường khác nhau. Việc lựa chọn giao thức phù hợp sẽ tùy thuộc vào các yếu tố như quy mô tổ chức, loại ứng dụng cần bảo vệ và các yêu cầu về bảo mật. Bảng 1 mô tả sự khác biệt giữa các giao thức dựa trên các tiêu chí quan trọng như tính bảo mật, tính tương thích, tính linh hoạt, triển khai, khả năng mở rộng và chi phí triển khai.

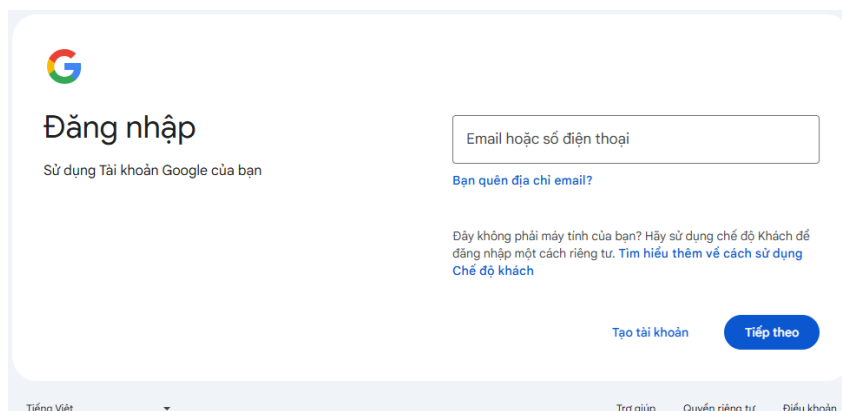
Tiêu chí	OpenID Connect	CAS (Central Authentication Service)	SAML (Security Assertion Markup Language)
Tính bảo mật	Dựa trên OAuth2, sử dụng Token JWT, PKCE và mã hóa mạnh	Cung cấp SSO cơ bản, bảo mật không mạnh bằng OIDC và SAML	Mã hóa dữ liệu mạnh mẽ, dùng XML và chữ ký số bảo vệ thông tin
Tính tương thích	Tương thích với các hệ thống hiện đại, dịch vụ đám mây	Tương thích tốt với ứng dụng web nội bộ	Tương thích với ứng dụng doanh nghiệp, ít tương thích với di động
Tính linh hoạt	Rất linh hoạt, phù hợp cho ứng dụng di động, web, đám mây	Linh hoạt trong môi trường nội bộ nhưng ít hỗ trợ phức tạp	Linh hoạt trong doanh nghiệp, nhưng khó dùng cho ứng dụng hiện đại
Triển khai	Dễ triển khai với API rõ ràng, dễ tích hợp	Triển khai đơn giản cho nội bộ nhưng không dễ mở rộng	Phức tạp, yêu cầu cấu hình và quản lý chứng chỉ chi tiết
Khả năng mở rộng	Linh hoạt, dễ tích hợp dịch vụ đám mây và ứng dụng hiện đại	Mở rộng tốt trong môi trường web nội bộ, nhưng hạn chế với phức tạp	Mở rộng mạnh trong tổ chức lớn, khó tích hợp với dịch vụ mới
Chi phí triển khai	Chi phí thấp nhờ sự hỗ trợ rộng rãi và dễ tích hợp	Chi phí thấp cho SSO đơn giản, nhưng không hiệu quả khi mở rộng	Chi phí cao vì yêu cầu cấu hình và quản lý chứng chỉ phức tạp

Bảng 1: So sánh giữa các giao thức dựa trên nhiều tiêu chí khác nhau

2.3 Một số mô hình triển khai Single Sign-On

Với sự phát triển mạnh mẽ của công nghệ và nhu cầu bảo mật thông tin, nhiều mô hình triển khai SSO đã ra đời để đáp ứng các yêu cầu khác nhau của doanh nghiệp. Mỗi mô hình mang lại những lợi ích riêng, từ việc đơn giản hóa trải nghiệm người dùng đến tối ưu hóa quản lý và bảo vệ dữ liệu.

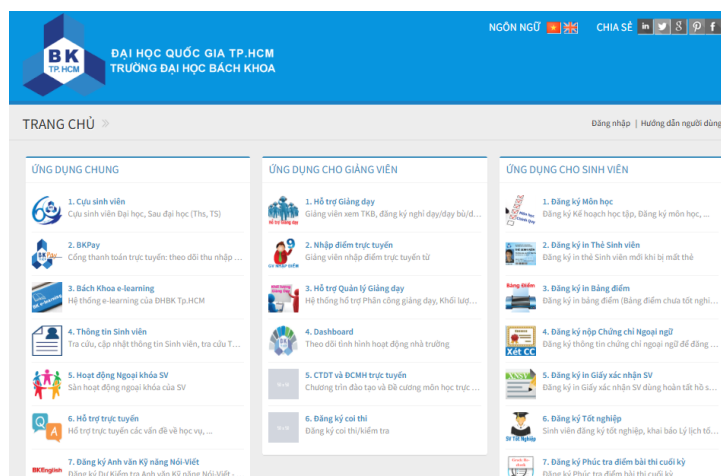
Tại các doanh nghiệp lớn như Google, người dùng chỉ cần đăng nhập một lần bằng email hoặc số điện thoại và mật khẩu để truy cập toàn bộ dịch vụ như Gmail, Google Drive, YouTube. Google sử dụng các giao thức bảo mật như OAuth 2.0 và OpenID Connect (OIDC), cùng với xác thực hai yếu tố (2FA) để bảo vệ thông tin người dùng. Hệ thống SSO này giúp tiết kiệm thời gian và loại bỏ việc phải nhớ nhiều tài khoản, mật khẩu.



The image shows the Google login page. At the top left is the Google logo. Below it, the text "Đăng nhập" (Login) is displayed, followed by "Sử dụng Tài khoản Google của bạn" (Use your Google Account). To the right, there is a text input field labeled "Email hoặc số điện thoại" (Email or phone number). Below this field is a link "Bạn quên địa chỉ email?" (Forgot your email address?). Further down, a message states: "Đây không phải là máy tính của bạn? Hãy sử dụng chế độ Khách để đăng nhập một cách riêng tư. [Tìm hiểu thêm về cách sử dụng Chế độ khách](#)" (This isn't your computer? Use Guest mode to log in privately. [Learn more about Guest mode](#)). At the bottom right, there are two buttons: "Tạo tài khoản" (Create account) and "Tiếp theo" (Next). At the very bottom, there are links for "Tiếng Việt" (Vietnamese), "Trợ giúp" (Help), "Quyền riêng tư" (Privacy), and "Điều khoản" (Terms).

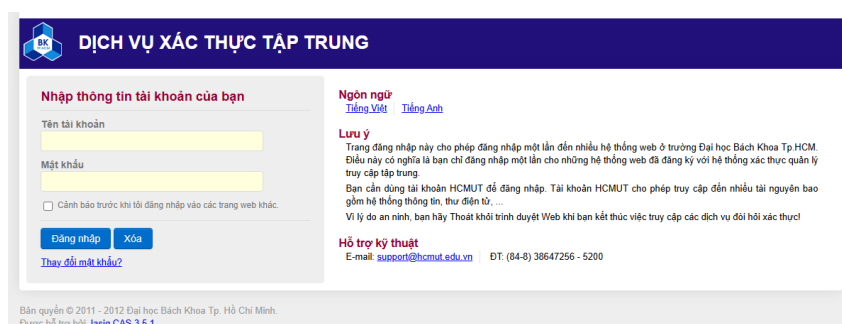
Hình 7: Trang đăng nhập của Google

Ngoài ra, tại các trường học cũng áp dụng nhiều mô hình triển khai SSO để cung cấp cho sinh viên, giảng viên quyền truy cập vào các hệ thống quản lý học tập (LMS), email và thư viện số. Cụ thể, tại Trường Đại học Bách khoa - Đại học Quốc gia Thành phố Hồ Chí Minh (HCMUT - VNUHCM). SSO đã được triển khai bằng cách sử dụng CAS nhằm tối ưu hóa việc truy cập vào các hệ thống số hóa phục vụ quản lý và học tập. Một số ứng dụng tại Trường có thể kể đến bao gồm **Thông tin sinh viên**, **Đăng ký môn học**, **Bách khoa e-learning** hay một số ứng dụng đặc thù dành riêng cho sinh viên.



