

Software Security – IE5042

Assignment 2 Report

Student ID: MS20900304
Name: E.M.P.S. Dehideniya
Course: Software Security - IE5042
Assignment: OAuth2 demonstration
Batch: MSc-CS - Jan 2020
Email: ms20900304@my.sliit.lk

Table of Contents

1	Assignment Objectives.....	1
2	Configure Authorization Server and Resource Server.....	2
2.1	Configure Authorization Server.....	2
2.1.1	Create a new project.....	2
2.1.2	Generate OAuth Client ID	4
2.2	Configure Resource Server.....	13
3	Deploy and Run the Web Application.....	15
3.1	Deploying the Web Application.....	15
3.1.1	Download the contents as a ZIP file:.....	15
3.1.2	Set up PHP environment in Ubuntu:.....	15
3.1.3	Configure PHP interpreter in PhpStorm:	16
3.1.4	Install Composer dependency manager:.....	16
3.1.5	Initialize Composer for the project:.....	16
3.1.6	Resolve Google API client dependencies using Composer:.....	16
3.1.7	Update composer.phar file:	16
3.2	Running the Web Application	16
4	Webpage Flow.....	17
4.1	Authorization Flow	17
4.1.1	login.php.....	17
4.1.2	index.php	17
4.1.3	oauth2callback.php.....	17
4.1.4	logout.php.....	18
4.1.5	style.css.....	18
4.2	Web pages	19
4.2.1	Access not granted from Google Permissions.....	19
4.2.2	Web Application Login page	19
4.2.3	OAuth2 redirect (Sinhala).....	19
4.2.4	OAuth2 redirect (English).....	21
4.2.5	Email address accepted.....	22
4.2.6	Password accepted, Google Drive permissions request.....	22
4.2.7	index.php loaded with file list (Callback completed)	23

4.2.8	Access Token Details.....	24
4.2.9	Logout page.....	25
4.2.10	Access granted on Google Permissions.....	26
4.2.11	List of files on Google Drive.....	26
References		27
Appendix - Artefacts		28
	/resources/client/client_secrets.json.....	28
	login.php.....	28
	/pages/index.php.....	29
	/pages/oauth2callback.php.....	32
	/pages/logout.php.....	33
	/resources/files/style.css.....	33

1 Assignment Objectives

During this assignment, following objectives were required to be met:

Objective 1:

Send a request to the OAuth authorization server for obtaining the access token. During the flow, it will prompt for user authentication (eg: google). **You may use any supported OAuth grant type (eg: authorization code, implicit etc.).**

Solution

Use the Google OAuth2 framework and use as the authorization server to send a request and obtain the access token upon prompting for user authentication, using authorization code grant type.

Objective 2:

Once the OAuth access token is received, invoke the resource server APIs and obtain the protected resources or perform the particular action.

Solution:

Use Google Drive as the resource server after receiving OAuth2 access token from the above activity. Obtain files and folders of the user and print the items as a table.

2 Configure Authorization Server and Resource Server

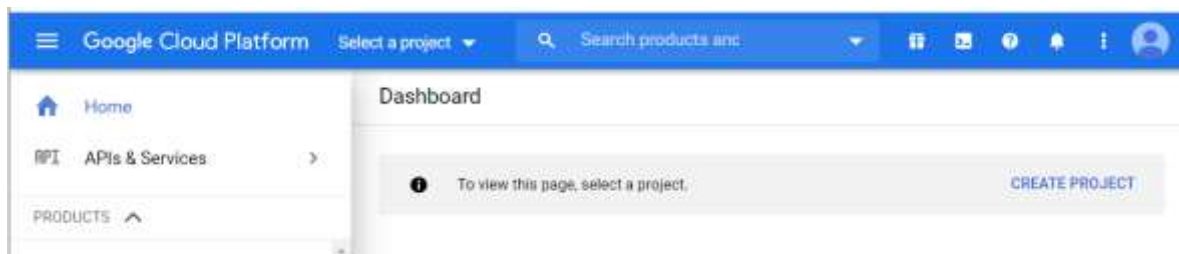
Before proceeding to deploy the web application, authorization server (Google OAuth2) and resource server (Google Drive) must be configured. Please refer the following steps:

2.1 Configure Authorization Server

2.1.1 Create a new project

Navigate to <https://console.developers.google.com/> [7]

If you have no projects created, a new project should be created:

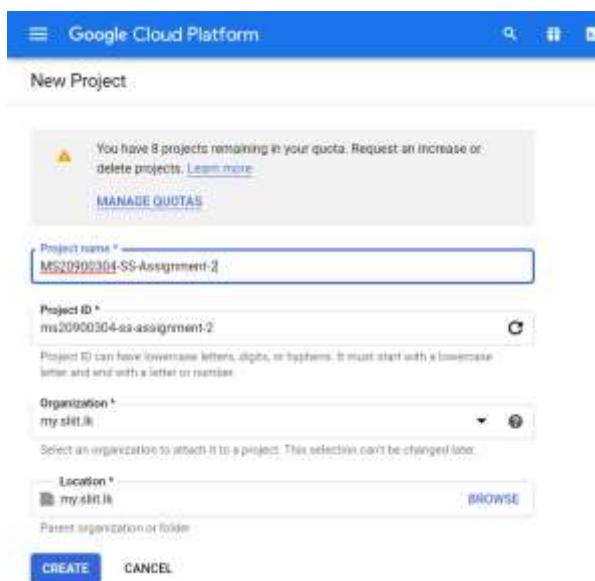


Click **CREATE PROJECT**

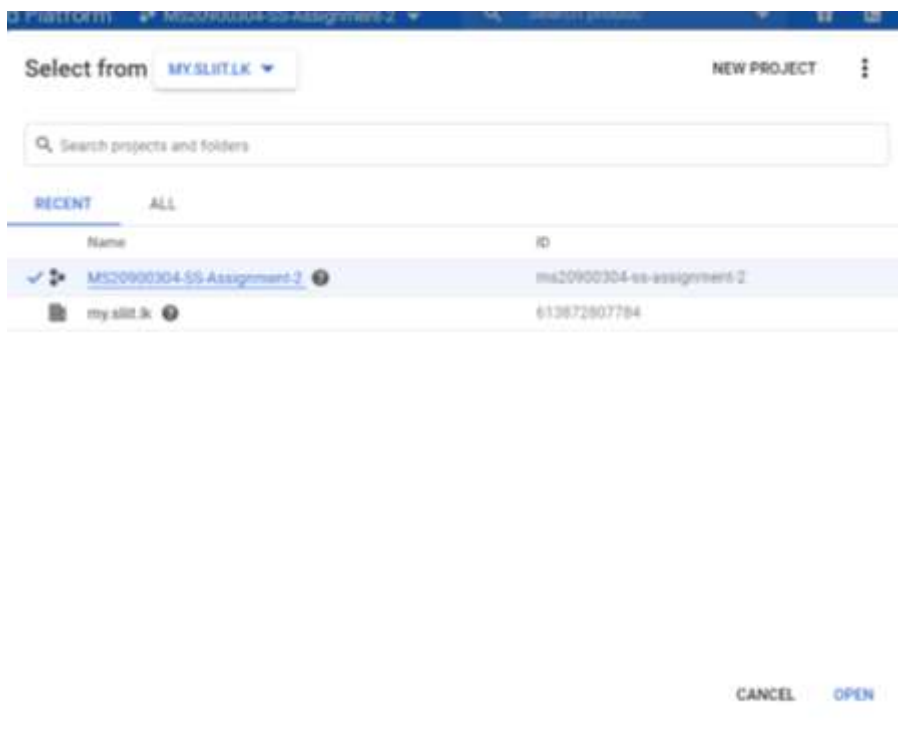
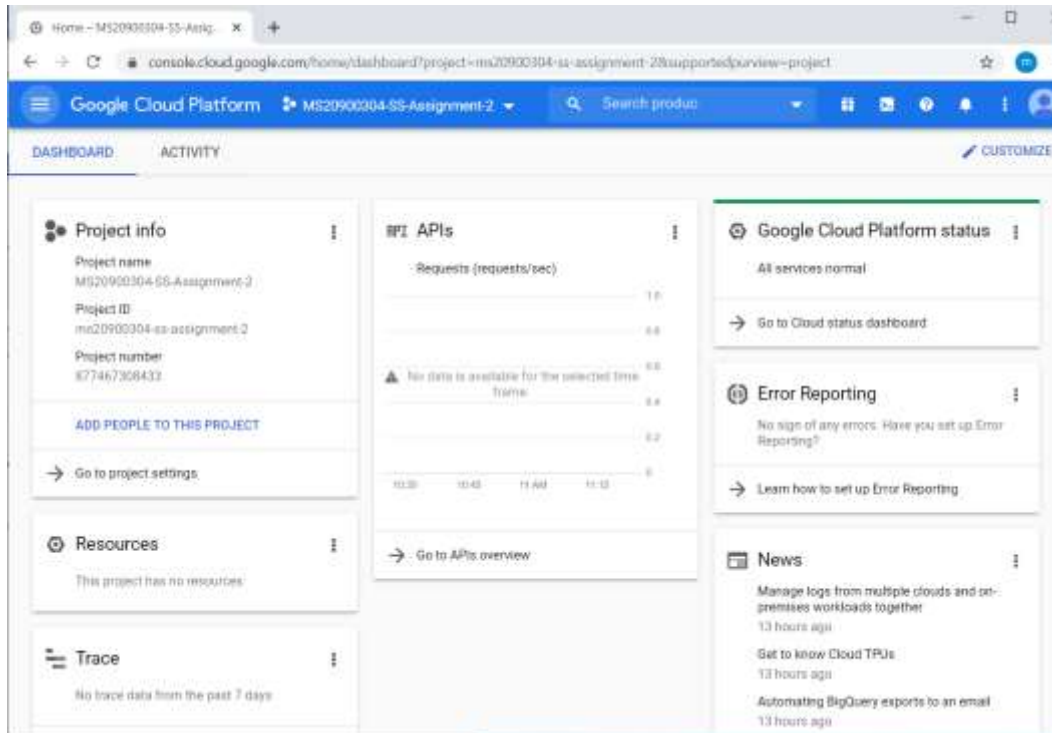
Enter the project name, click EDIT button to change the project ID. For this, following values were used:

Project Name: MS20900304-SS-Assignment-2

ID: ms20900304-ss-assignment-2



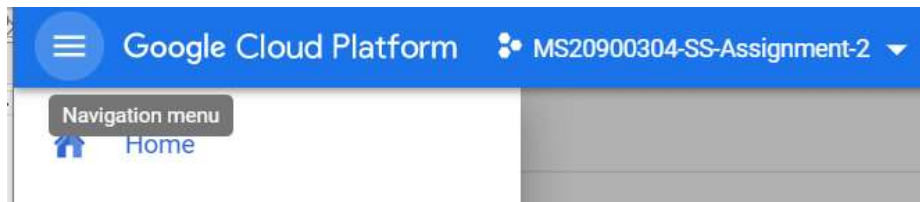
Click "Create" to create the project.



