Application, Information System and Software Development Journal
Appissode Journal
Vol. 1, No. 1, Dec 2023, Page 20-26
E-ISSN: 3032-5625, P-ISSN: 3032-5269

Web-based Single Sign-On Approach for the Authorization Server Using OAuth 2.0 Protocol in Academic System

Tri Herdiawan Apandi[1], Mohammad Iqbal[1], Rian Piarna[1], Dwi Vernanda[1], Aulia Rizky Muhammad Hendrik Noor Asegaff[2]
[1]Politeknik Negeri Subang, Indonesia
[2]Universitas Islam Kalimantan Muhammad Arsyad Al Banjari, Indonesia

Corresponding Author: Tri Herdiawan Apandi
Information System Department
Politeknik Negeri Subang, Indonesia
Email: tri@polsub.ac.id

Article History

Received Nov 21, 2023
Accepted Dec 9, 2023
Published Dec, 2023

Keywords

single sign-on, oauth 2.0, authorization server, authentication, authorization.

Abstract

Users can access several services simultaneously by remembering just one valid login and password thanks to Single Sign On (SSO), an access control method. Using Oauth 2.0 technology, this research creates a system that offers Single Sign-On services via a web application. OAuth is a mechanism to streamline the data exchange flow process, which involves at least two applications interacting with one another. It is an entity that can provide access rights to protected resources. The resource is supplied by an application known as an Oauth Provider, and the application that receives it is known as an Oauth client. The purpose of this study is to develop and put into practice protocols for the OAuth 2.0 technology's operating mechanism, which involves server authorization—the server resource that handles client credential authentication and authorization. Three (three) client apps that implemented the OAuth 2.0 authentication work procedure were employed as single sign-on applications in this study. By using the Authorization Server role for OAuth 2.0 authentication, the client's credentials will be verified against the data kept in the database. This is accomplished by issuing an authorization page (Authorise App), which will direct users to the home page of each individual web application. The creation of an access token for every web application serves as the last line of verification for the accuracy of a client's credentials.

1 INTRODUCTION

The rise in internet usage has been influenced by the development of more advanced information technologies. It also promotes the creation of diverse online content. Because there is so much stuff on the internet, users have to memorise several usernames and passwords for every piece of content that has to be authenticated or logged in. A campus web portal is made up of several systems (applications) that are available for instructors, staff, and students. Users must enter their login credentials for each application they use. As a result, the user must commit each application's unique username and password to memory. In order to reset their login password and use the programme, users must notify the administrator if they lose their username and password.

Open authorization standards like the OAuth protocol let consumers access apps without disclosing their credentials. Access privileges can be granted to someone by means of OAuth, which converts credentials (password and username) into access tokens. By entrusting the existence of a third party that is regarded as a reliable server resource with the authority to order and maintain all current credentials, OAuth authentication is set up to ensure that users of application services remain safe. In this instance, Twitter uses OAuth, for instance, third parties such as Uber, Dabr, TweetDeck, or Tuitwit. You can use TweetDeck, Dabr, Uber, or Tuitwit to check your status on Twitter. This application allows you to easily access the status in the Dabr application (third party), including the inbox.

2 SINGLE SIGN-ON

With SSO, users can access many services simultaneously by simply remembering one valid login and password. After a first SSO authentication, authentication happens automatically each time a user opens a new website within a session. Among the advantages of SSO are [1]:

a. Use distinct usernames and passwords to lessen password weariness (password fatigue).
b. Cut down on time lost by constantly typing in the same password.
c. Lower IT expenses associated with the quantity of password-related inquiries.
d. Security at every point of entrance, departure, and system entry without requiring a password reset.
e. Centralised logs or records.

One technology that will advance in web technology in the future is the web portal. Multiple portlets that can transmit data from multiple sources are combined into a single portal page [1].

Through the creation of an interaction agreement between the resource owner and the HTTP service, or by granting access to third-party applications on their own behalf, OAuth is a protocol that enables party applications to obtain restricted access to HTTP services [2].