The image shows the MyBK system dashboard. At the top, there is a blue header with the BK logo and the text "ĐẠI HỌC QUỐC GIA TP.HCM TRƯỜNG ĐẠI HỌC BÁCH KHOA". To the right of the header are links for "NGÔN NGỮ" (Language) and "CHIA SẺ" (Share), along with social media icons. Below the header, the main content area is titled "TRANG CHỦ" (Home) and "Đăng nhập | Hướng dẫn người dùng" (Login | User guide). The dashboard is divided into three columns: "ỨNG DỤNG CHUNG" (Common applications), "ỨNG DỤNG CHO GIẢNG VIÊN" (Applications for lecturers), and "ỨNG DỤNG CHO SINH VIÊN" (Applications for students). Each column lists various services with icons and brief descriptions. For example, under "ỨNG DỤNG CHUNG", there are links for "Cựu sinh viên" (Former students), "BKPay", "Bách Khoa e-learning", "Thông tin Sinh viên" (Student information), "Hoạt động Ngoại khóa SV" (Student extracurricular activities), "Hỗ trợ trực tuyến" (Online support), and "Đăng ký Anh văn Kỹ năng Nói-Viết" (English registration). Under "ỨNG DỤNG CHO GIẢNG VIÊN", there are links for "Hỗ trợ Giảng dạy" (Teaching support), "Nhập điểm trực tuyến" (Online grade entry), "Hỗ trợ Quản lý Giảng dạy" (Teaching management support), "Dashboard", "CTĐT và ĐCMH trực tuyến" (Online degree and course management), and "Đăng ký coi thi" (Exam registration). Under "ỨNG DỤNG CHO SINH VIÊN", there are links for "Đăng ký Môn học" (Course registration), "Đăng ký in Thẻ Sinh viên" (Student ID card registration), "Đăng ký in Bằng điểm" (Degree registration), "Đăng ký nộp Chứng chỉ Ngoại ngữ" (Foreign language certificate registration), "Đăng ký in Giấy xác nhận SV" (Student confirmation registration), "Đăng ký Tốt nghiệp" (Graduation registration), and "Đăng ký Phúc tra điểm bài thi cuối kỳ" (Final exam score review registration).

Hình 8: Hệ thống MyBK với nhiều ứng dụng dành cho sinh viên, giảng viên



The image shows the CAS login page for HCMUT. The header is dark blue with the BK logo and the text "DỊCH VỤ XÁC THỰC TẬP TRUNG" (Central Authentication Service). Below the header, there are two main sections. The left section is titled "Nhập thông tin tài khoản của bạn" (Enter your account information) and contains fields for "Tên tài khoản" (Username) and "Mật khẩu" (Password). Below these fields are checkboxes for "Cảnh báo trước khi tôi đăng nhập vào các trang web khác" (Warn me before I log in to other websites) and buttons for "Đăng nhập" (Login) and "Xóa" (Clear). There is also a link "Thay đổi mật khẩu?" (Change password?). The right section is titled "Ngôn ngữ" (Language) with links for "Tiếng Việt" (Vietnamese) and "Tiếng Anh" (English). Below this is a "Lưu ý" (Note) section with text: "Trang đăng nhập này cho phép đăng nhập một lần đến nhiều hệ thống web ở trường Đại học Bách Khoa Tp.HCM. Điều này có nghĩa là bạn chỉ đăng nhập một lần cho những hệ thống web đã đăng ký với hệ thống xác thực quản lý truy cập tập trung. Bạn cần dùng tài khoản HCMUT để đăng nhập. Tài khoản HCMUT cho phép truy cập đến nhiều tài nguyên bao gồm hệ thống thông tin, thư điện tử, ... Vì lý do an ninh, bạn hãy Thoát khỏi trình duyệt Web khi bạn kết thúc việc truy cập các dịch vụ đối nội xác thực!" (This login page allows you to log in once to multiple web systems at HCMUT. This means you only need to log in once for web systems registered with the central access management system. You need to use your HCMUT account to log in. The HCMUT account allows access to many resources including the information system, email, ... For security reasons, please log out of the web browser when you finish using the internal services!). At the bottom right, there is a "Hỗ trợ kỹ thuật" (Technical support) section with an email address "support@hcmut.edu.vn" and a phone number "ĐT: (84-8) 38647256 - 5200". At the very bottom, there is a footer with the text "Bản quyền © 2011 - 2012 Đại học Bách Khoa Tp. Hồ Chí Minh. Được hỗ trợ bởi [Jasig CAS 3.5.1](#)".

Hình 9: Trang đăng nhập với mô hình SSO sử dụng CAS

3 Phân tích và thiết kế

3.1 Phân tích yêu cầu

3.1.1 Yêu cầu hệ thống

Để thiết kế hệ thống đáp ứng đủ các tiêu chí và điều kiện cần thiết, việc xác định yêu cầu, phạm vi cũng như môi trường sao cho hệ thống có thể vận hành hiệu quả là vô cùng cần thiết.

Hệ thống cần phải đảm bảo dữ liệu người dùng được bảo vệ, tránh rủi ro từ các lỗ hổng bảo mật. Tất cả thông tin và dữ liệu truyền tải cần phải được mã hoá bằng các giao thức an toàn. Ngoài ra, các token cần có thời gian sống hợp lý, tránh bị lạm dụng. Hệ thống cũng cần đảm bảo khả năng làm việc trên nhiều nền tảng và ngôn ngữ lập trình khác nhau.

Quan trọng nhất, hệ thống phải mang đến trải nghiệm người dùng thông qua việc đăng nhập một lần để truy cập tất cả các ứng dụng liên quan.

3.1.2 Yêu cầu chức năng

Bên cạnh việc xác định yêu cầu của hệ thống, việc xác định hệ thống sẽ thực hiện những gì, phục vụ người dùng như thế nào và cách các chức năng liên kết với nhau để đáp ứng các yêu cầu đặt ra cũng quan trọng không kém.

Hệ thống cần cung cấp giao diện cho người dùng để tiến hành xác thực thông tin khi đăng nhập vào hệ thống. Đồng thời, hệ thống cũng cung cấp và quản lý token sau khi người dùng tiến hành đăng nhập thành công. Token cần phải chứa các thông tin như phân quyền, thời gian sống và thông tin của người dùng. Token cũng cần được hệ thống làm mới để tránh phải yêu cầu đăng nhập lại khi token hết hạn.

Ngoài ra, hệ thống cũng cần quản lý phiên làm việc của người dùng để duy trì trạng thái đăng nhập của người dùng trong toàn bộ hệ thống và đảm bảo các ứng dụng liên quan có thể kiểm tra trạng thái phiên làm việc của người dùng.

Không chỉ đăng nhập, khi người dùng tiến hành đăng xuất, hệ thống phải thông báo đến tất cả các ứng dụng liên kết để hủy bỏ các phiên hoạt động, đảm bảo trạng thái đăng xuất được đồng bộ trên toàn hệ thống.

3.2 Thiết kế hệ thống

3.2.1 Kiến trúc hệ thống

Trước khi tiến hành triển khai hệ thống, việc trình bày cách thức tổ chức và vận hành các thành phần trong hệ thống giúp hệ thống được thiết kế với chất lượng tốt nhất, mang đến trải nghiệm người dùng liền mạch và đảm bảo tính bảo mật.

