Software Security - Assignment 2

OAuth 2.0 Web Application Report

SLIIT | Sri Lanka Institute of Information Technology

Prepared by:

U.S. Thiwanka | MS20919832

H.A.I. Priyashantha | MS20919382

Table of Contents

Table of Figure

1 Introduction

This application initiated because the university student misses their Student Association meetings. As we noticed the need for some applications to keep the scheduled meetings remembering programs, that may be important for students. The problem is sometimes the Association cannot find the relevant group to inform the meeting. Hence, came up with the SAMeet app (Student Association Meetings App). When using this app students can simply log on to that web app using their Google account and add the reminder to their calendar. After that Google calendar will remind the scheduled meeting the students. The backbone of this application is the OAuth used, to authorize the app, through the student's Google account provided by the university or school, to add reminders in the Google calendar.

A framework that lends applications, permission to access some user accounts of preference on an HTTP facility, is basically what OAuth 2.0 is. It normally delivers flows of authorization for mobiles & desktop applications. The facility that hosts the user account, is assigned to the users by OAuth, & also the authorization of 3rd party applications to approach it. The app was basically made using, PHP, JS, HTML & CSS. This report will focus on how the process and its arrangements were made, Furthermore, all the PHP codes used will be attached.

Basically, the implemented application utilizes the services of OAuth Authorization Servers and OAuth Resource Servers, sends requests to the OAuth authorization server to obtain the “access token”, which during the flow, will stimulate user authentication, & finally, after the “OAuth access token” is acknowledged, invokes the resource server APIs to gain the protected resources or perform the particular action. This whole process is customized and elaborated to explain the use case that we used.

2 Implementation

2.1 SAMeet Application

SAMeet app was created using HTML, CSS, PHP & JS (all codes/ project can be found in GitHub https://github.com/ThiwankaUmagiliya/OAuth-2.0-with-Google-Calendar : Check APPENDIX), to be finally hosted in the Apache webserver – localhost. A webserver was used in this scenario because the SAMeet app uses an “Authorization Code” grant type. The above-mentioned, OAuth was then used to proceed with the authorization process. But before going any further, the roles, in this process, the actors, & the interactions among them should be described.

2.2 OAuth Roles

- Resource owner: User/ University Students → The students who authorize SAMeet to access their user account (Corporate Google Account), using their user credentials. A read & write access is given SAMeet.
- Client: SAMeet app → The user's particular account, is wished to be accessed by the SAMeet app. But before that, students handling this must authorize it to do so, & API should validate the authorization.

- Resource Server: Google calendar → The protected user accounts are hosted by Google calendar.
- Authorization server: Google → The user identity is verified using Google and then issues access tokens to SAMeet.

2.3 OAuth roles interaction & Abstract flow

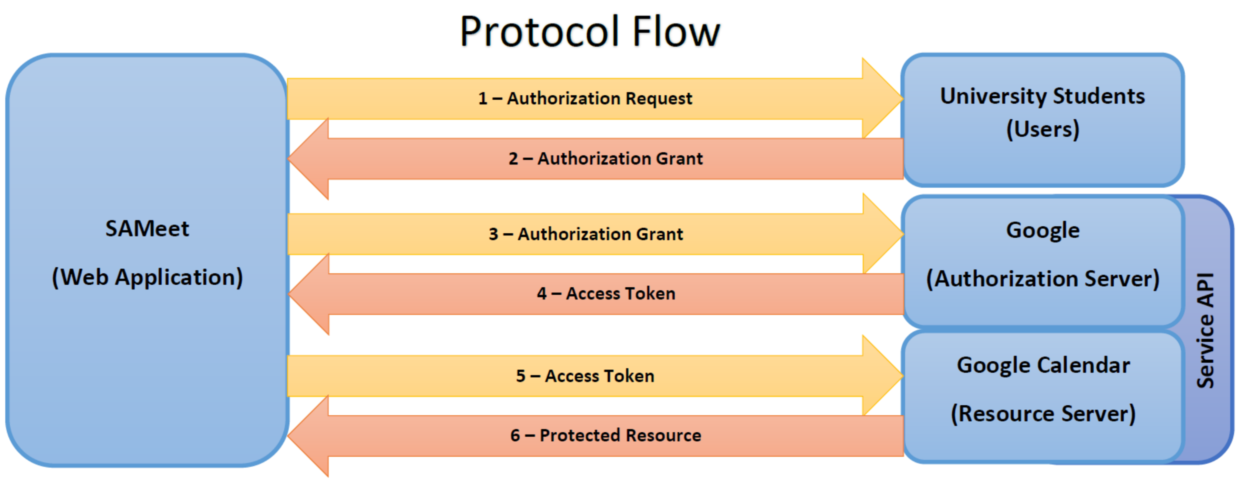


Figure 1 Abstract Protocol Flow

The used Protocol Flow can be discussed completely as done below.

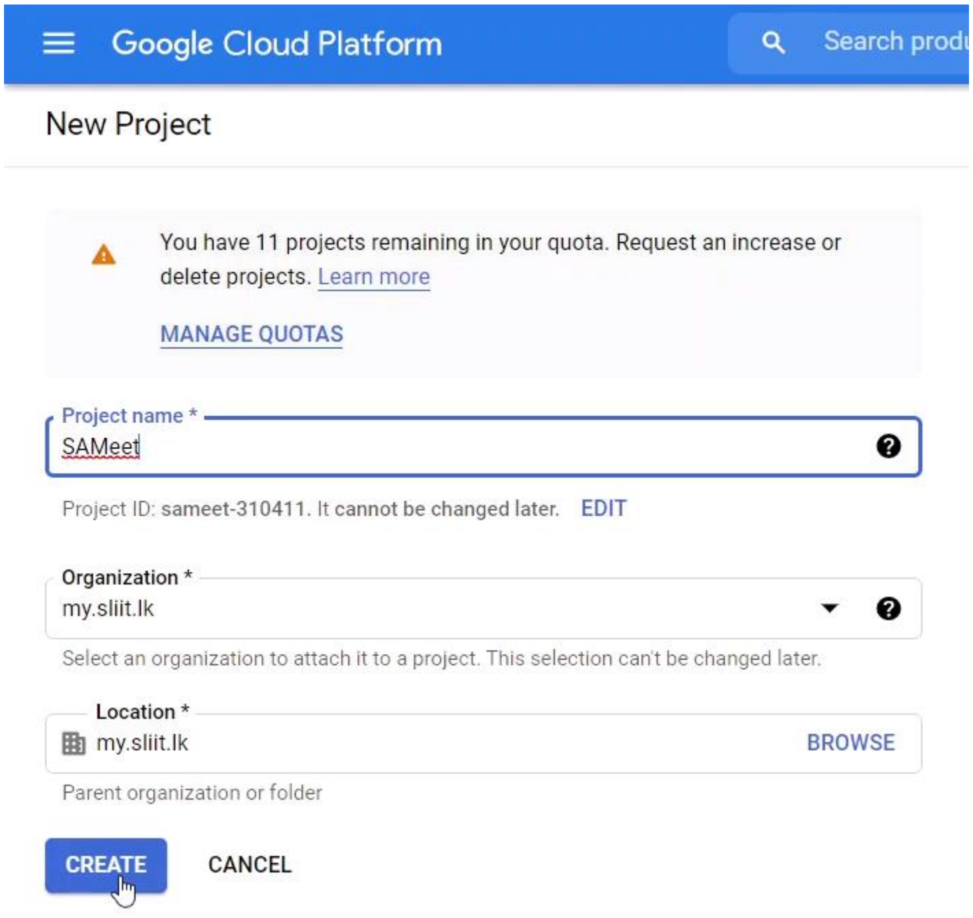
1) The SAMeet app requests from the students, authorization to access service resources.
2) When the students request authorization, the SAMeet app receives an authorization grant.
3) SAMeet app requests an access token from Google by presenting authentication of its own identity, and the authorization grant.
4) If the SAMeet app identity is authenticated and the authorization grant is valid, Google issues an access token to the SAMeet app. Authorization is complete.
5) SAMeet app then requests the resource from Google Calendar and presents the access token for authentication.
6) Upon the validation of the access token, Google Calendar serves as the resource to the SAMeet app. Which is to add reminders in the calendar.