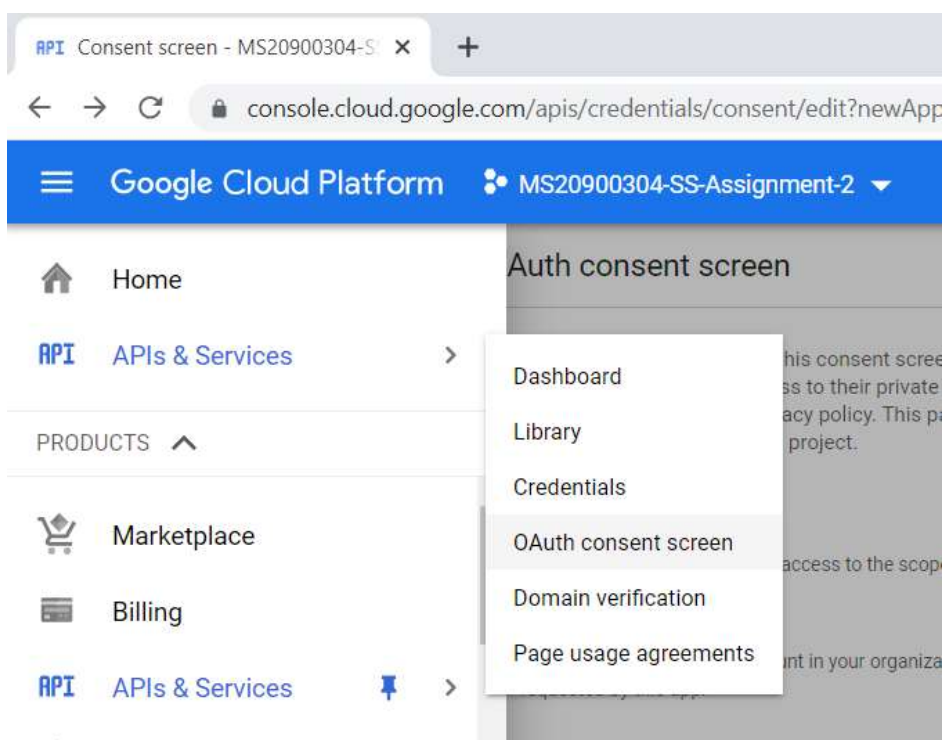
2.1.2 Generate OAuth Client ID

2.1.2.1 Configuring OAuth Consent Screen

Go to "Navigation Menu"



Navigate to **APIs & Services** -> **OAuth consent screen**



In the **OAuth consent screen**, a user type should be selected from the below options:

- **Internal:** Provides access only to the accounts belonging to the organization (in this scenario, my.sliit.lk)
- **External:** Provides access to any user with a Google Account.

In this project, **Internal** option is selected.

Click **CREATE**

The screenshot shows the Google Cloud Platform console for project 'MS20900304-SS-Assignment-2'. The left sidebar lists 'APIs & Services' with options: Dashboard, Library, Credentials, OAuth consent screen (selected), Domain verification, and Page usage agreements. The main content area is titled 'OAuth consent screen' and contains the following text: 'Choose how you want to configure and register your app, including your target users. You can only associate one app with your project.' Below this is the 'User Type' section with two radio buttons: 'Internal' (selected) and 'External'. The 'Internal' option is described as 'Only available to users within your organization. You will not need to submit your app for verification.' The 'External' option is described as 'Available to any user with a Google Account.' At the bottom of the main content area is a blue 'CREATE' button.

User will be redirected to configure the OAuth consent screen:

Keep the option **Internal**, based on the previous input.

This screenshot shows the same Google Cloud Platform console page, but with more configuration options visible. The 'Application type' section has two radio buttons: 'Public' and 'Internal' (selected). The 'Public' option is described as 'Any Google Account can grant access to the scopes required by this app.' The 'Internal' option is described as 'Only users with a Google Account in your organization can grant access to the scopes requested by this app.' Below this is the 'Application name' section, which includes a text input field with the placeholder 'Application name' and a red error message: 'Application name must not be empty'. At the bottom is the 'Application logo' section, which includes a text input field with the placeholder 'Local file for upload' and a 'Browse' button.

Provide an application name. This will be the name of the app asking for consent:

For this assignment, following name is used:

MS20900304-SS-Assignment-2-OAuth-Demo

The screenshot shows the Google Cloud Platform console interface. The top navigation bar includes the Google Cloud Platform logo, the project name 'MS20900304-SS-Assignment-2', and a search bar. The left sidebar shows the 'APIs & Services' menu with options like Dashboard, Library, Credentials, OAuth consent screen (selected), Domain verification, and Page usage agreements. The main content area is titled 'OAuth consent screen' and contains the following fields:

- Application type:** Two radio buttons are present: 'Public' (unselected) and 'Internal' (selected). Below 'Public' is a link to 'Learn more about scopes'. Below 'Internal' is a description: 'Only users with a Google Account in your organization can grant access to the scopes requested by this app.'
- Application name:** A text input field containing 'MS20900304-SS-Assignment-2-OAuth-Demo'. A tooltip indicates: 'The name of the app asking for consent.'
- Application logo:** A section with a description: 'An image on the consent screen that will help users recognize your app'. It includes a 'Local file for upload' button and a 'Browse' button.