Thành phần **User Access Request** giúp xác định và chuyển hướng người dùng đến IdP để bắt đầu quy trình xác thực. Cụ thể, khi người dùng truy cập vào hệ thống, nếu chưa đăng nhập thì hệ thống sẽ chuyển hướng người dùng đến IdP và tạo URL chuyển hướng với các thông tin cần thiết như `client_id`,

`redirect_uri`, `scope`, `state`. Nếu người dùng đã đăng nhập trước đó, hệ thống sẽ chuyển hướng người dùng đến tài nguyên ứng dụng và kết thúc quá trình.

Thành phần **Authentication** đảm bảo người dùng đã xác thực thành công trước khi tiếp tục luồng. Người dùng cần phải cung cấp thông tin xác thực như tên đăng nhập và mật khẩu và IdP sẽ tiến hành xác minh thông tin đó dựa trên cơ sở dữ liệu của người dùng. Nếu kiểm tra hợp lệ, IdP sẽ đánh dấu người dùng đã được xác thực, tạo Authorization Code, thiết lập thời gian sống cho mã và chuyển hướng người dùng trở lại ứng dụng với mã đã được tạo. Ngược lại, IdP gửi thông báo lỗi nếu thông tin xác thực không hợp lệ và kết thúc quá trình.

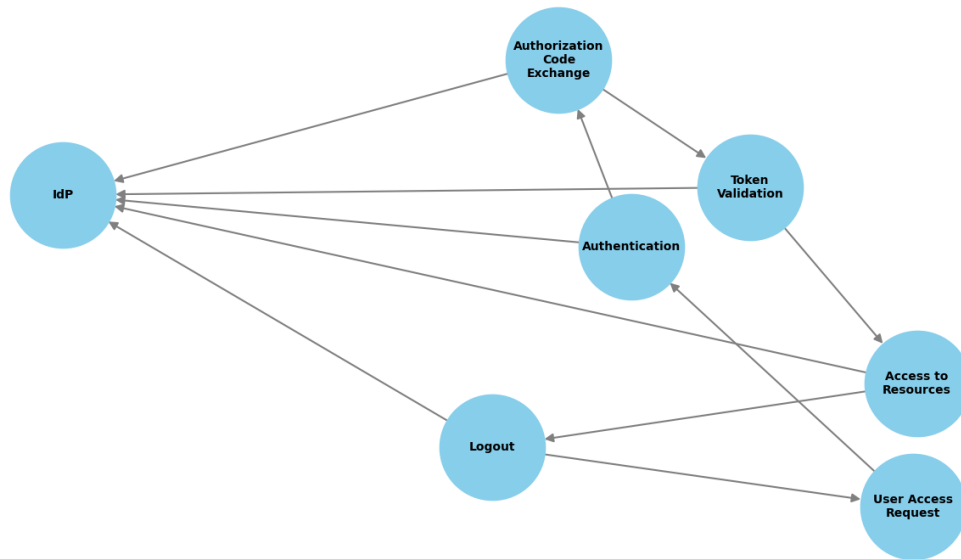
Thành phần **Authorization Code Exchange** cung cấp quyền truy cập an toàn thông qua token, tránh việc tiết lộ thông tin người dùng, đặc biệt sử dụng Refresh Token để duy trì phiên đăng nhập lâu dài. Đầu tiên, IdP tiến hành chuyển hướng trở lại ứng dụng với Authorization Code đã được tạo bằng cách tạo URL chuyển hướng và gửi phản hồi chuyển hướng qua giao thức HTTP đến trình duyệt của người dùng để trình duyệt điều hướng đến URL chuyển hướng của ứng dụng. Sau đó, ứng dụng tiến hành trao đổi Authorization Code bằng cách gửi yêu cầu đến endpoint của IdP với Authorization Code và các thông tin như `client_id`, `redirect_uri`. Nếu IdP kiểm tra thông tin hợp lệ, Access Token và Refresh Token sẽ được tạo và chuyển đến ứng dụng. Access Token được sử dụng để truy cập tài nguyên trong thời gian ngắn, trong khi Refresh Token cho phép ứng dụng yêu cầu Access Token mới khi token hiện tại hết hạn mà không yêu cầu người dùng đăng nhập lại.

Thành phần **Token Validation** giúp đảm bảo rằng các token được cấp là hợp lệ, an toàn và có thể sử dụng để truy cập tài nguyên hoặc thông tin người dùng. Cụ thể, ứng dụng sẽ tiến hành kiểm tra ID Token được tạo bởi IdP bằng cách kiểm tra chữ ký, thông tin nhà phát hành, đối tượng sử dụng và thời gian hết hạn của token. Nếu Access Token hết hạn, ứng dụng sử dụng Refresh Token để yêu cầu một Access Token mới từ IdP mà không cần người dùng xác thực lại, giúp duy trì trải nghiệm liền mạch và bảo mật.

Thành phần **Access to Resources** cung cấp trải nghiệm liền mạch cho người dùng, cho phép chuyển đổi giữa các ứng dụng mà không cần thực hiện nhiều lần đăng nhập. Sau khi người dùng đã xác thực, người dùng có thể truy cập vào các tài nguyên được bảo vệ thông qua việc sử dụng Access Token để gọi API. Khi Access Token hết hạn, hệ thống tự động sử dụng Refresh Token để yêu cầu token mới, giúp duy trì phiên làm việc của người dùng mà không cần thực hiện lại đăng nhập.

Thành phần **Log Out** đảm bảo rằng người dùng được thoát ra hoàn toàn khỏi hệ thống, tránh các lỗ hổng bảo mật do phiên làm việc còn sót lại. Khi người dùng nhấn nút đăng xuất khỏi ứng dụng, ứng dụng này sẽ gửi thông báo đến IdP cùng với các thông tin liên quan. IdP sẽ tiến hành xử lý việc xóa phiên làm việc (session) của người dùng trên toàn hệ thống. Đồng thời, IdP cũng thông báo tới các ứng dụng khác để đảm bảo trạng thái đăng xuất đồng bộ. Sau đó, người dùng sẽ nhận được thông báo đăng xuất để xác minh rằng người dùng đã đăng xuất khỏi hệ thống.

Tổng quan về hệ thống và mô tả sự tương tác giữa các thành phần được thể hiện qua Hình 10.

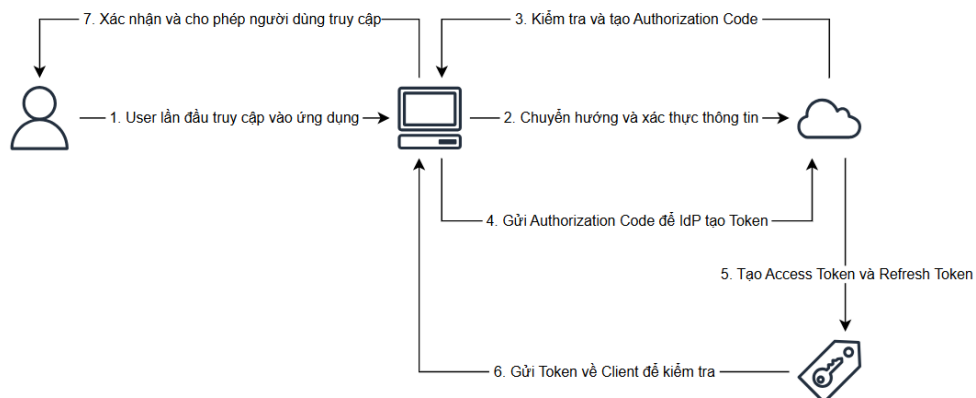


Hình 10: Tổng quan hệ thống và tương tác giữa các thành phần

3.2.2 Luồng hoạt động

Luồng hoạt động của hệ thống đóng vai trò quan trọng trong việc đảm bảo tính nhất quán và bảo mật trong quá trình tương tác của người dùng với hệ thống. Các luồng được thiết kế nhằm đáp ứng các tình huống khác nhau có thể xảy ra khi người dùng đăng nhập, đăng xuất, truy cập tài nguyên hoặc các trường hợp đặc biệt như quên mật khẩu.