2.4 Project Creation

This section would give the reader, a stepwise idea, of how the project was created.

2.4.1 Application Registration

First of all, the SAMeet app should be registered with the service by creating a new project using the Google developer dashboard.



New Project

You have 11 projects remaining in your quota. Request an increase or delete projects. Learn more

MANAGE QUOTAS

Project name *
SAMeet

Project ID: sameet-310411. It cannot be changed later. EDIT

Organization *
my.sliit.lk

Select an organization to attach it to a project. This selection can't be changed later.

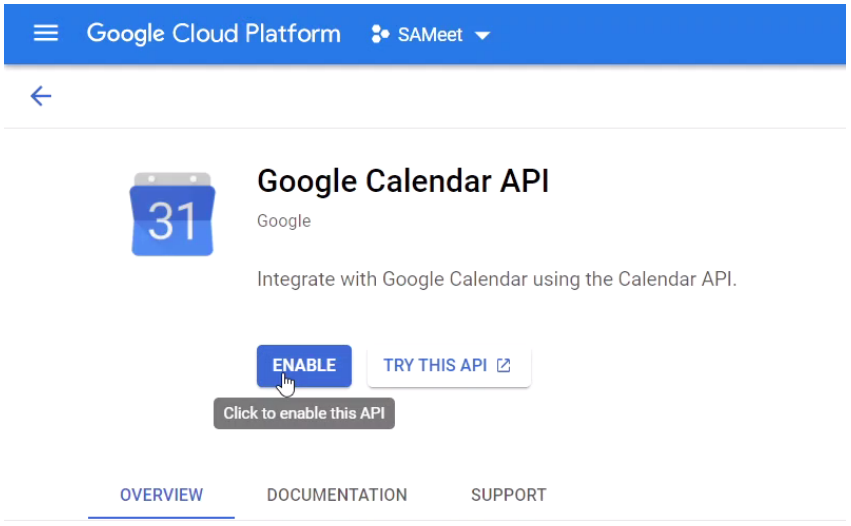
Location *
my.sliit.lk BROWSE

Parent organization or folder

Figure 2 Creating a new project in the Google developer dashboard

2.4.2 Google Calendar API

After the SAMeet project was created, Google Calendar API was added from the API library.



Google Calendar API

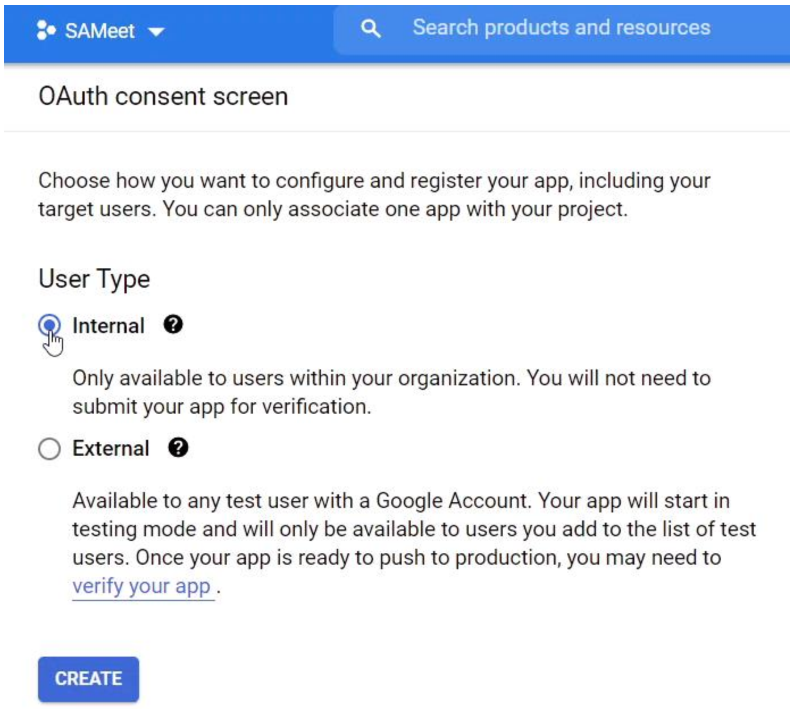
Google

Integrate with Google Calendar using the Calendar API.

Figure 3 Enabling Google Calendar API

2.4.3 OAuth Consent

This enabled us to tell the application, what type of data, is to be accessed from the student, in this scenario, just "Internal" (so that only SLIIT accounts are accessed).



OAuth consent screen

Choose how you want to configure and register your app, including your target users. You can only associate one app with your project.

User Type

[✓ Internal]

Only available to users within your organization. You will not need to submit your app for verification.

