PAPER • OPEN ACCESS

Implementation of google single sign On (SSO) in the library management system

To cite this article: A Purwinarko *et al* 2021 *J. Phys.: Conf. Ser.* **1918** 042129

Implementation of google single sign On (SSO) in the library management system

A Purwinarko[1]*, W Hardyanto[2] and M A Adhi[2]

[1]Department of Computer Science, Faculty of Mathematics and Natural Sciences, Universitas Negeri Semarang
[2]Department of Physics, Faculty of Mathematics and Natural Sciences, Universitas Negeri Semarang

*Corresponding author: aji.purwinarko@mail.unnes.ac.id

Abstrak. Almost all organizations provide web-based services as an effort to provide excellent services to users, such as a library, academic, financial, and other services. Organizations only need to provide Single Sign-on (SSO) services for their users, and users need one set of credentials (user name and password) to access all organizational services. The SSO service developed is based on the Google Authentication API. This SSO makes communication between systems more effective and efficient.

1. Introduction

At present, almost all organizations (universities) provide web-based services (information systems) as an effort to provide excellent services to users, such as a library, academic, financial, and other services. Between information systems in this organization are expected to provide fast and accurate services to users [1].

Thus, an organization's Information Technology (IT) team only needs to provide Single Sign-on (SSO) services to its users, and users only need one set of credentials (say, user names and passwords) to access all organizational services [2].

Web-based SSO systems can allow users to enter a single interface (such as through a web browser) and then provide SSO services to users to log into one or several web applications with the same identity. Various methods can be applied in SSO, such as form-fill, Federated (OIF), SSO Protected (OAM), and other policies [3].

In the SSO service system, IDP Providers (physical or logical) IDP verifies the credentials of all users who want to receive web-based services. At present, several social networking sites such as Facebook, Google, or LinkedIn work as IDPs, known as social logins [2,4]. Facilitating SSO in several browser instances such that user authentication on one browser instance is used as a basis to allow access to other protected browser instances [5,6,7]. Simple SSO will only have one set of credentials, for example, using an email system as an authentication authority for several systems [8].

Two variants might technically implement Single Sign-On in a heterogeneous software environment: 1) Using specialized software that cuts authentication requests sent by various web portal services to users and automatically sends their credentials, such as account names and words password. Because authentication data is stored in the Single Sign-On database, interception is not visible to end-users; 2) Integration with other software authentication processes. The main advantages of this solution are: 1)

Reducing the amount of authentication data needed to work with the organization's network services; 2) Reducing the time needed for authentication on many organizational network portals. The main disadvantage of this concept is the risk of a single password obtained by a third party, which can lead to unauthorized access to the entire work environment [9].

OAuth 2.0 is an authorization protocol that allows one to provide service rights to access resources on other services. This fact makes OAuth suitable for use in heterogeneous web service environments. The protocol eliminates the need to trust application logins and passwords, and also allows issuing a limited set of rights, rather than all at once [8]. OAuth 2.0 involves three parties, namely client, web service, and server, as shown in Figure 1.