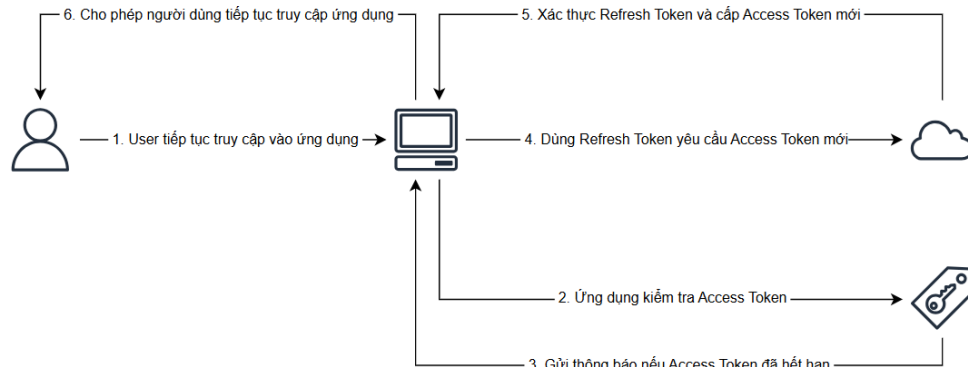
Khi người dùng lần đầu truy cập vào ứng dụng, hệ thống sẽ kiểm tra trạng thái đăng nhập của người dùng. Nếu chưa đăng nhập, người dùng sẽ được chuyển hướng đến IdP để thực hiện xác thực thông qua tên đăng nhập và mật khẩu. Sau khi xác thực thành công, IdP sẽ tạo và chuyển Authorization Code về ứng dụng. Từ đó, ứng dụng trao đổi mã này để nhận Access Token và Refresh Token, cho phép truy cập tài nguyên một cách an toàn. Luồng hoạt động này được thể hiện qua Hình 11.



Hình 11: Luồng hoạt động khi lần đầu truy cập ứng dụng

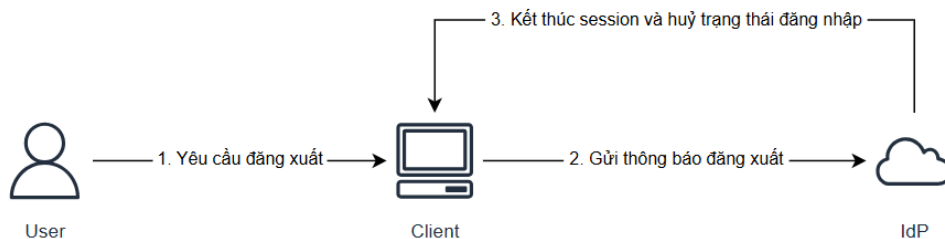
Khi người dùng tiếp tục truy cập vào một ứng dụng khác trong hệ thống, ứng dụng sẽ kiểm tra trạng thái của Access Token hiện tại. Nếu Access Token vẫn còn hiệu lực, người dùng được phép truy cập ngay vào ứng dụng. Trong trường hợp Access Token đã hết hạn, hệ thống sẽ gửi thông báo để khởi tạo luồng

làm mới token. Ứng dụng sẽ sử dụng Refresh Token đã lưu trữ để gửi yêu cầu cấp Access Token mới tới IdP. Sau khi IdP xác thực Refresh Token và đảm bảo tính hợp lệ, một Access Token mới sẽ được cấp. Với Access Token mới này, người dùng có thể tiếp tục sử dụng ứng dụng một cách liền mạch mà không cần đăng nhập lại. Luồng hoạt động này được thể hiện qua Hình 12.



Hình 12: Luồng hoạt động khi tiếp tục truy cập ứng dụng khác

Khi người dùng đăng xuất khỏi ứng dụng, hệ thống sẽ gửi yêu cầu đăng xuất đến IdP để đảm bảo hủy phiên làm việc trên toàn hệ thống. IdP sau đó thông báo tới tất cả các ứng dụng liên kết để đồng bộ trạng thái đăng xuất. Người dùng được xác nhận đăng xuất hoàn toàn, giúp đảm bảo bảo mật và tránh lạm dụng phiên đăng nhập còn sót lại. Luồng hoạt động này được thể hiện qua Hình 13.

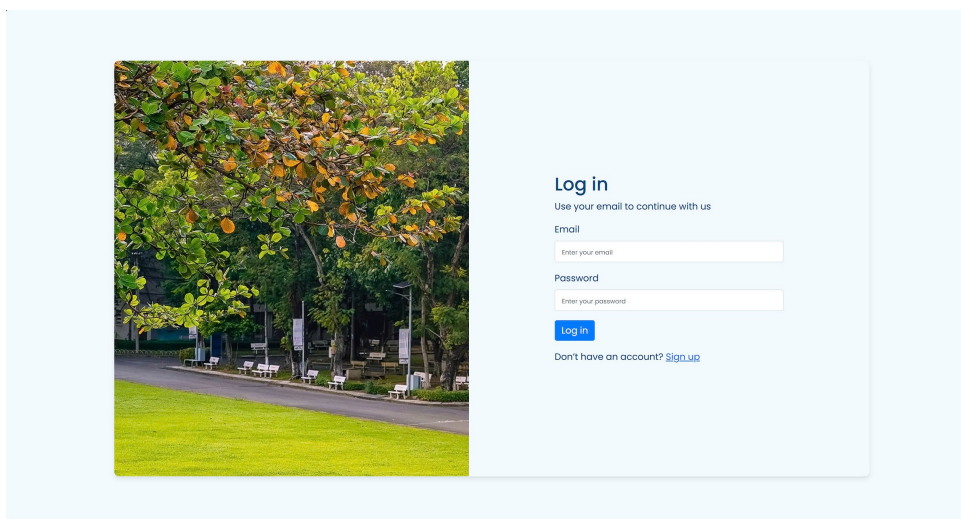


Hình 13: Luồng hoạt động khi đăng xuất khỏi ứng dụng

3.2.3 Giao diện người dùng

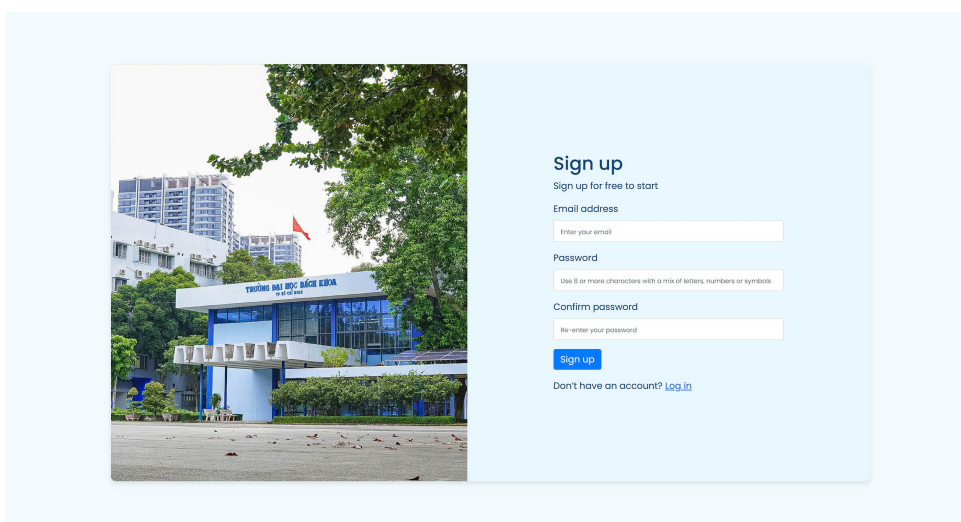
Giao diện người dùng đóng vai trò quan trọng trong việc mang đến trải nghiệm hệ thống một cách trực quan và thân thiện cho người dùng. Thiết kế giao diện không chỉ đảm bảo tính dễ sử dụng mà còn phải hỗ trợ đầy đủ các luồng hoạt động như đăng nhập, đăng xuất, truy cập tài nguyên cũng như xử lý các tình huống đặc biệt. Mỗi thành phần của giao diện được xây dựng nhằm mang đến sự tương tác hiệu quả với hệ thống, sự thuận tiện và bảo mật cao nhất cho người dùng.