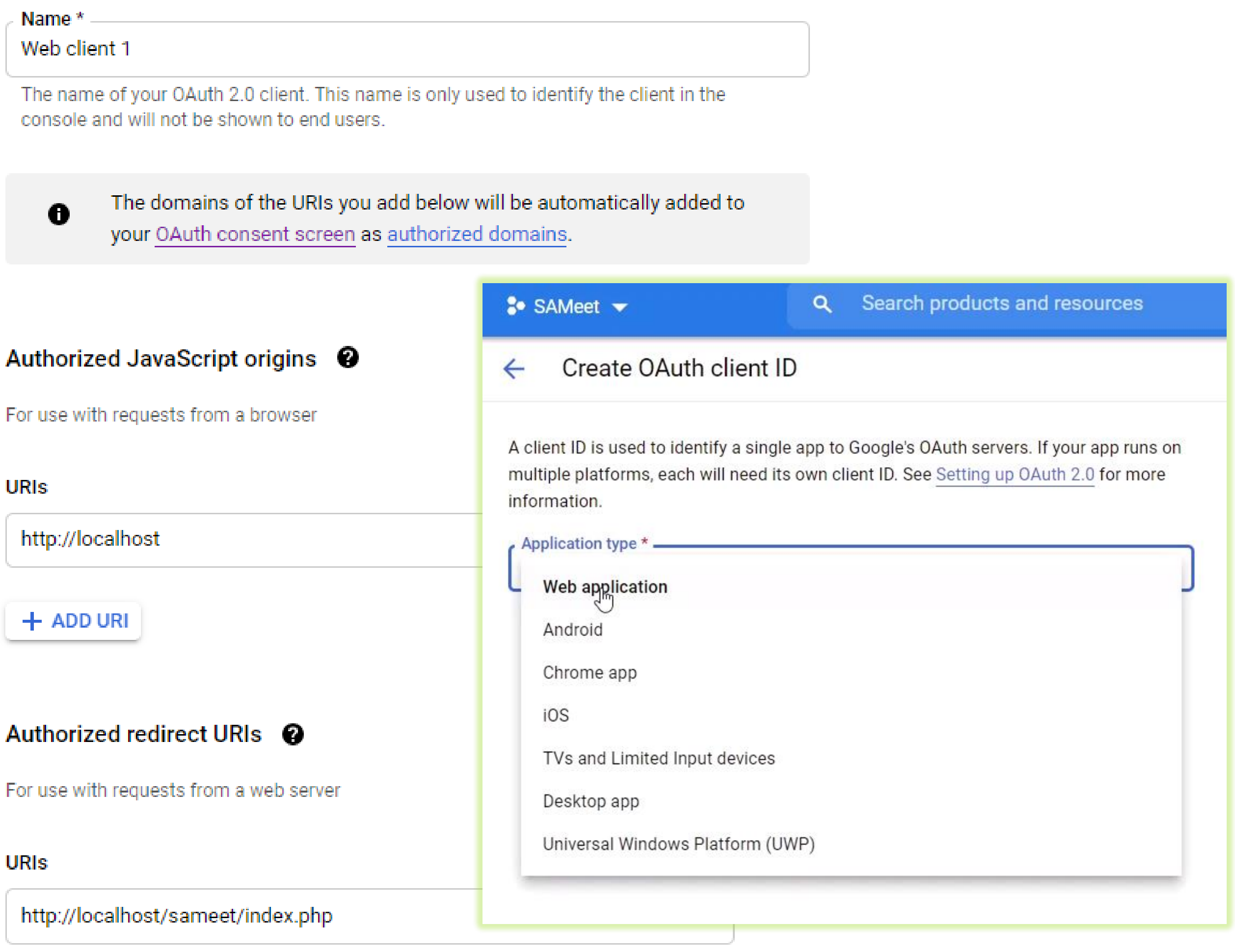
External

Available to any test user with a Google Account. Your app will start in testing mode and will only be available to users you add to the list of test users. Once your app is ready to push to production, you may need to verify your app .

Figure 4 Consent Screen - User Type

2.4.4 OAuth Client ID

Then, the following form was filled with application type and redirection details. The "redirect URI" is anywhere the service will "redirect" afterward they authorize SAMeet and therefore the handling part of authorization codes or access tokens part of the SAMeet app will be done by it.



Name *
Web client 1
The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

The domains of the URIs you add below will be automatically added to your OAuth consent screen as authorized domains.

Authorized JavaScript origins
For use with requests from a browser
URIs
http://localhost

Authorized redirect URIs
For use with requests from a web server
URIs
http://localhost/sameet/index.php

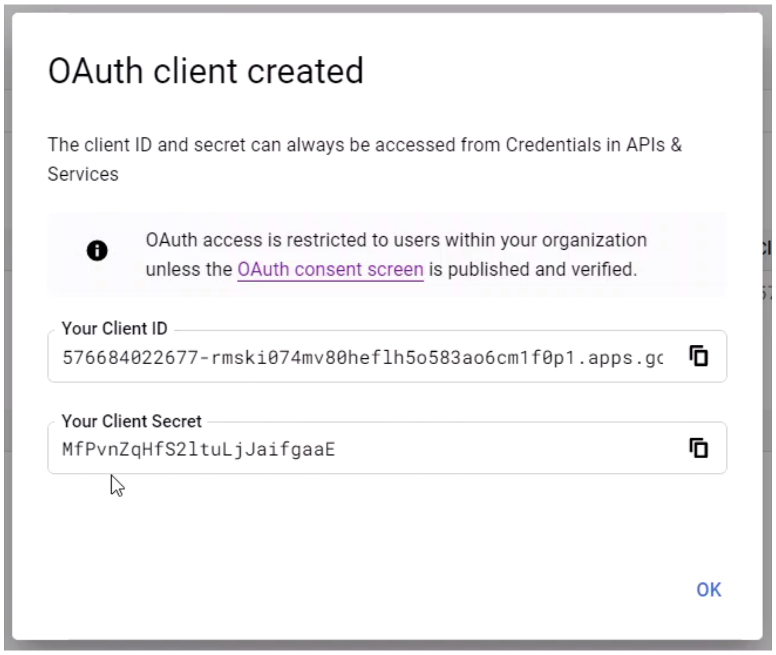
Create OAuth client ID
A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See Setting up OAuth 2.0 for more information.
Application type *

SAMeet
Search products and resources
+ ADD URI
Web application
Android
Chrome app
iOS
TVs and Limited Input devices
Desktop app
Universal Windows Platform (UWP)

Figure 5 Creating OAuth Client ID

2.5 Client ID and Client Secret

After the SAMeet app is enumerated, a "client identifier" and a "client secret" will be produced by the service. The Client ID: To identify the SAMeet app, and to build authorization URLs that are presented to the students, a string, better a "publicly exposed string" that is used by the service API. The Client Secret: Consumed to authenticate the identity of the SAMeet app to the service API when it requests to access the SLIIT account & is kept private between the SAMeet app and the API.



OAuth client created

The client ID and secret can always be accessed from Credentials in APIs & Services

OAuth access is restricted to users within your organization unless the OAuth consent screen is published and verified.