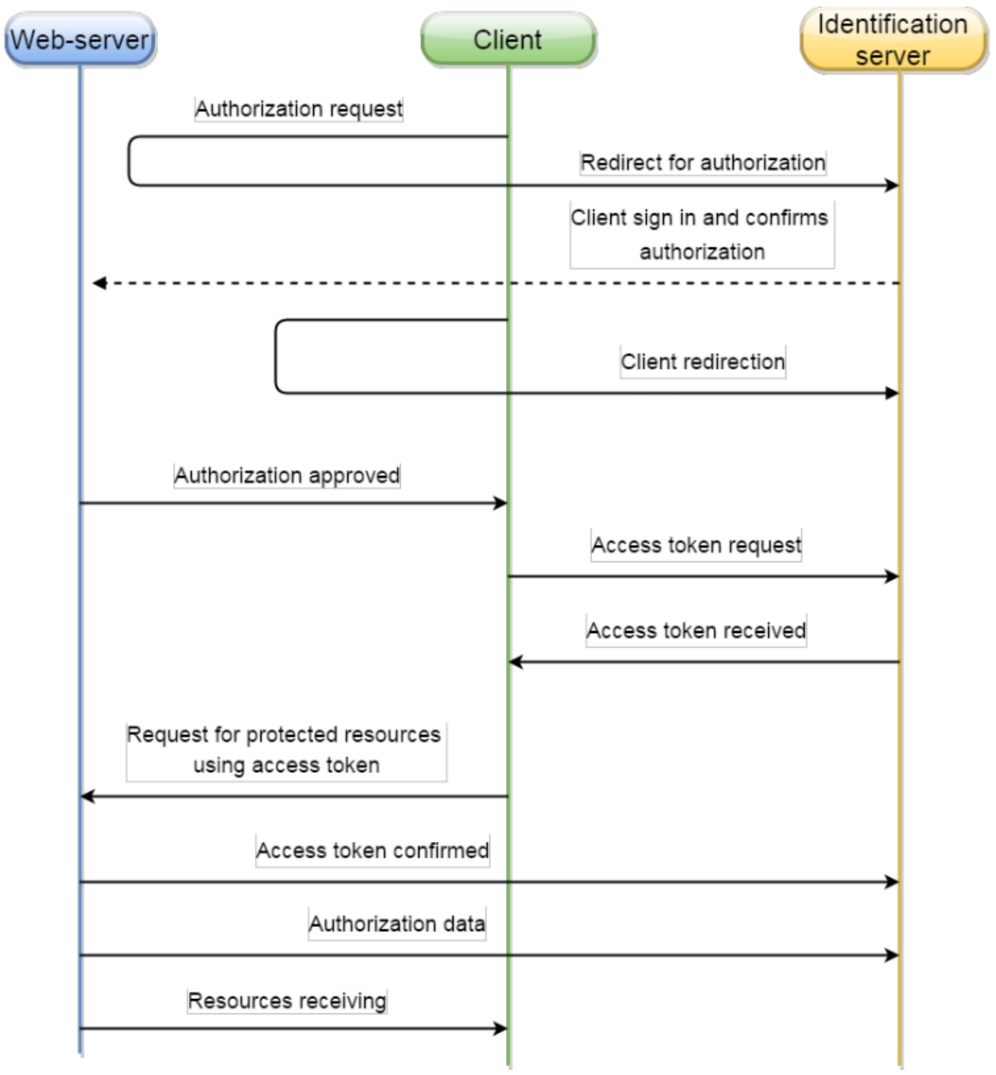


Figure 1. OAuth authorization flow [8].

Social networking services widely use application Programming Interfaces (APIs) and make them available to third-party application developers to embed social logins into their applications [10]. Google has provided an API authentication version of JavaScript, where content can only be accessed after logging in with a Google account, this authentication flow is shown in Figure 2. Most application developers apply Google Api as an SSO service [11].