Trang đăng nhập là nơi người dùng cần xác thực danh tính để tiếp tục sử dụng hệ thống. Giao diện được thiết kế đơn giản, trực quan với các ô nhập liệu như tài khoản, mật khẩu và nút đăng nhập. Ngoài ra, trang còn cung cấp các chức năng hỗ trợ như quên mật khẩu, đăng ký giúp đảm bảo người dùng dễ dàng truy cập và bắt đầu phiên làm việc một cách an toàn. Giao diện của trang đăng nhập được thể hiện qua Hình 14.



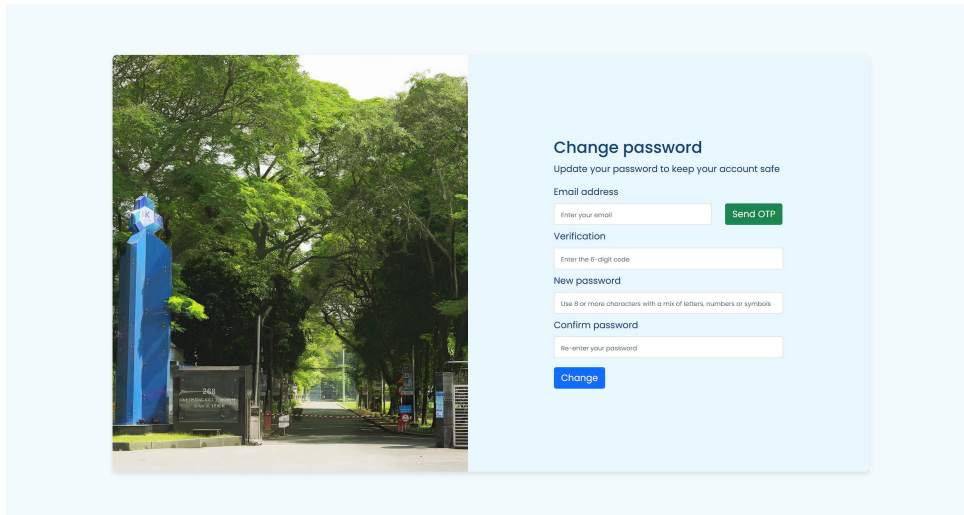
Hình 14: Giao diện trang đăng nhập

Trang đăng ký cho phép người dùng khởi tạo tài khoản mới để có thể sử dụng các tài nguyên của hệ thống. Giao diện được thiết kế rõ ràng với các ô nhập liệu như email, mật khẩu và xác nhận mật khẩu. Ngoài ra, trang còn có một số thông báo lỗi giúp người dùng nhanh chóng hoàn thành quy trình đăng ký. Giao diện của trang đăng nhập được thể hiện qua Hình 15.



Hình 15: Giao diện trang đăng ký

Trang đổi mật khẩu tích hợp tính năng gửi OTP qua email được thiết kế để đảm bảo tính bảo mật cao và trải nghiệm thân thiện cho người dùng. Giao diện được thiết kế đơn giản nhằm nổi bật các tính năng quan trọng như **Gửi OTP** nằm bên cạnh ô nhập liệu email. Đồng thời, giao diện cũng có ô nhập liệu dùng để nhập mã OTP được gửi qua email cũng như nhập mật khẩu mới và xác thực mật khẩu. Trang cũng có thể tích hợp thông báo lỗi hoặc xác nhận thành công để cung cấp phản hồi kịp thời. Giao diện của trang đổi mật khẩu được thể hiện qua Hình 16.



Change password
Update your password to keep your account safe

Email address

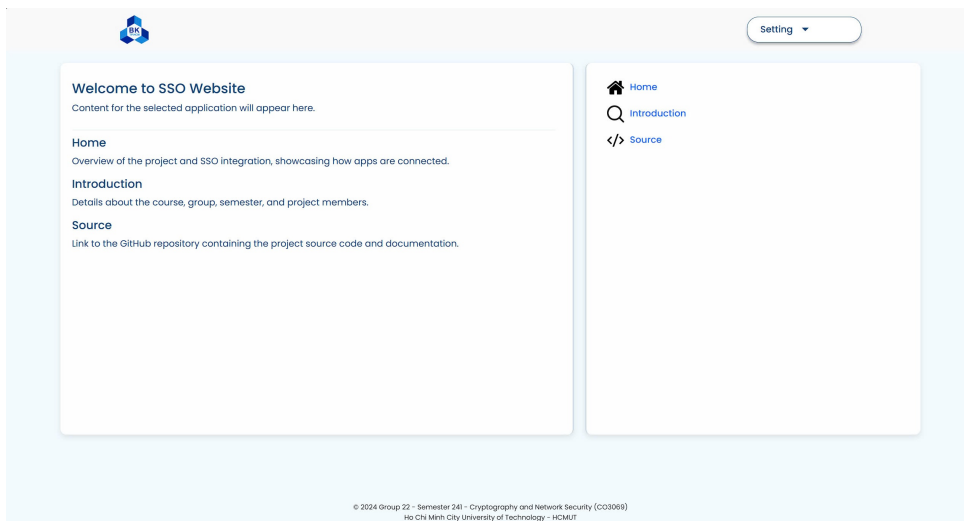
Verification

New password

Confirm password

Hình 16: Giao diện trang đổi mật khẩu

Ngoài ra, trang chủ là nơi mà người dùng thường xuyên tương tác với hệ thống. Trang này bao gồm nút **Setting** chứa các lựa chọn **Đăng nhập** hoặc **Đăng ký**, đồng thời hiển thị các tài nguyên trong hệ thống cũng như mô tả của chúng. Giao diện của trang chủ được thể hiện qua ??.



Welcome to SSO Website
Content for the selected application will appear here.

Home
Overview of the project and SSO integration, showcasing how apps are connected.

Introduction
Details about the course, group, semester, and project members.

Source
Link to the GitHub repository containing the project source code and documentation.

Setting

Home
Introduction
Source

© 2024 Group 22 - Semester 241 - Cryptography and Network Security (C03069)
Ho Chi Minh City University of Technology - HCMUT

Hình 17: Giao diện trang chủ

4 Hiện thực và đánh giá

4.1 Cài đặt và cấu hình SSO

Trong bài tập lớn này, để triển khai hệ thống SSO, nhóm đã sử dụng một tập hợp các thư viện và công nghệ hiện đại nhằm đảm bảo tính linh hoạt, bảo mật và khả năng mở rộng. Các công cụ, thư viện và cấu hình được sử dụng cụ thể được mô tả ở Phụ lục.

Có thể kể đến một số thư viện nổi bật như:

- `jsonwebtoken` : Sử dụng để tạo và xác thực JWT (JSON Web Tokens).
- `bcryptjs` : Hỗ trợ mã hóa mật khẩu người dùng để đảm bảo bảo mật.
- `passport` và `passport-local` : Quản lý xác thực người dùng dựa trên tài khoản và mật khẩu.
- `sequelize` : ORM giúp giao tiếp với MySQL thông qua cú pháp JavaScript dễ sử dụng.
- `nodemailer` : Cung cấp chức năng gửi email, dùng để gửi thông báo hoặc xác thực email.
- `dotenv` : Quản lý biến môi trường thông qua tệp `.env`.
- `express-session` và `connect-session-sequelize` : Hỗ trợ quản lý phiên người dùng.