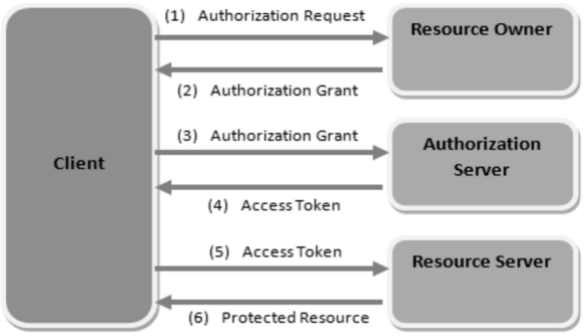

Figure 1. Authorization flow on Oauth 2.0 (Greg Brail & Sam Ramji, 2012)

Terdapat 4 peran utama dalam mekanisme kerja OAuth 2.0 yakni [3]:

a. Resource Server
   Users using an API (Application Programming Interface) and OAuth protection use the Resource Server (Server Resources). An API publisher called Resource Server is able to store and handle data including calendars, contacts, movies, and images.
b. Resource Owner
   Present yourself as the owner of the application and a resource. Resource owners can use installed programmes to access server resources.
c. Client
   An application that performs authorization on a Resource Server to make API requests that have been secured for the Resource Owner's benefit.
d. Authorization Server
   By executing and granting clients access tokens to access protected resources that are already available on the Resource Server, the Authorization Server obtains permission from the Resource Owner.

The four active roles that make up the OAuth 2.0 functioning mechanism are the Client, Resource Owner, Authorization Server, and Resource Server [4]. The OAuth 2.0 authorization stages are as follows:

a. The Resource Owner is contacted by the client to request authorization. Requests for authorization can be sent straight to the Resource Owner or, in the event that they can't be made directly, via the Authorization Server.
b. Authorization approval is given to the client; this credential signifies client ownership authorization. Depending on the client's approach and the kind that the Authorization Server supports, this authorization may be granted.
c. The Authorization Server provides the client with a grant presentation and an authorization form once the client requests token access with authentication.
d. Access tokens are distributed by the Authorization Server (Authorization Server), which also provides client authentication and verifies that the client has been granted authorization when necessary.

e. The client authenticates by presenting an access token and requests resources that are protected from the Resource Server.
f. The client's request to access the protected application is fulfilled by the Resource Server once it has verified the access token and determined it is valid and suitable.

3 ANALYSIS AND DESIGN

In monolith applications, different usernames and passwords for online applications X and Y are possible due to the abundance of account data in each web application database on campus. Because usernames and passwords are saved in separate databases for each programme that a user accesses, there is a high probability that a user will forget them if they use many applications. This makes it challenging for users to log in to multiple online applications.

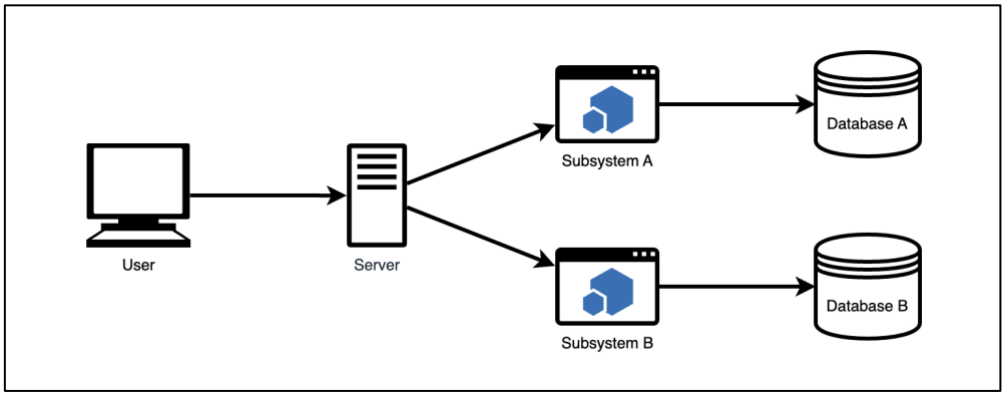

Figure 2. Monolith architecture

The app that needs to be made has the following general description:

a. Client authentication requests are sent over the OAuth API to the Provider application or single sign-on web application through a browser by visitors to the client application.
b. The authorization server determines and processes the account that belongs to the visitor (client), checking to see if the account information is consistent with the resource server database.
c. By returning an authentication code, the authorization server verifies that the client's credentials were correctly authorised through the browser, assuming that the client's account has been deemed valid.
d. Upon being deemed compliant, clients will receive an authorise app page, which is an authorization request page forwarded to the authorization server. To obtain an access token, the application callback script transmits an authentication code.
e. The access token will be shown to the client together with the main page of the client application web if the client completes authorization.

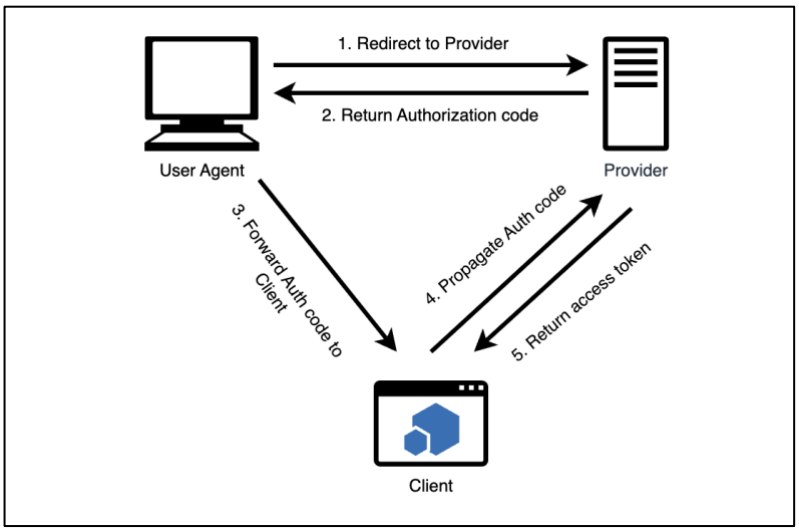

Figure 3. Authorization flow

The main flow of the campus portal application's single sign-on is explained in the use-case diagram. The use case diagram will describe the actors' identities and capabilities.

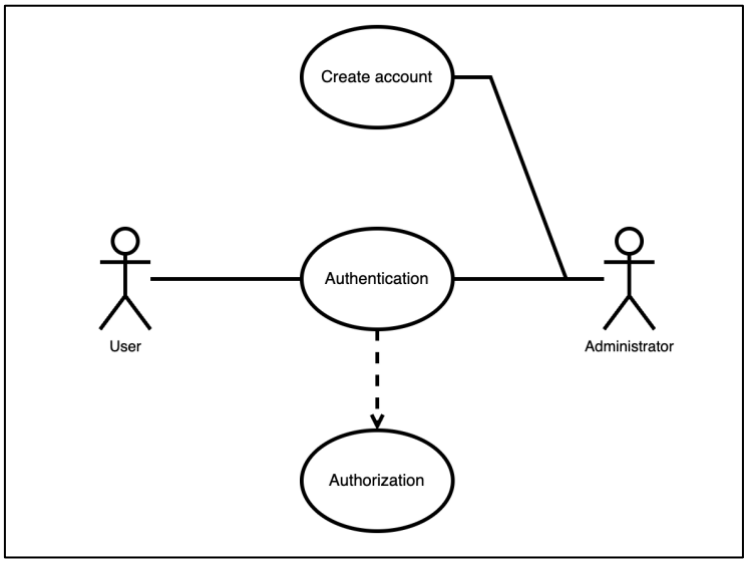

Figure 4. Provider's usecase

Table 1. Provider's usecase specification

Main Actor	User, Admin
Start Condition	Not yet authenticated
End Condition	Has been authenticated