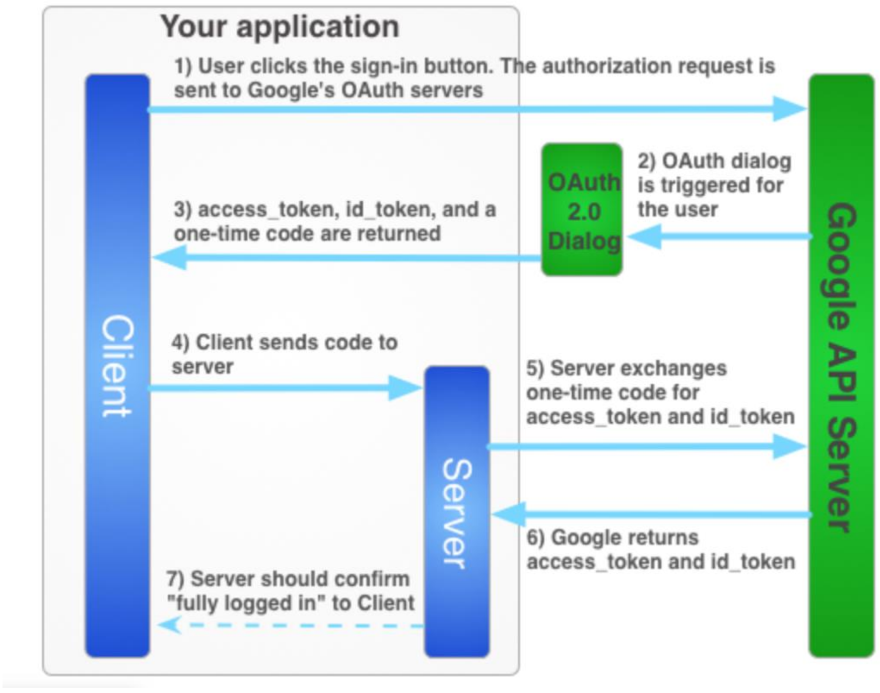

Your application
1) User clicks the sign-in button. The authorization request is sent to Google's OAuth servers
2) OAuth dialog is triggered for the user
OAuth 2.0 Dialog
3) access_token, id_token, and a one-time code are returned
4) Client sends code to server
5) Server exchanges one-time code for access_token and id_token
6) Google returns access_token and id_token
7) Server should confirm "fully logged in" to Client
Client
Server
Google API Server

Figure 2. Google authentication flow.
(developers.google.com)

2. Method

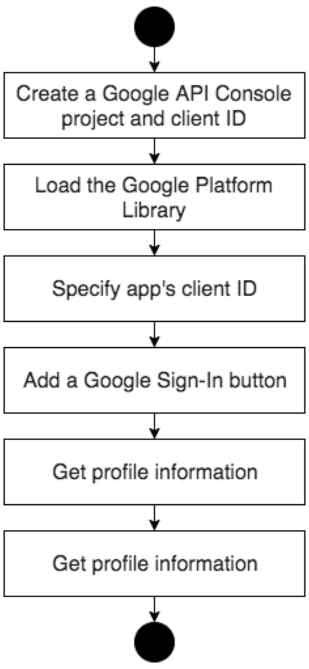
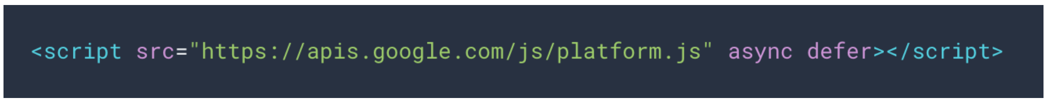

Create a Google API Console project and client ID
Load the Google Platform Library
Specify app's client ID
Add a Google Sign-In button
Get profile information
Get profile information

Figure 3. SSO method.

Figure 3 shows the designing SSO method and is described as follows:

a. Application developers need to create a Google API Console project and client ID.

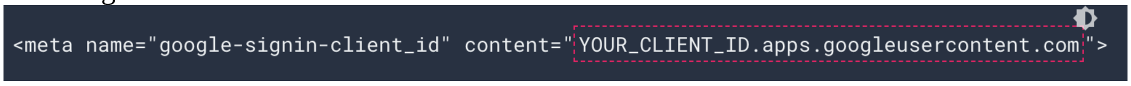
b. Need to include Google Platform Library on the page that will be integrated with SSO



```
<script src="https://apis.google.com/js/platform.js" async defer></script>
```

Figure 4. Load Google Platform Library.

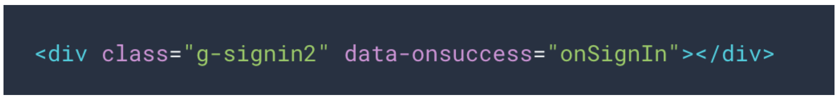
c. Including the client ID that has been created on the meta element



```
<meta name="google-signin-client_id" content="YOUR_CLIENT_ID.apps.googleusercontent.com">
```