Hệ thống được cấu hình để sử dụng ORM Sequelize, cho phép dễ dàng làm việc với cơ sở dữ liệu.

```
1 const path = require('path');
2 module.exports = {
3   'config': path.resolve('./src/config', 'config.json'),
4   'migrations-path': path.resolve('./src/', 'migrations'),
5   'models-path': path.resolve('./src/', 'models'),
6   'seeders-path': path.resolve('./src/', 'seeders')
7 }
```

File cấu hình môi trường `.env` chứa các biến quan trọng như thông tin cơ sở dữ liệu, thông số token, và cài đặt cookie. Ví dụ cụ thể:

```
1 PORT=8080
2 REACT_URL=http://localhost:3000
3 JWT_SECRET=nameuser
4 JWT_EXPIRES_IN=1h
5 DB_USERNAME=root //based on username MySQL Workbench 8.0 CE
6 DB_PASSWORD=1234567890 //based on password MySQL Workbench 8.0 CE
7 DB_NAME=sso_user //'based on database MySQL Workbench 8.0 CE
8 DB_HOST=localhost
9 DB_DIALECT=mysql
10 MAX_AGE_ACCESS_TOKEN=900000 #15 MIN
```



```
11 MAX_AGE_REFRESH_TOKEN=3600000 # 60 MIN
12 COOKIE_DOMAIN=localhost
13 GOOGLE_APP_PASSWORD=ipfknenhzjuzlbmu
14 GOOGLE_APP_EMAIL=hoangtanlam2004@gmail.com
```

Những thông số này được sử dụng xuyên suốt hệ thống để đảm bảo tính bảo mật và đồng nhất khi xác thực người dùng.

4.2 Hiện thực tính năng đăng nhập trên Website

4.2.1 Đăng nhập người dùng

Khi người dùng gửi yêu cầu đăng nhập, hệ thống kiểm tra thông tin email hoặc số điện thoại và mật khẩu. Nếu thông tin hợp lệ, hệ thống sẽ trả về mã thông báo (JWT) cho người dùng.

```
1  const handleUserLogin = async (rawData) => {
2    try {
3      let user = await db.User.findOne({
4        where: {
5          [Op.or]: [
6            { email: rawData.valueLogin },
7            { phone: rawData.valueLogin }
8          ]
9        }
10     })
11     if (user) {
12       let isCorrectPassword = checkPassword(rawData.password, user
13         .password);
14       if (isCorrectPassword === true) {
15         const code=uuidv4();
16         let groupWithRoles = await getGroupWithRoles(user);
17         return {
18           EM: 'ok!',
19           EC: 0,
20           DT: {
21             code: code,
22             email: user.email,
23             groupWithRoles,
24             username: user.username
25           }
26         } else {
```

```
27         return {
28             EM: 'Your email or password is incorrect!',
29             EC: 1,
30             DT: ''
31         }
32     }
33 }
34 return {
35     EM: 'Your email/phone number is incorrect!',
36     EC: 1,
37     DT: ''
38 }
39 } catch (error) {
40     console.log(error)
41     return {
42         EM: 'Somthing wrongs in service...',
43         EC: -2
44     }
45 }
46 }
```

Sau khi người dùng đăng nhập, hệ thống tạo ra một JWT (JSON Web Token) chứa các thông tin quan trọng như `id`, `email`, `username`. Token này được sử dụng để xác thực người dùng trong các yêu cầu API tiếp theo mà không cần phải đăng nhập lại.

```
1 const createJWT = (payload) => {
2     let key = process.env.JWT_SECRET;
3     let token = null;
4     try {
5         token = jwt.sign(payload, key, {
6             expiresIn: process.env.JWT_EXPIRES_IN
7         });
8     } catch (err) {
9         console.log(err)
10    }
11    return token;
12 }
13 const verifyToken = (token) => {
14     let key = process.env.JWT_SECRET;
15     let decoded = null;
16     try {
```

```
17     decoded = jwt.verify(token, key);
18   } catch (err) {
19     console.log(err);
20   }
21   return decoded;
22 }
```

Sau khi nhận được token từ client (thông qua cookies hoặc header), hệ thống sử dụng `verifyToken` để giải mã và kiểm tra tính hợp lệ của token. Nếu token hợp lệ, các thông tin người dùng sẽ được gán vào `req.user` để sử dụng cho các yêu cầu tiếp theo.

```
1  const checkUserJWT = (req, res, next) => {
2    if (nonSecurePaths.includes(req.path)) return next();
3    let cookies = req.cookies;
4    let tokenFromHeader = extractToken(req);
5    if ((cookies && cookies.access_token) || tokenFromHeader) {
6      let access_token = cookies && cookies.access_token ? cookies.
access_token : tokenFromHeader;
7      let decoded = verifyToken(access_token);
8      if (decoded) {
9        decoded.access_token = access_token;
10       decoded.refresh_token = cookies.refresh_token;
11       req.user = decoded;
12       next();
13     } else {
14       return res.status(401).json({
15         EC: -1,
16         DT: '',
17         EM: 'Not authenticated the user'
18       })
19     }
20   }
21   else {
22     return res.status(401).json({
23       EC: -1,
24       DT: '',
25       EM: 'Not authenticated the user'
26     })
27   }
28 }
```

Refresh token được sử dụng để tạo lại access token sau khi token đã hết hạn. Refresh token được lưu trữ trong cơ sở dữ liệu và được liên kết với người dùng. Hệ thống sẽ kiểm tra refresh token và cấp lại access token khi cần thiết.

```
1 const updateUserRefreshToken = async (email, token) => {
2   try {
3     let a = await db.User.update(
4       { refreshToken: token },
5       {
6         where: { email: email.trim() }
7       }
8     );
9     console.log(a)
10  } catch (error) {
11    console.log(">>> error: ", error)
12  }
13 }
```

Sau khi xác thực JWT thành công, hệ thống sẽ kiểm tra quyền truy cập của người dùng đối với các tài nguyên bảo mật. Middleware `checkUserPermission` sẽ đảm bảo rằng người dùng có quyền truy cập vào các API được yêu cầu.

```
1 const checkUserPermission = (req, res, next) => {
2   if (nonSecurePaths.includes(req.path) || req.path === '/account')
3     return next();
4   if (req.user) {
5     let email = req.user.email;
6     let roles = req.user.groupWithRoles.Roles;
7     let currentUrl = req.path;
8     if (!roles || roles.length === 0) {
9       return res.status(403).json({
10         EC: -1,
11         DT: '',
12         EM: `you don't permission to access this resource...`
13       })
14     }
15     let canAccess = roles.some(item => item.url === currentUrl ||
16       currentUrl.includes(item.url));
17     if (canAccess === true) {
18       next();
19     } else {
20       return res.status(403).json({
21         EC: -1,
22         DT: '',
23         EM: `you don't permission to access this resource...`
24       })
25     }
26   }
27 }
```

```
19         EC: -1,
20         DT: '',
21         EM: `you don't permission to access this resource...`
22     })
23 }
24 } else {
25     return res.status(401).json({
26         EC: -1,
27         DT: '',
28         EM: 'Not authenticated the user'
29     })
30 }
31 }
```