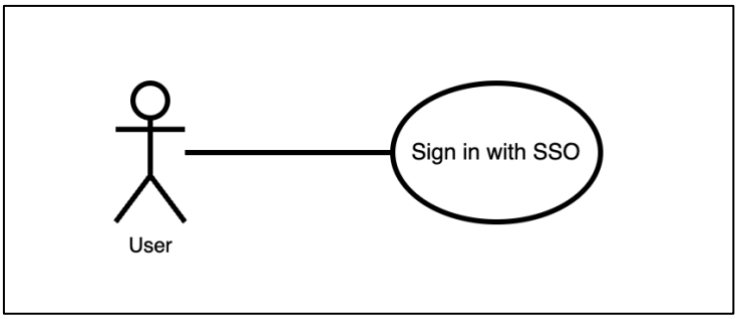

Figure 5. Client's usecase

Table 2. Client's usecase specification

Main Actor	User
Start Condition	Not yet authenticated
End Condition	Has been authenticated

The workflow for each use case that has been previously outlined is represented by an activity diagram. The process for each use case in the single sign-on system will therefore be represented in the activity diagram that will be made below.

Implementation Environment

This application uses the following hardware and software in its implementation environment:

Processor	Intel Core i5, 2.4GHz, 3MB Cache
RAM	2GB DDR3 1066MHz
OS	Windows 10, 64-bit
Server	Apache HTTP Server
Database	MySQL
Web Browser	Mozilla Firefox