Navigate to the bottom of the screen and click **SAVE**

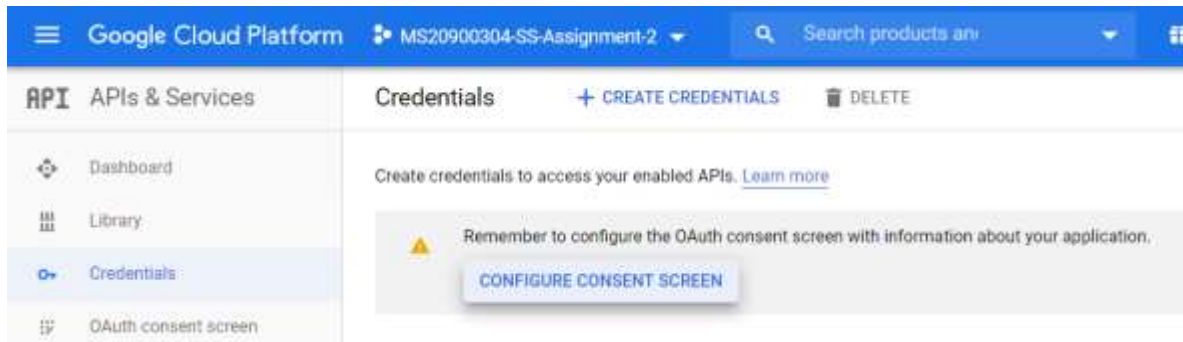
This screenshot shows the bottom portion of the 'OAuth consent screen' configuration page. The top navigation bar and sidebar are identical to the previous screenshot. The main content area includes the following fields:

- Domain:** A text input field containing 'example.com'. Below it is a prompt: 'Type in the domain and press Enter to add it'.
- Application Homepage link:** A text input field containing 'https:// or http://'. A tooltip indicates: 'Shown on the consent screen. Must be hosted on an Authorized Domain.'
- Application Privacy Policy link:** A text input field containing 'https:// or http://'. A tooltip indicates: 'Shown on the consent screen. Must be hosted on an Authorized Domain.'
- Application Terms of Service link (Optional):** A text input field containing 'https:// or http://'. A tooltip indicates: 'Shown on the consent screen. Must be hosted on an Authorized Domain.'

At the bottom of the form are two buttons: 'Save' (highlighted in blue) and 'Cancel'.

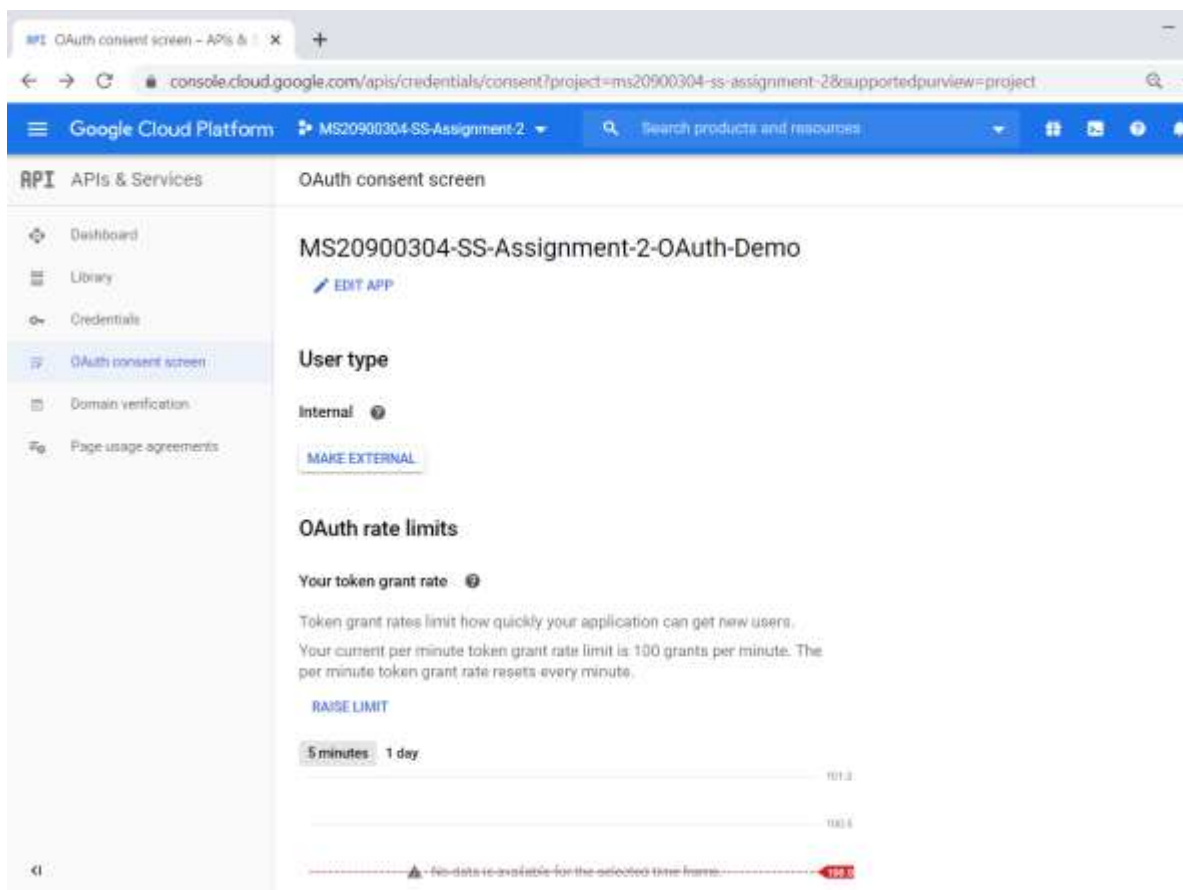
If this complete the Configure Consent Screen process, before creating an OAuth Client ID