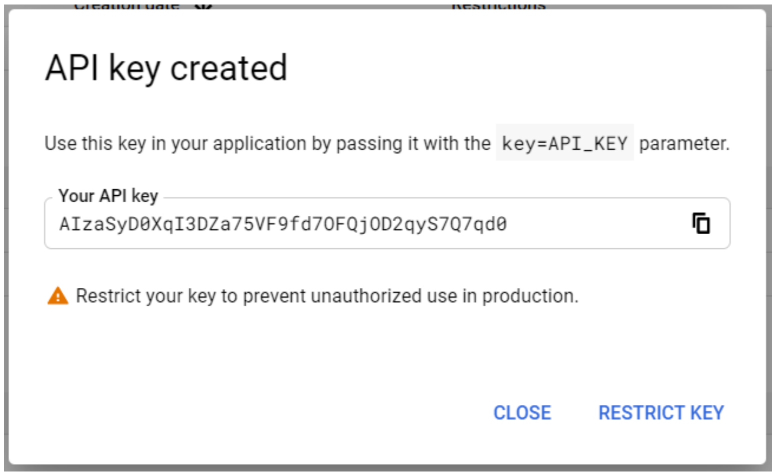
Your Client ID
576684022677-rmski074mv80heflh5o583ao6cm1f0p1.apps.gc

Your Client Secret
MfPvnZqHfS2ltuLjJaifgaaE

Figure 6 Client ID and Client Secret

2.6 API Key

This used to uniquely identify the SAMeet app.



API key created

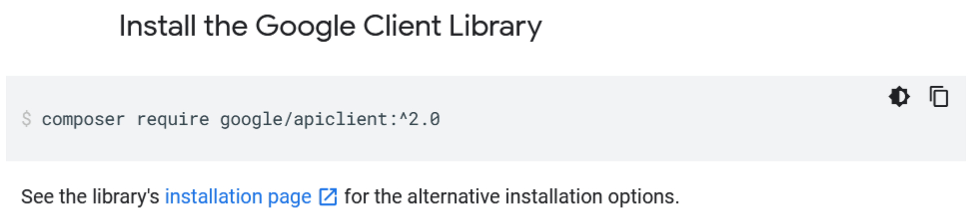
Use this key in your application by passing it with the `key=API_KEY` parameter.

Your API key
AIzaSyD0XqI3DZa75VF9fd7OFQjOD2qyS7Q7qd0

Restrict your key to prevent unauthorized use in production.

Figure 7 API Key

2.7 Installment of Google Client Library



Install the Google Client Library

```
$ composer require google/apiclient:^2.0
```

See the library's installation page for the alternative installation options.

Figure 8 Install the Google Client Library using the command

2.8 PHP coding

PHP is used for the implementation of the authorization process and event creating process. This code is further included in GitHub.

(https://github.com/ThiwankaUmagiliya/OAuth-2.0-with-Google-Calendar : Check APPENDIX)

3 How system operates

3.1 Authorization Grant

The authorization grant type, or "How the authorization is given to this application", for this use case is an Authorization Code.

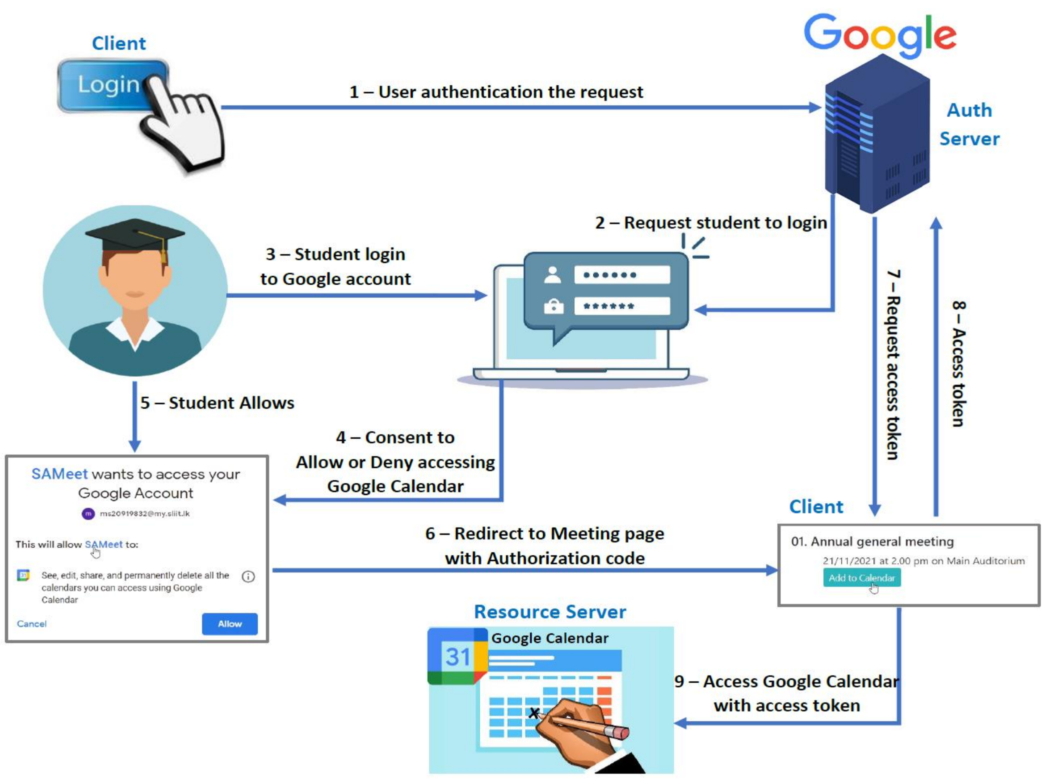

Client
Login
Google
1 – User authentication the request
Auth Server
2 – Request student to login
3 – Student login to Google account
7 – Request access token
8 – Access token
5 – Student Allows
4 – Consent to Allow or Deny accessing Google Calendar
SAMeet wants to access your Google Account
ms20919832@my.sliit.lk
This will allow SAMeet to:
See, edit, share, and permanently delete all the calendars you can access using Google Calendar
Cancel
Allow
6 – Redirect to Meeting page with Authorization code
Client
01. Annual general meeting
21/11/2021 at 2.00 pm on Main Auditorium
Add to Calendar
Resource Server
Google Calendar
31
9 – Access Google Calendar with access token

Figure 9 Authentication code message flow

3.2 Message flows and grants explained

The use case of this scenario, considering all the message flows and descriptions are stated below, stepwise in order.

Step – 1: The students will access the app, which is hosted on the web, and should have to sign into the service using the university-provided Google account (Other Gmail accounts haven't the login capability).