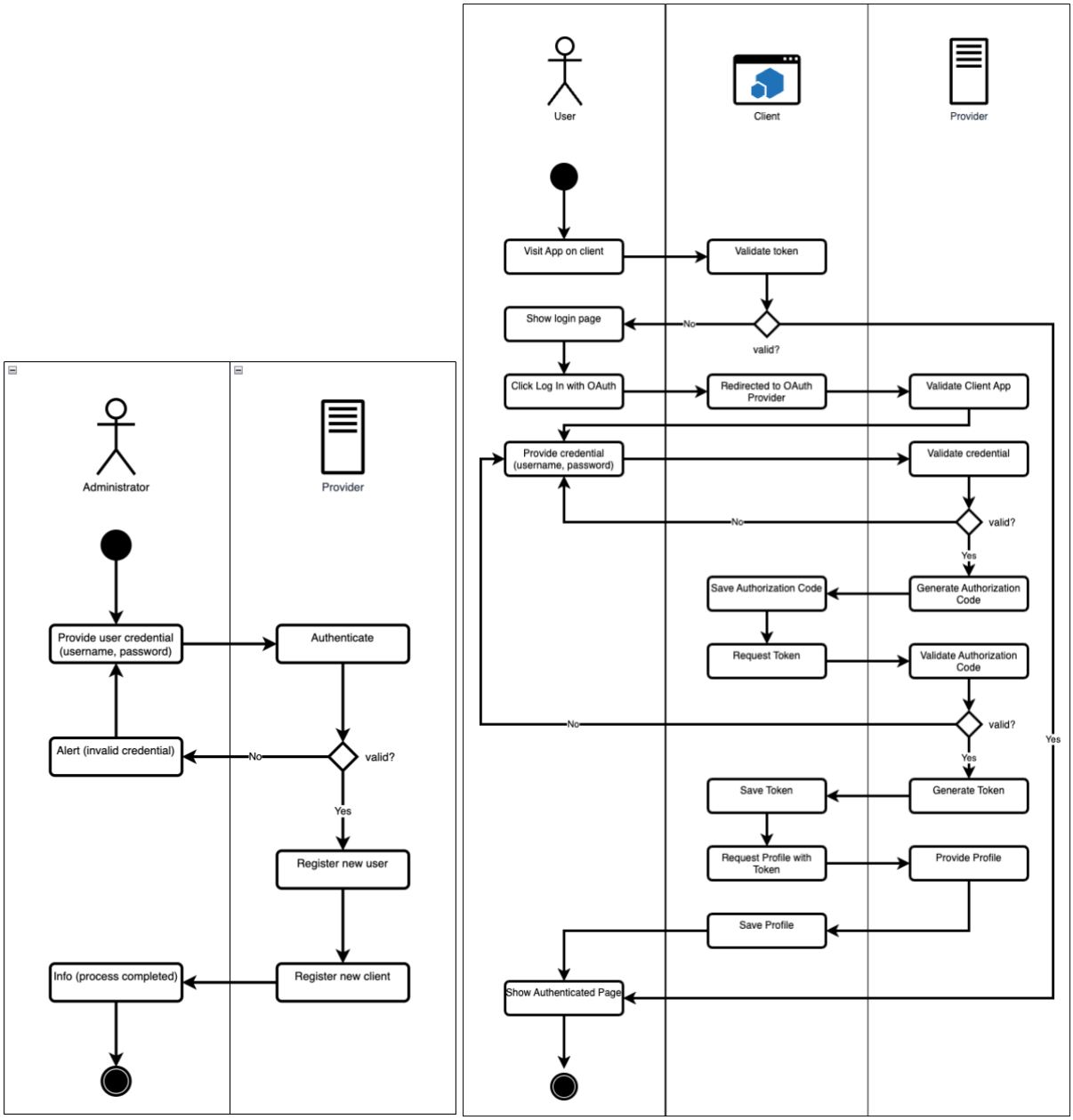

Figure 6. Activity diagram: Administrator authentication (left); Authentication with SSO (right)

Limitations

The following are the limits of the system's implementation:

a. Implementation of the Administrator's sign-in procedures.
b. Putting into action plans to bring on additional users (system users).
c. Putting user sign-in procedures into practice on provider apps.
d. An authorization server is used to implement the OAuth 2 authentication process on client applications. It issues an authorise app page, or application authorization, and then authorises it by providing an access token.

System Testing

Based on the outcomes of the black box method testing, it is possible to draw the conclusion that the system functions flawlessly and is error-free when executing system operations.