To do this, click **CONFIGURE CONSENT SCREEN**



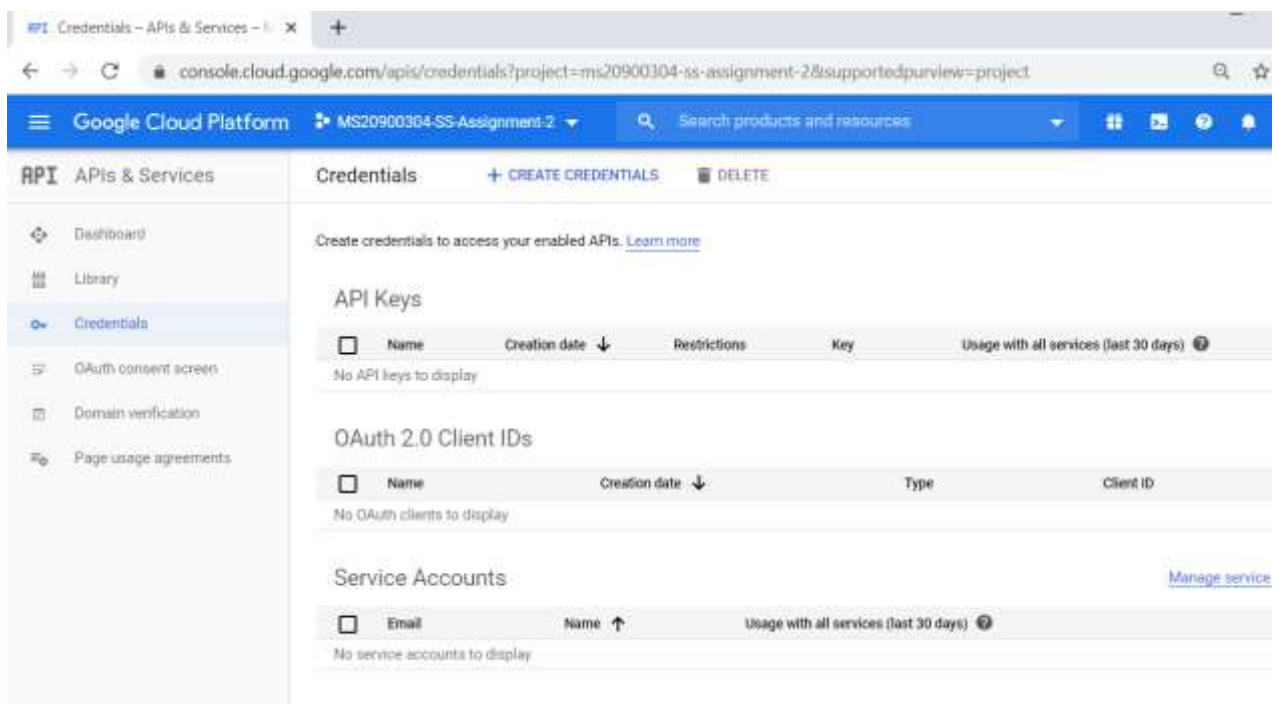
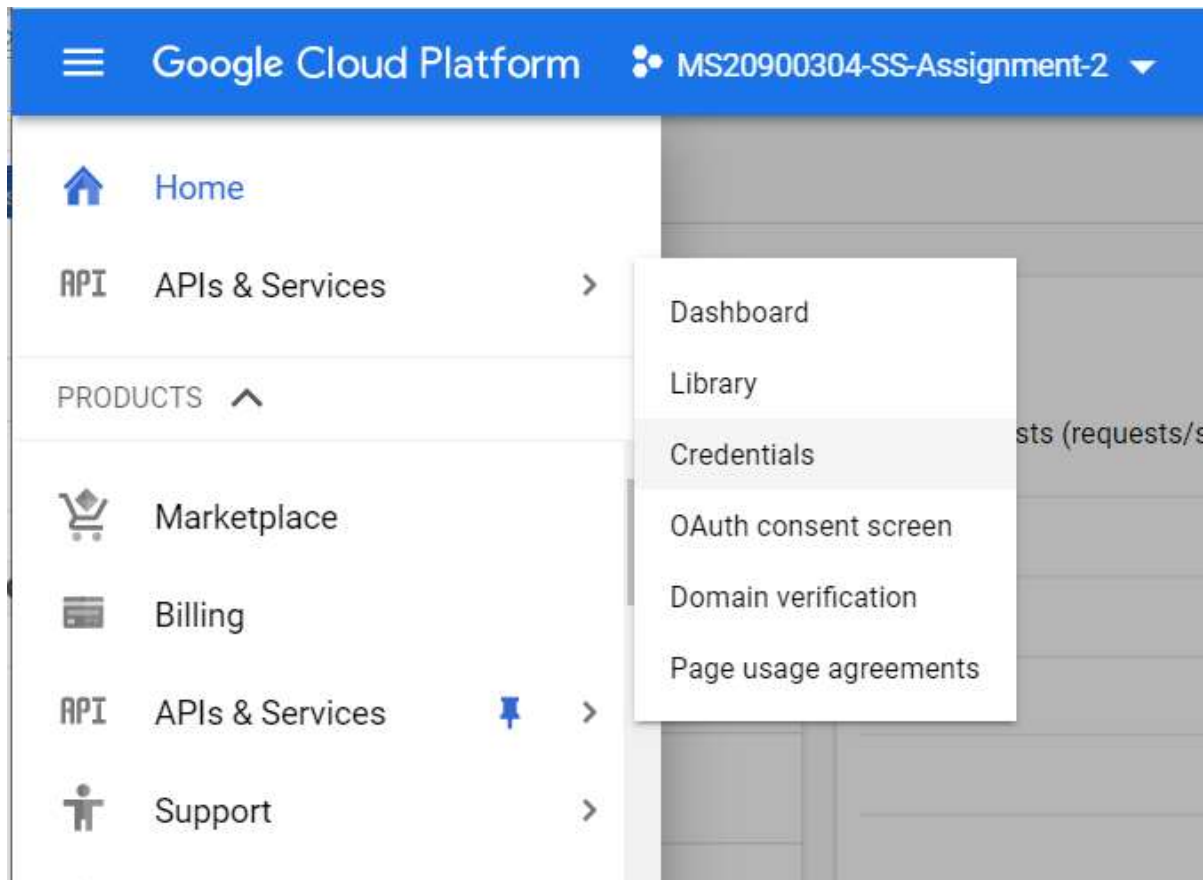
User will be redirected to the following page:

OAuth rate limits can be configured on needs basis.