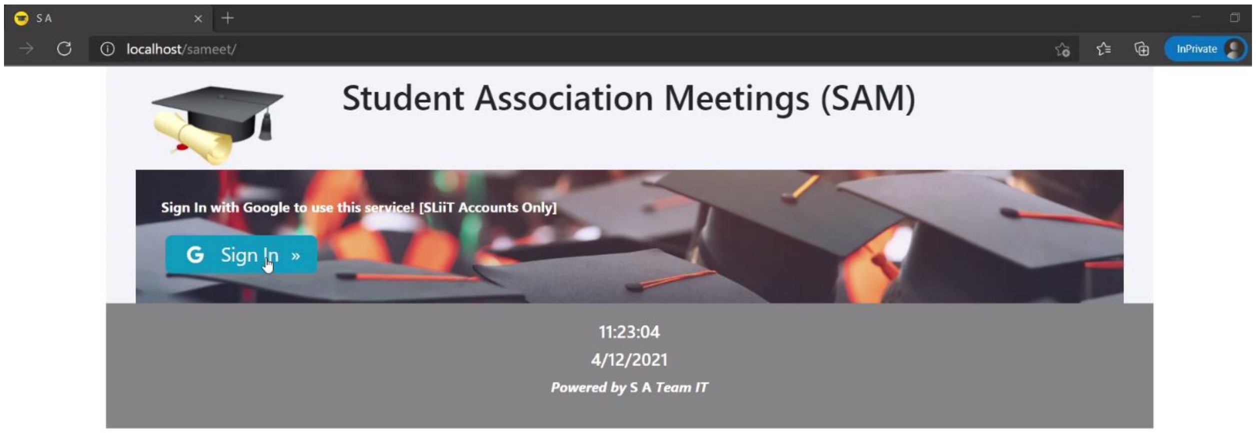

SA
localhost/sameet/
InPrivate
Student Association Meetings (SAM)
Sign In with Google to use this service! [SLiiT Accounts Only]
Sign In »
11:23:04
4/12/2021
Powered by S A Team IT

Figure 10 Home Page

Step – 2 & 3: If the student is not logged in, then they will have to sign in using the university-provided email.

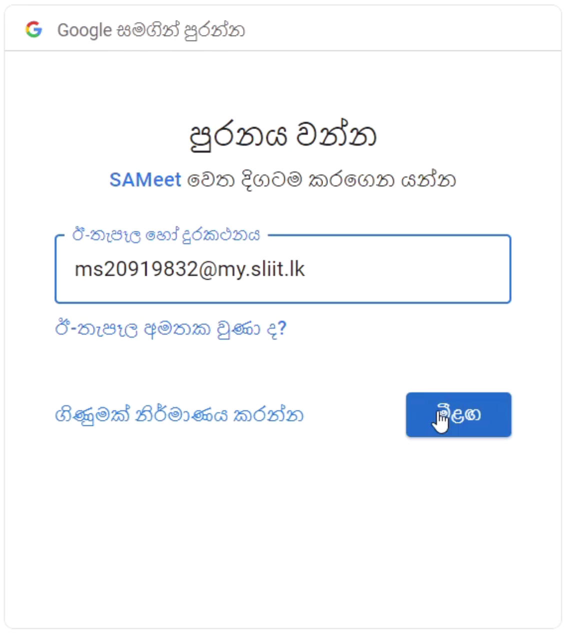
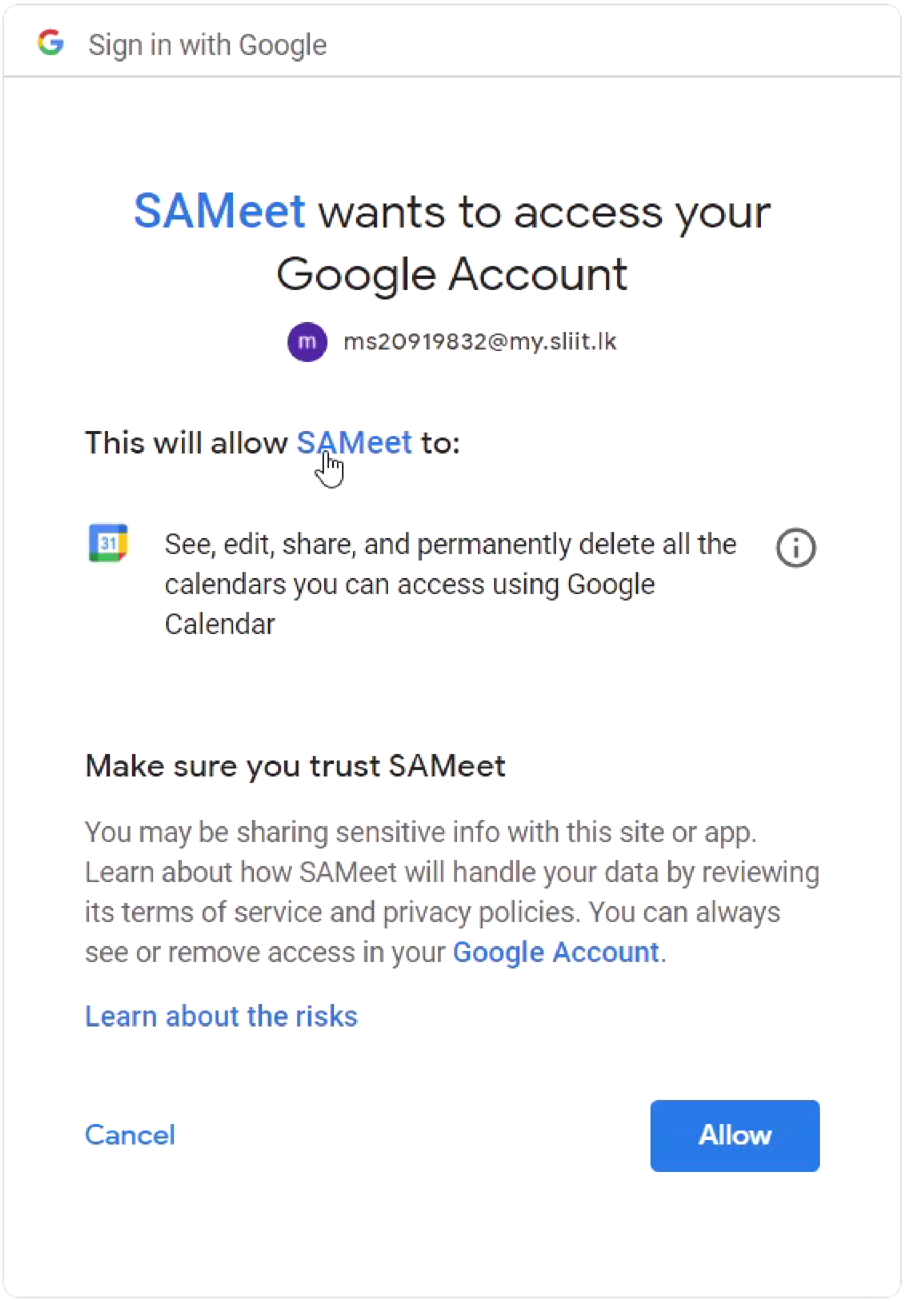


Figure 11 Google authorization

Step – 4 & 5: After login to Google account, they will pop up the consent screen to allow or deny the Google calendar accessing the system. To access the system user must allow the request.