Figure 5. Meta element

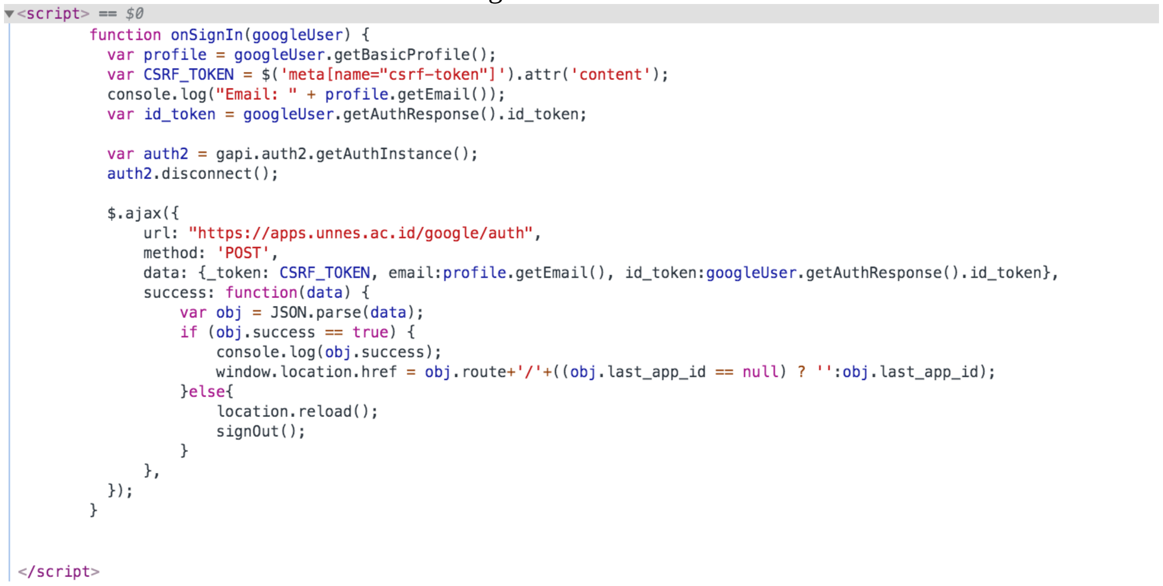
d. Add a sign-in button code



```
<div class="g-signin2" data-onsuccess="onSignIn"></div>
```

Figure 6. Sign-in button code

e. Get user information after successful login



```
▼<script> == $0
        function onSignIn(googleUser) {
          var profile = googleUser.getBasicProfile();
          var CSRF_TOKEN = $('meta[name="csrf-token"]').attr('content');
          console.log("Email: " + profile.getEmail());
          var id_token = googleUser.getAuthResponse().id_token;

          var auth2 = gapi.auth2.getAuthInstance();
          auth2.disconnect();

          $.ajax({
              url: "https://apps.unnes.ac.id/google/auth",
              method: 'POST',
              data: {_token: CSRF_TOKEN, email:profile.getEmail(), id_token:googleUser.getAuthResponse().id_token},
              success: function(data) {
                  var obj = JSON.parse(data);
                  if (obj.success == true) {
                      console.log(obj.success);
                      window.location.href = obj.route+'/'+((obj.last_app_id == null) ? '':obj.last_app_id);
                  }else{
                      location.reload();
                      signOut();
                  }
              },
          });
        }


</script>
```

Figure 7. Get user profile.

3. Result and Discussion

This study produces a login page, as shown in Figure 8.

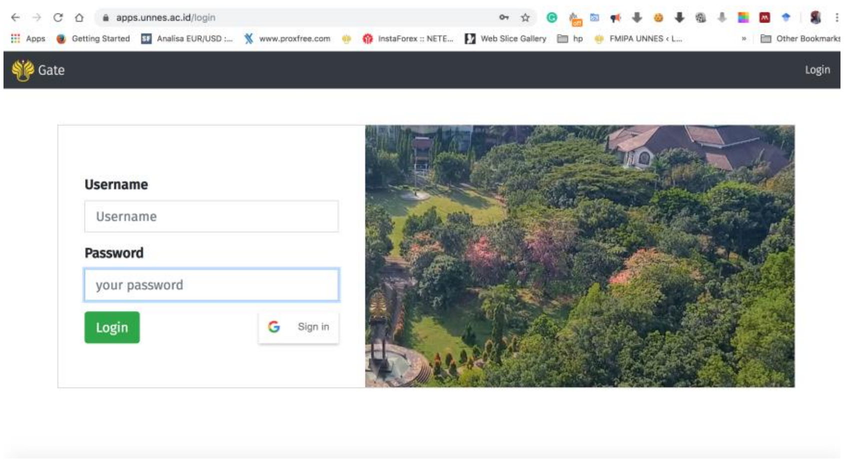


Figure 8. Login page.