4.2.2 Đăng xuất người dùng

Khi người dùng yêu cầu đăng xuất, hệ thống thực hiện xóa JWT và refresh token. Nếu token được lưu trong cookies, hệ thống sẽ xóa cookies bằng cách gửi phản hồi từ server và Token không còn hợp lệ sau khi đăng xuất. Người dùng được đưa về localhost:3000.

```
1 const handleLogout = (req, res) => {
2     try {
3         res.clearCookie("refresh_token", {domain: process.env.
4 COOKIE_DOMAIN, path: "/"});
5         res.clearCookie("access_token", {domain: process.env.
6 COOKIE_DOMAIN, path: "/"});
7         return res.status(200).json({
8             EM: 'clear cookies done!', // error message
9             EC: 0, //error code
10            DT: '', //data
11        })
12    } catch (error) {
13        console.log(error)
14        return res.status(500).json({
15            EM: 'error from server', // error message
16            EC: '-1', //error code
17            DT: '', //date
18        })
19    }
20 }
```

4.2.3 Đăng ký tài khoản

Khi người dùng đăng ký tài khoản, hệ thống sẽ kiểm tra xem email của người dùng nhập vào đã tồn tại trong DB chưa. Nếu email đã tồn tại, hệ thống sẽ in ra thông báo **The email is already exist**. Sau khi kiểm tra, thông tin người dùng bao gồm mật khẩu được lưu vào DB. Hệ thống sẽ gửi phản hồi đến người dùng với thông báo tài khoản đã được tạo thành công.

```
1 const createNewUser = async (data) => {
2   try {
3     //check email
4     let isEmailExist = await checkEmailExist(data.email);
5     if (isEmailExist === true) {
6       return {
7         EM: 'The email is already exist',
8         EC: 1,
9         DT: 'email'
10      }
11    }
12
13    //hash user password
14    let hashPassword = hashUserPassword(data.password);
15
16    //hash user password
17    await db.User.create({ ...data, password: hashPassword });
18    return {
19      EM: 'create ok',
20      EC: 0,
21      DT: []
22    }
23  } catch (e) {
24    console.log(e);
25    return {
26      EM: 'something wrongs with servies',
27      EC: 1,
28      DT: []
29    }
30  }
31 }
```

4.2.4 Kết quả hiện thực

Để đánh giá hệ thống hoạt động một cách tốt nhất, nhóm nghiên cứu đã thực thi các luồng đăng ký, đăng nhập, đăng xuất, đổi mật khẩu cũng như truy cập vào tài nguyên của hệ thống trước và sau khi đăng nhập.

Kết quả hiện thực của các luồng hoạt động trên đều được thể hiện dưới dạng video và được đăng tải trên [Google Drive](#).

Ngoài ra, toàn bộ mã nguồn của hệ thống được lưu trữ tại repo [ANM](#) trên GitHub.

4.3 Đánh giá kết quả

Dựa vào kết quả hiện thực, nhóm nghiên cứu nhận thấy quy trình đăng nhập, đăng xuất và xác thực dựa trên JWT mang đến nhiều ưu điểm như sau:

- Tính bảo mật cao: JWT được ký bằng khóa bí mật, khó bị làm giả, Không cần lưu trữ trạng thái trên server, giảm thiểu rủi ro bị đánh cắp session.
- Hiệu suất tốt: Với JWT, không cần truy cập cơ sở dữ liệu liên tục trong mỗi yêu cầu, Token chứa sẵn thông tin, giúp giảm tải cho server.
- Mở rộng dễ dàng: Việc thêm các thông tin vào JWT như quyền truy cập, dữ liệu người dùng rất dễ dàng mà không ảnh hưởng đến hiệu suất của hệ thống.
- Tính tương thích cao: JWT có thể được sử dụng với nhiều loại công nghệ và nền tảng khác nhau, giúp xây dựng các hệ thống đa dạng.

5 Kết luận

Qua quá trình phân tích và thực hiện Bài tập lớn, nhóm nghiên cứu đã phân tích được các yêu cầu chức năng và yêu cầu hệ thống, đặc biệt là yêu cầu về triển khai hệ Single Sign-On (SSO) giúp người dùng đăng nhập một lần và có thể truy cập nhiều ứng dụng mà không cần phải nhập lại thông tin đăng nhập. Đặc biệt, nhóm cũng đã tìm hiểu chi tiết về các giao thức liên quan đến SSO, đưa ra những điểm mạnh, điểm yếu của từng giao thức để từ đó đánh giá các giao thức dựa trên những tiêu chí liên quan.

Nhóm nghiên cứu cũng đã triển khai được kiến trúc hệ thống SSO, bao gồm việc xây dựng các thành phần chính như Identity Provider (IdP), Authorization Code Exchange và các quy trình xác thực người dùng. Hệ thống đã được triển khai với đầy đủ các tính năng cần thiết như quản lý tài khoản người dùng, trao đổi mã ủy quyền và cấp token bảo mật.

Một trong những yếu tố quan trọng mà nhóm đã thực hiện được chính là việc thiết kế giao diện người dùng dễ sử dụng và trực quan. Giao diện đã được tối ưu hóa cho trải nghiệm người dùng, giúp người dùng dễ dàng đăng nhập và truy cập vào các ứng dụng khác nhau. Các chức năng như đăng nhập, đăng ký, quên mật khẩu, và hỗ trợ đã được tổ chức hợp lý.

Mặc dù nhóm đã đạt được các mục tiêu cơ bản, nhưng vẫn còn một số yếu tố cần cải thiện, đặc biệt là việc hỗ trợ đa nền tảng và tích hợp nhiều hệ thống IdP khác nhau. Hệ thống chưa được kiểm thử hiệu suất với số lượng lớn người dùng và thiếu các biện pháp bảo mật nâng cao như Multi-Factor Authentication hay phát hiện hành vi bất thường. Ngoài ra, khả năng mở rộng để hỗ trợ các ứng dụng phức tạp hơn hoặc môi trường phân tán vẫn chưa được tối ưu.

Trong tương lai, nhóm hướng tới việc mở rộng hỗ trợ đa IdP, bao gồm các nhà cung cấp phổ biến như Google, Facebook và Microsoft, đồng thời phát triển cơ chế chuyển đổi IdP linh hoạt. Các biện pháp bảo mật nâng cao như xác thực đa yếu tố, phát hiện hành vi bất thường và nâng cấp mã hóa cũng sẽ được triển khai để tăng tính an toàn cho hệ thống. Nhóm nghiên cứu cũng sẽ tập trung vào việc tối ưu hiệu suất để hỗ trợ số lượng lớn người dùng đồng thời và đảm bảo hệ thống hoạt động tốt trên mọi nền tảng, bao gồm các thiết bị di động và trình duyệt cũ. Với những định hướng phát triển này, hệ thống SSO hứa hẹn sẽ ngày càng hoàn thiện, đáp ứng tốt hơn các yêu cầu thực tế và mở rộng quy mô sử dụng.

Tài liệu tham khảo

- [1] Frontegg (2024). *What Is SSO (Single Sign-On)?*. Truy cập từ <https://frontegg.com/guides/single-sign-on-sso>.
- [2] Phạm Hary (2020). *Login-Register Node.js with Keep Old Inputs*. Truy cập từ <https://github.com/haryphamdev/login-register-nodejs-keep-old-inputs>.
- [3] Phạm Hary (2021). *React Basic - Hoi Dan IT*. Truy cập từ <https://github.com/haryphamdev/react-basic-hoi-dan-it>.
- [4] Mark Scapicchio (2024). *What is SSO?*. Truy cập từ <https://www.ibm.com/topics/single-sign-on>.
- [5] Microsoft (2024). *Protocol flow - sign-in*. Truy cập từ <https://learn.microsoft.com/en-us/entra/identity-platform/v2-protocols-oidc#protocol-flow-sign-in>.
- [6] Rahul Awati (2024). *OpenID (OpenID Connect)*. Truy cập từ <https://www.techtarget.com/whatis/definition/OpenID>.
- [7] Đình Tài (2023). *Tìm hiểu về OAuth2, OpenID Connect, SAML và Kerberos: Các phương thức xác thực và ủy quyền trong ứng dụng hiện đại*. Truy cập từ https://viblo.asia/p/tim-hieu-ve-oauth2-openid-connect-saml-va-kerberos-cac-phuong-thuc-xac-thuc-va-uy-quyen-trong-ung-dung-hien-dai-GAWVpx2oV05#_53-ung-dung-cua-saml-trong-single-sign-on-sso-19.
- [8] Trịnh Duy Thanh (2022). *OpenID Connect là gì? Cách xác minh danh tính bằng OIDC*. Truy cập tại <https://bkhost.vn/blog/openid-connect-oidc/>.
- [9] Thuý Vy (2024). *Tìm Hiểu Single Sign-On (SSO): Giải pháp đăng nhập một lần*. Truy cập từ https://2001lab.io/blog/sso-la-gi/?srsltid=AfmBOorvZXmFEJKSmv0zG_rcw6-IHEZy3uKyOYRG8fciM6st1xwYbea_.