Sign in with Google

SAMeet wants to access your Google Account

ms20919832@my.sliit.lk

This will allow SAMeet to:

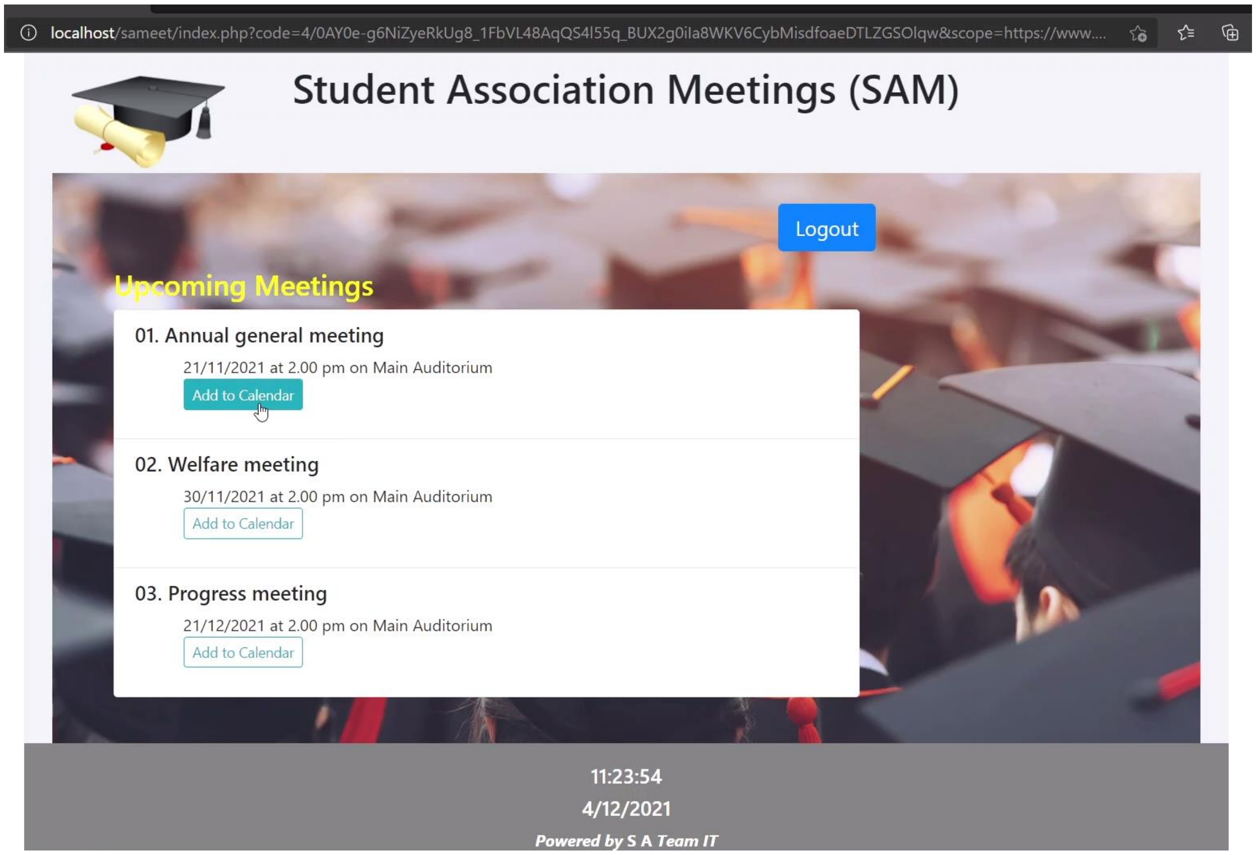
See, edit, share, and permanently delete all the calendars you can access using Google Calendar

Make sure you trust SAMeet

You may be sharing sensitive info with this site or app. Learn about how SAMeet will handle your data by reviewing its terms of service and privacy policies. You can always see or remove access in your Google Account.

Figure 12 Google consent screen

Step – 6: When the student clicks on allow, the service redirects the web browser to the app redirect URI, which was detailed at the client registration, along with an authorization code. The redirect page looks like as below.



localhost/sameet/index.php?code=4/0AY0e-g6NiZyeRkUg8_1FbVL48AqQS4l55q_BUX2g0ila8WKV6CybMisdfoaeDTLZGSOlqw&scope=https://www....

Student Association Meetings (SAM)

Logout

Upcoming Meetings

01. Annual general meeting
21/11/2021 at 2.00 pm on Main Auditorium
Add to Calendar

02. Welfare meeting
30/11/2021 at 2.00 pm on Main Auditorium
Add to Calendar

03. Progress meeting
21/12/2021 at 2.00 pm on Main Auditorium
Add to Calendar

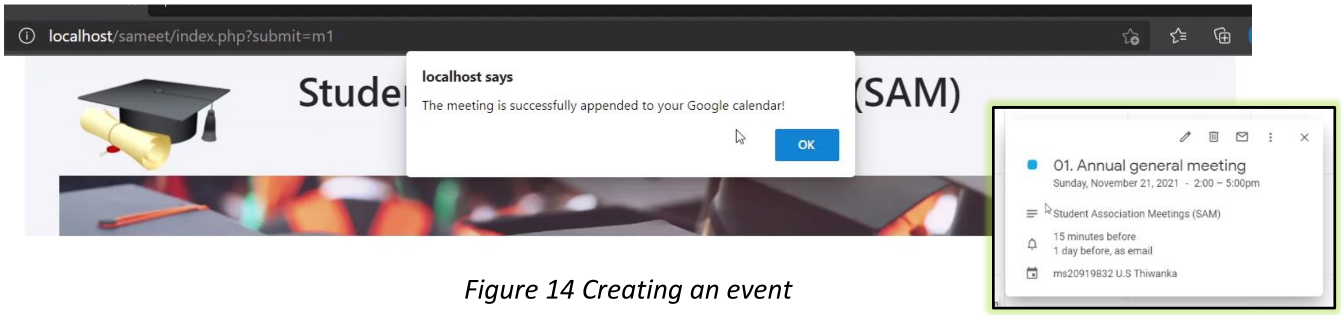
11:23:54
4/12/2021
Powered by S A Team IT

Figure 13 Redirecting page

Step – 7: The app then requests an access token from the API, sending the authorization code along with all the authentication details, which contains the client secret, to the endpoint token of the API.