Figure 8 is the initial SSO login page display. The Google login form will appear when the Sign-in button clicked, as shown in Figure 9.

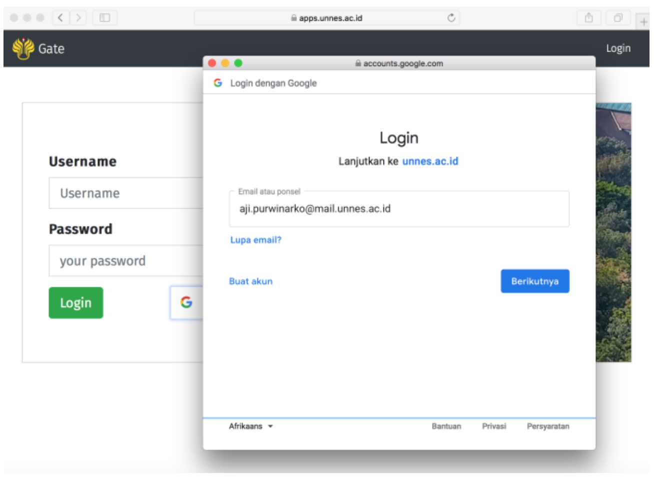


Figure 9. Google login form.

After Google successfully authenticates a user's account, the user page will redirect to an SSO home page (Figure 10). Users do not need a login to enter into each of the existing applications.

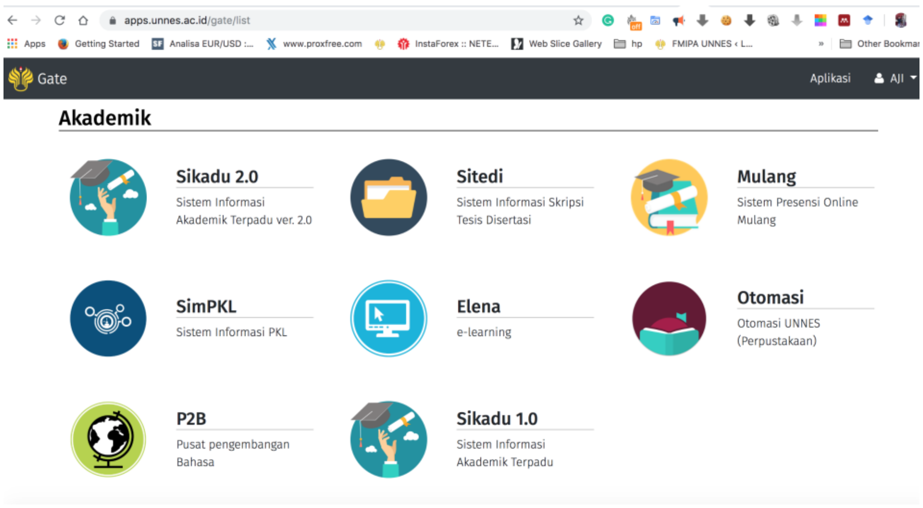
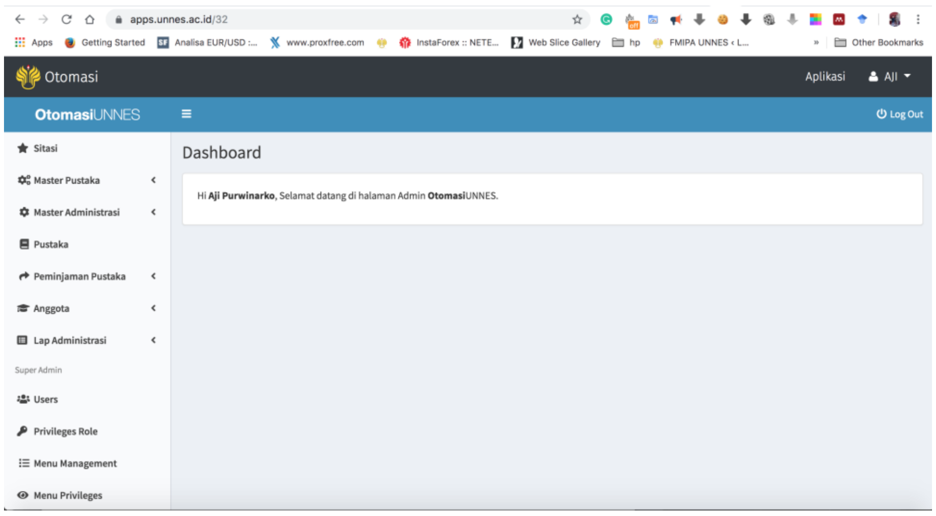