Phụ lục

1. Môi trường phát triển ứng dụng

Trong quá trình phát triển ứng dụng, việc chọn lựa môi trường phù hợp vô cùng quan trọng để tối ưu hóa hiệu quả công việc. Môi trường phát triển ứng dụng của nhóm được xây dựng trên nền tảng các công nghệ phổ biến và mạnh mẽ, giúp việc phát triển ứng dụng trở nên dễ dàng và nhanh chóng hơn. Các công cụ và công nghệ được sử dụng trong quá trình phát triển bao gồm `Node.js`, `React.js`, `MySQL` và hệ điều hành `Windows 11`.

- `Node.js`: Là nền tảng JavaScript phía server cho phép phát triển các ứng dụng web nhanh chóng và hiệu quả. Nhờ vào mô hình bất đồng bộ và sự kiện, `Node.js` có khả năng xử lý đồng thời nhiều tác vụ mà không làm giảm hiệu suất của hệ thống.
- `React.js`: Là thư viện JavaScript mạnh mẽ giúp phát triển giao diện người dùng (UI) linh hoạt và dễ bảo trì. `React.js` giúp tạo ra các thành phần giao diện có thể tái sử dụng, tăng cường hiệu suất của ứng dụng.
- `MySQL`: Là hệ quản trị cơ sở dữ liệu quan hệ giúp lưu trữ, truy vấn và quản lý dữ liệu một cách hiệu quả. Đây là lựa chọn lý tưởng cho các ứng dụng yêu cầu một cơ sở dữ liệu ổn định và dễ quản lý.
- `Windows 11`: Là hệ điều hành sử dụng trên máy tính phát triển, cung cấp các công cụ phát triển mạnh mẽ, hỗ trợ việc lập trình, biên dịch và kiểm thử hệ thống một cách hiệu quả.

Môi trường phát triển này không chỉ giúp nhóm tối ưu hóa các công việc liên quan đến lập trình và kiểm thử mà còn giúp hệ thống trở nên linh hoạt, dễ bảo trì và có khả năng mở rộng trong tương lai.

2. Thư viện

Trong quá trình phát triển hệ thống, nhóm đã sử dụng nhiều thư viện mã nguồn mở để hỗ trợ xây dựng và tối ưu hóa hệ thống. Các thư viện này được lựa chọn cẩn thận, đảm bảo đáp ứng đầy đủ yêu cầu về bảo mật, hiệu năng, tính năng, và khả năng mở rộng. Dựa trên chức năng và vai trò, thư viện được phân thành hai nhóm chính là backend và frontend.

Các thư viện backend được sử dụng để xây dựng server, xử lý logic, bảo mật dữ liệu và kết nối với cơ sở dữ liệu. Những thư viện này giúp phát triển một hệ thống mạnh mẽ, an toàn và dễ dàng tích hợp với các công nghệ khác.

```
1 {  
2   "@babel/core": "7.15.4",  
3   "@babel/node": "7.15.4",  
4   "@babel/preset-env": "7.15.4",  
5   "bcryptjs": "2.4.3",  
6   "bluebird": "3.7.2",  
7   "body-parser": "~1.20.3",
```

```
8  "connect-flash": "~0.1.1",
9  "connect-session-sequelize": "~7.1.3",
10 "cookie-parser": "1.4.7",
11 "dotenv": "10.0.0",
12 "ejs": "3.1.10",
13 "express": "~4.21.1",
14 "express-session": "~1.18.1",
15 "jsonwebtoken": "8.5.1",
16 "lodash": "~4.17.21",
17 "mysql2": "2.3.3",
18 "nodemailer": "6.7.5",
19 "nodemon": "~3.1.7",
20 "passport": "0.5.2",
21 "passport-local": "1.0.0",
22 "sequelize": "6.37.5",
23 "sequelize-cli": "6.3.0",
24 "uuid": "8.3.2",
25 "yargs": "~17.7.2"
26 }
```

Các thư viện frontend tập trung vào việc xây dựng giao diện người dùng (UI), quản lý trạng thái ứng dụng và cung cấp trải nghiệm mượt mà. Các công cụ này không chỉ giúp giảm thiểu thời gian phát triển mà còn đảm bảo giao diện hiện đại và dễ sử dụng.

```
1  {
2    "@babel/core": "7.15.4",
3    "@babel/node": "7.15.4",
4    "@babel/preset-env": "7.15.4",
5    "bcryptjs": "2.4.3",
6    "bluebird": "3.7.2",
7    "body-parser": "~1.20.3",
8    "connect-flash": "~0.1.1",
9    "connect-session-sequelize": "~7.1.3",
10   "cookie-parser": "1.4.7",
11   "dotenv": "10.0.0",
12   "ejs": "3.1.10",
13   "express": "~4.21.1",
14   "express-session": "~1.18.1",
15   "jsonwebtoken": "8.5.1",
16   "lodash": "~4.17.21",
17   "mysql2": "2.3.3",
18   "nodemailer": "6.7.5",
19   "nodemon": "~3.1.7",
20   "passport": "0.5.2",
21   "passport-local": "1.0.0",
```

```
22 "sequelize": "6.37.5",  
23 "sequelize-cli": "6.3.0",  
24 "uuid": "8.3.2",  
25 "yargs": "^17.7.2"  
26 }
```

3. Mã nguồn mở

Trong quá trình phát triển ứng dụng, nhóm nghiên cứu đã sử dụng mã nguồn mở từ các repo [GitHub](#). Các mã nguồn mở này cung cấp các module và giải pháp hữu ích cho việc xây dựng các tính năng cơ bản của ứng dụng, từ việc xử lý đăng nhập và đăng ký người dùng đến việc tạo giao diện người dùng.

[Login-Register Node.js with Keep Old Inputs](#) là mã nguồn mở được xây dựng bằng `Node.js`, cung cấp một hệ thống đăng nhập và đăng ký người dùng với tính năng giữ lại thông tin đã nhập khi xảy ra lỗi trong quá trình đăng nhập hoặc đăng ký. Mã nguồn này sử dụng `Express.js` cho phần backend và có thể dễ dàng mở rộng để tích hợp các tính năng bảo mật khác như mã hóa mật khẩu và xác thực hai yếu tố.

Mã nguồn [React Basic Hoi Dan IT](#) là một ứng dụng React cơ bản, được xây dựng để tạo giao diện người dùng đơn giản cho các trang web. Mã nguồn này sử dụng `React.js` cho phần frontend và là một điểm khởi đầu tuyệt vời để xây dựng các ứng dụng web động.

Cả hai mã nguồn đều sử dụng giao thức HTTPS để đảm bảo tính bảo mật trong việc truyền tải dữ liệu giữa client và server. Việc sử dụng HTTPS giúp mã hóa thông tin và đảm bảo an toàn cho người dùng khi họ thực hiện các giao dịch hoặc tương tác với ứng dụng.