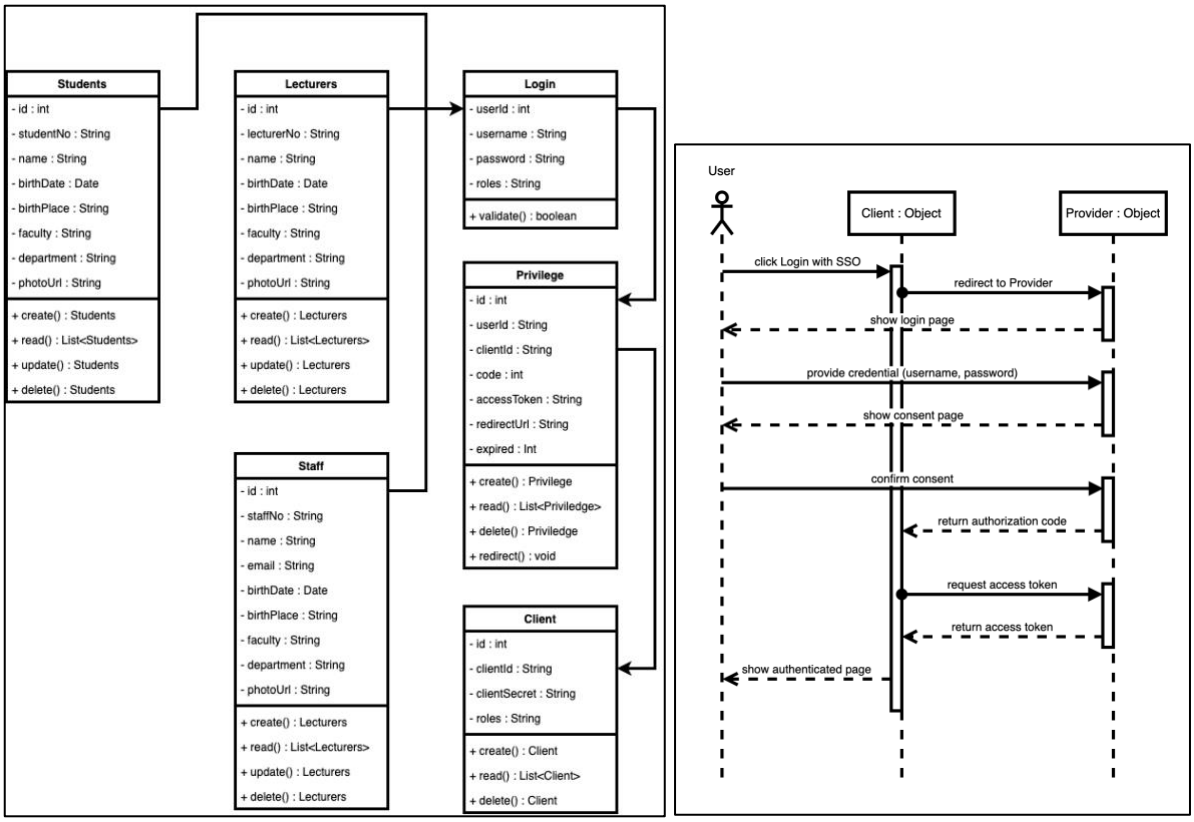

Figure 7. Class diagram (left); Sequence diagram (right)

4 DISCUSSIONS

Some of the conclusions drawn from conducted experiments are listed below:

a. Using OAuth 2 authentication technology, which is part of the client credential authentication process that involves a visitor to the campus portal web application, the application has, in general, been able to reduce the use of usernames and passwords on the campus web portal, which is based on single sign on.
b. An authorization server, also known as a resource server, is a component of OAuth 2 authentication system that is responsible for displaying the application authorization process page (authorise app).
c. The authorization server will deliver an access token in the form of app authorizations that the client has approved.
d. Every time a client completes a sign-in activity successfully, the authorization server generates an access token that differs from the previous one.

REFERENCES

[1] M. Irsan, D. F. Murad and Ahsanfile, "Single Authentication for Multiple Access with SSO (Single Sign On)," in *Seminar Nasional Inovasi dan Teknologi (SNIT) 2012*, 2012.

[2] E. Hammer, "OAuth 2.0," 11 2012. [Online]. Available: http://hueniverse.com/oauth/.

[3] E. D. Hardt, "The OAuth 2.0 Authorization Framework," 10 2023. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-oauth-v2-31.

[4] Yosrinal, "Perancangan dan Implementasi Resource Server dan Authorization Server Menggunakan Teknologi Otentikasi OAuth 2.0," Universitas Sumatera Utara, Medan, 2014.

[5] "About OAuth 2.0," 10 2023. [Online]. Available: http://oauth.net/2/. [Accessed 2023].

[6] R. Byod, Getting Started With OAuth 2.0, California: O'Reilly Media, 2012.

[7] Chiragsh, "OAuth 2.0," 10 2023. [Online]. Available: https://code.google.com/p/google-api-php-client/wiki/OAuth2.

[8] A. Parecki, "OAuth 2 Simplified," 2023, 10. [Online]. Available: http://aaronparecki.com/articles/2012/07/29/1/oauth2-simplified.

[9] S. Putera, R. Fibrian, A. F. Rohim and Y. Christiyono, "Pembangunan Sistem Otentikasi Terpusat Berbasis Lightweight Directory Access Protocol," Universitas Diponegoro, Semarang, 2011.

[10] P. P. Nugroho, "Pengembangan Model Single Sign-On Untuk Layanan Internet dan Proxy IPB," Institut Pertanian Bogor, Bogor, 2014.

[11] "Using OAuth 2.0 to Access Google APIs," Google Developer, 10 2023. [Online]. Available: http://developers.google.com/accounts/docs/OAuth2.

BIOGRAPHY OF AUTHORS

Tri Herdiawan Apandi

He teaches at Politeknik Negeri Subang in Information System Department. His current research focus is web-based software development.

Mohammad Iqbal

He teaches at Politeknik Negeri Subang in Information System Department. His current research focus is web-based software development.

Rian Piarna

He teaches at Politeknik Negeri Subang in Information System Department. His current research focus is web-based software development.

Dwi Vernanda

She teaches at Politeknik Negeri Subang in Information System Department. His current research focus is web-based software development.

Aulia Rizky Muhammad Hendrik Noor Asegaff

He teaches at Universitas Islam Kalimantan Muhammad Arsyad Al Banjari in Information Technology Faculty. His current research focus is software engineering.