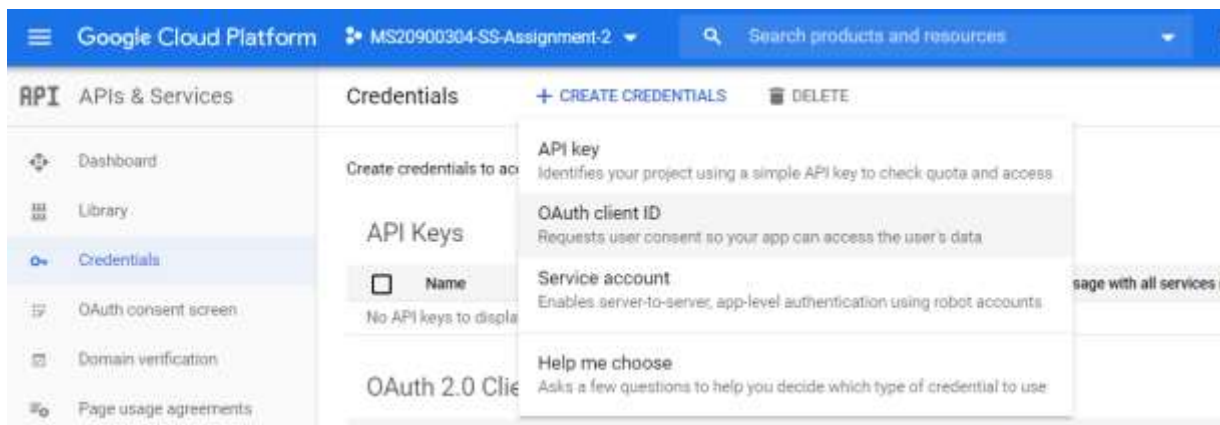
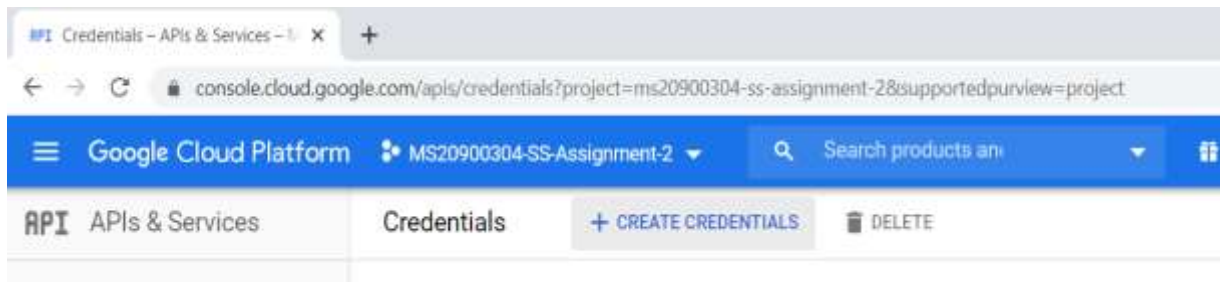


2.1.2.2 OAuth Client ID

Navigate to **APIs & Services** -> **Credentials**



Navigate to **CREATE CREDENTIALS** -> **OAuth Client ID**



For this assignment, select **Web Application** option:

Google Cloud Platform

MS20900304-SS-Assignment-2

←

Create OAuth client ID

For applications that use the OAuth 2.0 protocol to call Google APIs, you can use an OAuth 2.0 client ID to generate an access token. The token contains a unique identifier. See [Setting up OAuth 2.0](#) for more information.

Application type

☐ Web application

☐ Android [Learn more](#)

☐ Chrome App [Learn more](#)

☐ iOS [Learn more](#)

☐ Other

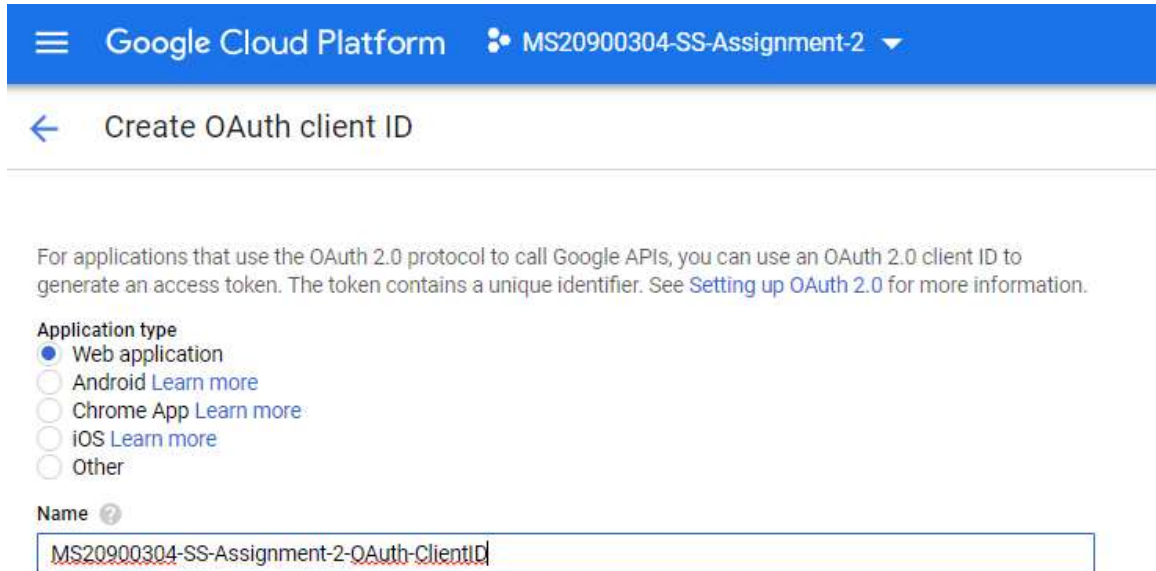
Create

Cancel

Enter a name for the OAuth client ID:

Following name was used:

MS20900304-SS-Assignment-2-OAuth-ClientID



Google Cloud Platform MS20900304-SS-Assignment-2

← Create OAuth client ID

For applications that use the OAuth 2.0 protocol to call Google APIs, you can use an OAuth 2.0 client ID to generate an access token. The token contains a unique identifier. See [Setting up OAuth 2.0](#) for more information.