Step – 8: When the authorization is valid, the API will send an answer with the access token, a refresh token, optionally, to the application.

Step – 9: When the student clicks on "Add to calendar", the app will take the access token to the resource server – Google calendar and get permission to create the event.



localhost/sameet/index.php?submit=m1

Stude (SAM)

localhost says
The meeting is successfully appended to your Google calendar!
OK

01. Annual general meeting
Sunday, November 21, 2021 · 2:00 – 5:00pm
Student Association Meetings (SAM)
15 minutes before
1 day before, as email
ms20919832 U.S Thiwanka

Figure 14 Creating an event

The application authorization process is completed. The token is used in accessing the user's account through the service API, in limitation of scope to access, until the token is modified, meaning its expiration or revoking. A refresh token exists, & could be used to demand new access tokens, after the expiration of the original.

This app contains an access token, & it can be used to access the student's SLIIT accounts using API, in the limitation of the scope to access, until the token is modified, meaning its expiration or revoking. If the access token is invalid, the API will show an error.

4 Conclusion

SAMeet was implemented by consuming services of an OAuth Authorization Server and an OAuth Resource Server, attaining the access token, prompts user authentication, along with the flow, & after receiving the OAuth access token, invokes the resource server APIs and gain the protected perform to add reminders in the calendar. The process, of implementation, was indeed a success, keeping aside some features, that will need upgrading (mentioned in the recommendation).

5 Appendix A

5.1 index.php

```
<!doctype html>
<?php
           $userTokenPath = 'token.json';

        if(isset($_POST['logout']) && file_exists($userTokenPath)) {
            unlink("token.json");
           }
?>
<html lang="en">
     <head>

    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="css/bootstrap.min.css">
    <link rel="stylesheet" href="css/tab.css">
     <link rel="icon" type="image/png" href="images/icon.png"/>
    <title>S A</title>

<script>
function startTime() {
  var today = new Date();
  var h = today.getHours();
  var m = today.getMinutes();
  var s = today.getSeconds();
  m = checkTime(m);
  s = checkTime(s);
  document.getElementById('timenow').innerHTML =
  h + ":" + m + ":" + s;
  var t = setTimeout(startTime, 500);
}
function checkTime(i) {
  if (i < 10) {i = "0" + i};  // add zero in front of numbers < 10
  return i;
}
</script>

     </head>
<body style="margin:0 10%;padding:0 10px;" onload="startTime()">

     <div class="jumbotron jumbotron-fluid" style="padding-
bottom:0px;padding-top:10px;margin-bottom:0px;">
           <div class="container">
```

```
          <div class="row">
               <div class="col-sm-2">
                    <a href="index.php"><img class="img-fluid"
src="images/logo.png" /></a>
               </div>
               <div class="col-sm-8">
                    <h1 class="text-center">Student Association
Meetings (SAM)</h1>
               </div>
               <div class="col-sm-2">
               </div>
          </div>
          </div>



     <div class="container" style="padding:30px; background-image:
url(images/bg.jpg);background-position: center;background-repeat: no-
repeat;background-size: cover;">

     <div class="row">

          <div class="col-sm-9">
           <!-- check for the user login-->
             <?php include 'connection.php';?>

          </div>
        <div class="col-sm-3">
        </div>

     </div>
     </div>

    <!-- footer -->

          <div class="text-center" style="background-
color:#808080;color:#ffffff;padding:20px 0;">
               <!--display date and time-->

               <h5 id="timenow"></h5>
                    <h5 id="date"></h5>
                    <script>
               var dt = new Date();
               document.getElementById("date").innerHTML =
dt.toLocaleDateString();
               </script>

                    <p><b><i>Powered by </i>S A<i> Team
IT</i></b></p>
          </div>

 </div>
```

```
    <!-- Optional JavaScript -->
    <!-- jQuery first, then Popper.js, then Bootstrap JS -->
    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"
integrity="sha384-
q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous"></script>
    <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"
></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/poppe
r.min.js" integrity="sha384-
UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous"></script>
    <script src="js/bootstrap.min.js"></script>

 </body>


</html>
```

5.2 connection.php

```
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">
<style>

.button {
  display: inline-block;
  border-radius: 8px;
  background-color: #008CBA;
  border: none;
  color: #FFFFFF;
  text-align: center;
  font-size: 22px;
  padding: 5px 12px;
  width: 175px;
  transition: all 0.5s;
  cursor: pointer;
  margin: 5px;
}

.button span {
  cursor: pointer;
  display: inline-block;
  position: relative;
  transition: 0.5s;
}

.button span:after {
  content: '\00bb';
  position: absolute;
  opacity: 0;
  top: 0;
  right: -20px;
  transition: 0.5s;
}

.button:hover span {
  padding-right: 25px;
}

.button:hover span:after {
  opacity: 1;
  right: 0;
}

</style>
<?php
require __DIR__ . '/vendor/autoload.php';

/**
 * Returns an authorized API client.
 * @return Google_Client the authorized client object
 */
function getClient()
{
    $gClient = new Google_Client();
    $gClient->setApplicationName('Test Project 01');
    $gClient->setScopes(Google_Service_Calendar::CALENDAR);
    $gClient->setAuthConfig('client_secret.json');
    $gClient->setAccessType('offline');
    $gClient->setPrompt('select_account consent');

    // Load previously authorized token from a file, if it exists.
    // The file token.json stores the user's access and refresh
tokens, and is
    // created automatically when the authorization flow completes for
the first
    // time.
    $userTokenPath = 'token.json';
    if (file_exists($userTokenPath)) {
        $appToken = json_decode(file_get_contents($userTokenPath),
true);
        $gClient->setAccessToken($appToken);
    }

    // If there is no previous token or it's expired.
    if ($gClient->isAccessTokenExpired()) {
        // Refresh the token if possible, else fetch a new one.
        if ($gClient->getRefreshToken()) {
            $gClient->fetchAccessTokenWithRefreshToken($gClient-
>getRefreshToken());
        } else {
            // Request authorization from the user.
            if(!browserCredentials()) {
                $authUrl = $gClient->createAuthUrl();
                return "<p style='color:white;'><b>Sign In with Google
to use this service! [SLiiT Accounts Only]</b></p><a
href='$authUrl'><button class='button' style='vertical-
align:middle'><i class='fa fa-google' style='padding:0px 20px 5px
5px'></i><span> Sign In </span></button></a>";
            }
            $authCode = $_GET['code'];

            // Exchange authorization code for an access token.
            $appToken = $gClient-
>fetchAccessTokenWithAuthCode($authCode);
            $gClient->setAccessToken($appToken);

            // Check to see if there was an error.
            if (array_key_exists('error', $appToken)) {
                throw new Exception(join(', ', $appToken));
            }
        }
        // Save the token to a file.
        if (!file_exists(dirname($userTokenPath))) {
            mkdir(dirname($userTokenPath), 0700, true);
        }
        file_put_contents($userTokenPath, json_encode($gClient-
>getAccessToken()));
    }
    return $gClient;
}

//check browser url has the auth code
function browserCredentials() {
    if(isset($_GET['code'])) return true;

    return false;
}

//get authorized API client
$gClient = getClient();

if(! is_a ($gClient, "Google_Client")) {
        echo $gClient;
}
else {
            //dispaly the content of the page
           include 'content.php'   ;
    }

?>
```