apps.unnes.ac.id/gate/list
Gate
Aplikasi
All
Akademik
Sikadu 2.0
Sistem Informasi Akademik Terpadu ver. 2.0
Sitedi
Sistem Informasi Skripsi Tesis Disertasi
Mulang
Sistem Presensi Online Mulang
SimPKL
Sistem Informasi PKL
Elena
e-learning
Otomasi
Otomasi UNNES (Perpustakaan)
P2B
Pusat pengembangan Bahasa
Sikadu 1.0
Sistem Informasi Akademik Terpadu

Figure 10. SSO home page

Figure 11 is the display when successfully entered into the library automation system.



Hi **Aji Purwinarko**, Selamat datang di halaman Admin **Otomasi**UNNES.

Figure 11. Library automation system dashboard.

4. Conclusion

By implementing Google Oauth 2 on this library system, users do not need to remember many user names and passwords. Users can sign-in using email to the SSO page, and if Google successfully authenticates, the user can open all applications. SSO makes it more efficient and effective because one login is enough to be able to access all applications.

Reference

[1] Purwinarko A & Sukestiyarno Y L 2014 *Sci. J. Inf.* **1**(2) 177

[2] Ramamoorthi L & Sarkar D 2020 Single Sign-on Implementation: Leveraging Browser Storage for Handling Tabbed Browsing Sign-outs. In *Developments and Advances in Defense and Security* 15 Springer, Singapore

[3] Olden, Eric, Darren C. Platt, Coby Royer, Keshava Berg, and Joseph H. Wallingford III 2015 *U.S. Patent 8,990,911* (Washington DC: U.S. Patent and Trademark Office)

[4] Rastogi, V., & Agrawal, A. (2015 All your Google and Facebook logins are belong to us: A case for single sign-off. In *2015 Eighth International Conference on Contemporary Computing*

(IC3) 416 IEEE.

[5] Akula N S & Chatoth V P 2016 *U.S. Patent No. 9,413,750* (Washington DC: U.S. Patent and Trademark Office)

[6] Sciarretta G, Armando A, Carbone R, & Ranise S 2016 Security of Mobile Single Sign-On: A Rational Reconstruction of Facebook Login Solution. In *SECRYPT* 147

[7] Kumar B, Abhishek K, Singh M P, & Kumar A 2015 An Improved Single Sign-On Mechanism by Enhancing the Functionality of Reverse Proxy. In *Emerging Research in Computing, Information, Communication and Applications* 77 Springer, New Delhi.

[8] Ivanova A I & Vodanovich S 2017 Single sign-on taxonomy. In *2017 IEEE 21st International Conference on Computer Supported Cooperative Work in Design (CSCWD)* 151 IEEE

[9] Lazarev S A, Demidov A V, Volkov V N, Stychuk A A, & Polovinkin D A 2016 Analysis of applicability of open single sign-on protocols in distributed information-computing environment. In *2016 IEEE 10th International Conference on Application of Information and Communication Technologies (AICT)* 1 IEEE

[10] Wijayarathna C & Arachchilage N A 2019 An Empirical Usability Analysis of the Google Authentication API. In *Proceedings of the Evaluation and Assessment on Software Engineering* 268 ACM

[11] Yang R, Lau W C, & Liu T 2016 Signing into one billion mobile app accounts effortlessly with oauth 2.0. *Blackhat Europe*