5.3 content.php

```
<div class="container">

                <div class="text-right">
                     <form method="post" action="<?php echo
$_SERVER['PHP_SELF']; ?>">
                     <button id="logout" name="logout" type="submit"
class="btn btn-primary btn-lg">Logout</button>
                     </form>
                </div>

                <div class="tab-pane active p-3" id="l1"
role="tabpanel">
                     <h3 style='color:#ffff00;'>Upcoming Meetings</h3>
                          <ul class="list-group">

                               <li class="list-group-item">
                                    <h5 class="list-group-item-
heading">01. Annual general meeting</h5>
                                    <p class="list-group-item-text
pl-5">21/11/2021 at 2.00 pm on Main Auditorium<br>
                                    <button id="submit"
name="submit" type="button" class="btn btn-outline-info btn-sm"
onclick='location.href="?submit=m1"'>Add to Calendar</button>
                                    </p>
                               </li>

                               <li class="list-group-item">
                                    <h5 class="list-group-item-
heading">02. Welfare meeting</h5>
                                    <p class="list-group-item-text
pl-5">30/11/2021 at 2.00 pm on Main Auditorium<br>
                                    <button id="submit"
name="submit" type="button" class="btn btn-outline-info btn-sm"
onclick='location.href="?submit=m2"'>Add to Calendar</button>
                                    </p>
                               </li>

                               <li class="list-group-item">
                                    <h5 class="list-group-item-
heading">03. Progress meeting</h5>
                                    <p class="list-group-item-text
pl-5">21/12/2021 at 2.00 pm on Main Auditorium<br>
                                    <button id="submit"
name="submit" type="button" class="btn btn-outline-info btn-sm"
onclick='location.href="?submit=m3"'>Add to Calendar</button>
```

```
                                    </p>
                               </li>

                          </ul>
                 </div>

</div>

<?php
if($_GET){
    include 'schedules.php';
    if(isset($_GET['submit'])){

       $request = $_GET['submit'];
        switch ($request) {
            case 'm1':
                m1();
                break;
            case 'm2':
                m2();
                break;
            case 'm3':
                m3();
                break;
            default:
                echo "Error!";
        }
    }
}
?>
```

5.4 schedules.php

```
<?php
function m1(){
    $summary = '01. Annual general meeting';
    $description = 'Student Association Meetings (SAM)';
    $start = '2021-11-21T14:00:00+05:30';
    $end = '2021-11-21T17:00:00+05:30';
    include 'calendar.php';
}

function m2(){
    $summary = '02. Welfare meeting';
    $description = 'Student Association Meetings (SAM)';
    $start = '2021-11-30T14:00:00+05:30';
    $end = '2021-11-30T16:00:00+05:30';
    include 'calendar.php';
}

function m3(){
    $summary = '03. Progress meeting';
    $description = 'Student Association Meetings (SAM)';
    $start = '2021-12-21T14:00:00+05:30';
    $end = '2021-12-21T16:00:00+05:30';
    include 'calendar.php';
}
?>
```

5.5 calendar.php

```
<?php
require __DIR__ . '/vendor/autoload.php';


$gClient = getClient();
$gService = new Google_Service_Calendar($gClient);
$gEvent = new Google_Service_Calendar_Event(array(
          'summary' => $summary ,
          'description' => $description ,
          'start' => array(
            'dateTime' => $start ,
            'timeZone' => 'Asia/Colombo',
          ),
          'end' => array(
            'dateTime' => $end ,
            'timeZone' => 'Asia/Colombo',
          ),


          'reminders' => array(
            'useDefault' => FALSE,
            'overrides' => array(
              array('method' => 'email', 'minutes' => 24 * 60),
              array('method' => 'popup', 'minutes' => 15),
            ),
          ),
));

        $gCalendarId = 'primary';
        $gEvent = $gService->events->insert($gCalendarId, $gEvent);
        echo '<script type="text/javascript">';
        echo ' alert("The meeting is successfully appended to your
Google calendar!");';
        echo 'window.location.href =
"http://localhost/sameet/index.php";';
        echo '</script>';

?>
```

6 References

[1] Google, “Google Calendar API.” https://developers.google.com/calendar (accessed Apr. 05, 2021).

[2] M. Anicas, “An Introduction to OAuth 2,” 2014. https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2 (accessed Mar. 30, 2021).

[3] M. Raible, “What the Heck is OAuth?,” 2017. https://developer.okta.com/blog/2017/06/21/what-the-heck-is-oauth (accessed Mar. 22, 2021).

[4] N. Hossain, M. A. Hossain, M. Z. Hossain, M. H. I. Sohag, and S. Rahman, “OAuth-SSO: A Framework to Secure the OAuth-Based SSO Service for Packaged Web Applications,” *Proceedings - 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications and 12th IEEE International Conference on Big Data Science and Engineering, Trustcom/BigDataSE 2018*, pp. 1575–1578, 2018, doi: 10.1109/TrustCom/BigDataSE.2018.00227.

[5] K. Dodanduwa and I. Kaluthanthri, “Role of trust in oauth 2.0 and OpenID connect,” pp. 1–4, 2018.

[6] Google, “Using OAuth 2.0 for Web Server Applications,” 2021. https://developers.google.com/identity/protocols/oauth2/web-server (accessed Mar. 20, 2021).

[7] GitHub, “Authorizing OAuth Apps,” 2021. https://docs.github.com/en/developers/apps/authorizing-oauth-apps (accessed Apr. 05, 2021).