Application type

- ☒ Web application
- ☐ Android [Learn more](#)
- ☐ Chrome App [Learn more](#)
- ☐ iOS [Learn more](#)
- ☐ Other

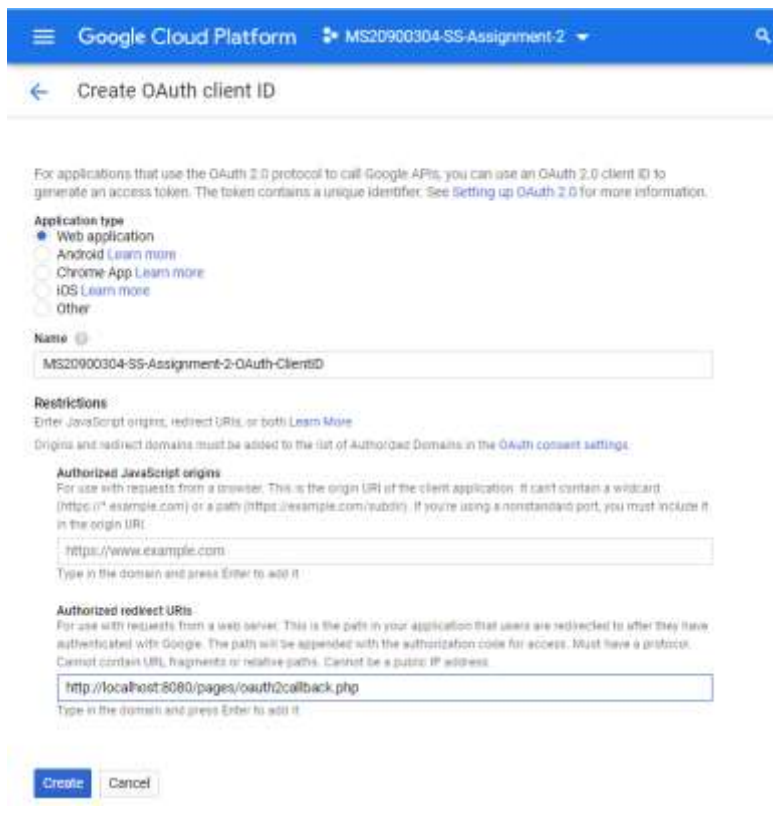
Name ⓘ

MS20900304-SS-Assignment-2-OAuth-ClientID

For the Authorized redirect URIs field, following path is used, which the application users are redirected to after authenticated with Google:

<http://localhost:8080/pages/oauth2callback.php>

Click **Create** to save changes.



Google Cloud Platform MS20900304-SS-Assignment-2

← Create OAuth client ID

For applications that use the OAuth 2.0 protocol to call Google APIs, you can use an OAuth 2.0 client ID to generate an access token. The token contains a unique identifier. See [Setting up OAuth 2.0](#) for more information.

Application type

- ☒ Web application
- ☐ Android [Learn more](#)
- ☐ Chrome App [Learn more](#)
- ☐ iOS [Learn more](#)
- ☐ Other

Name ⓘ

MS20900304-SS-Assignment-2-OAuth-ClientID

Restrictions

Enter JavaScript origins, redirect URIs, or both [Learn More](#)

Origins and redirect domains must be added to the list of Authorized Domains in the [OAuth consent settings](#).

Authorized JavaScript origins

For use with requests from a browser. This is the origin URI of the client application. It can't contain a wildcard (https://*.example.com) or a path (https://example.com/subdir). If you're using a nonstandard port, you must include it in the origin URI.

https://www.example.com

Type in the domain and press Enter to add it.

Authorized redirect URIs

For use with requests from a web server. This is the path in your application that users are redirected to after they have authenticated with Google. The path will be appended with the authorization code for access. Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address.

http://localhost:8080/pages/oauth2callback.php

Type in the domain and press Enter to add it.

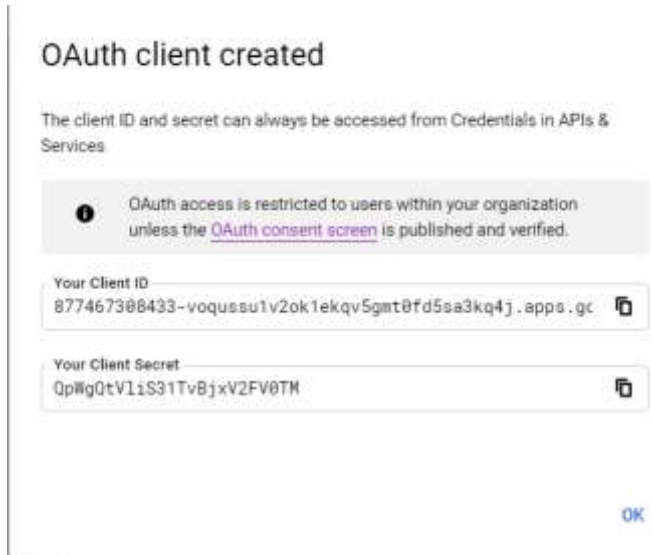
Create Cancel

After successful submission, following popup will appear, with the Client ID and Client Secret:

Client ID: 877467308433-

voqussu1v2ok1ekqv5gmt0fd5sa3kq4j.apps.googleusercontent.com

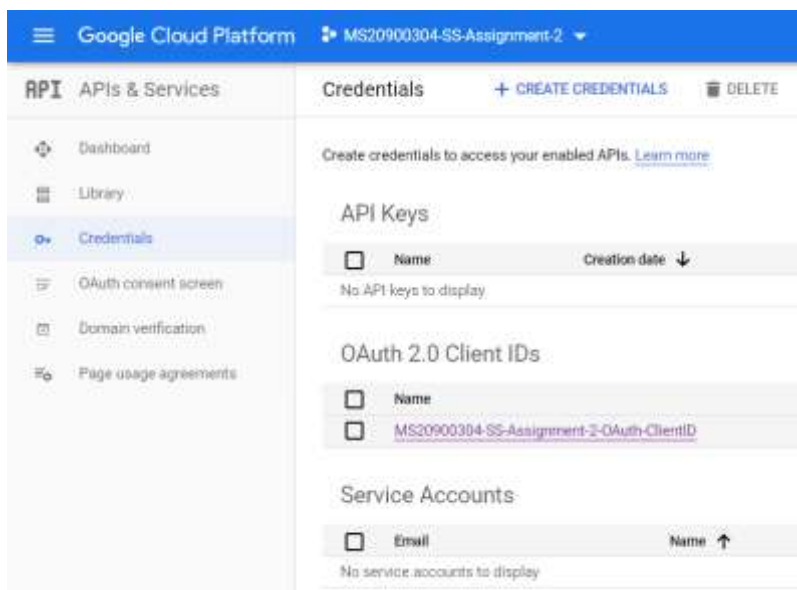
Client Secret: QpWgQtVliS31TvBjxV2FV0TM

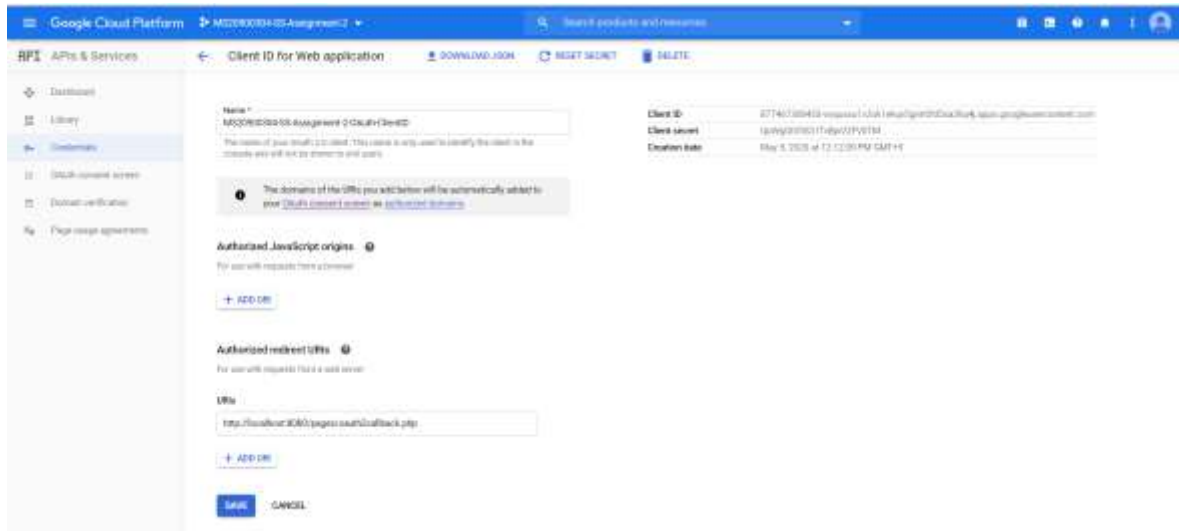


Click OK.

2.1.2.3 Generate OAuth Client ID JSON file

Navigate to **Credentials** - OAuth 2.0 Client IDs -> [Client ID Name] to view the created Client ID details





Click **DOWNLOAD JSON** button to get the Client ID information as a JSON file.

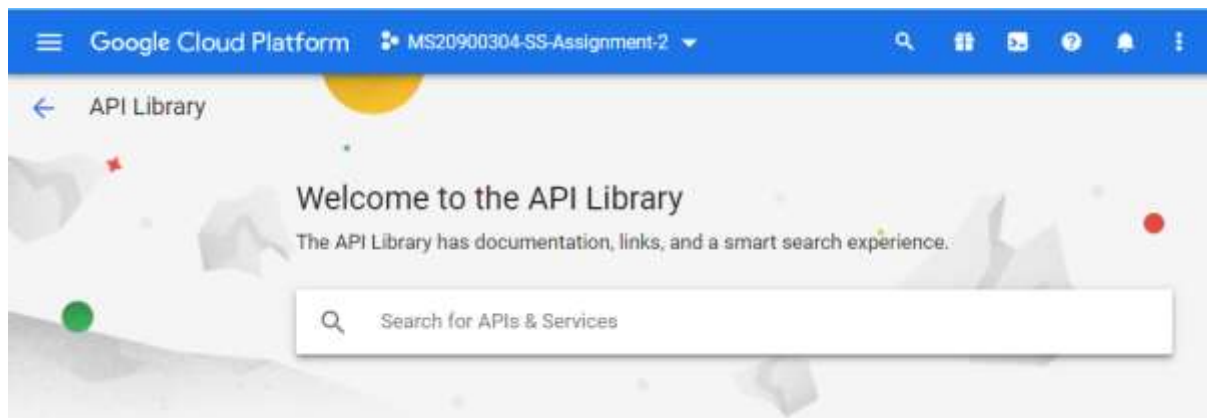
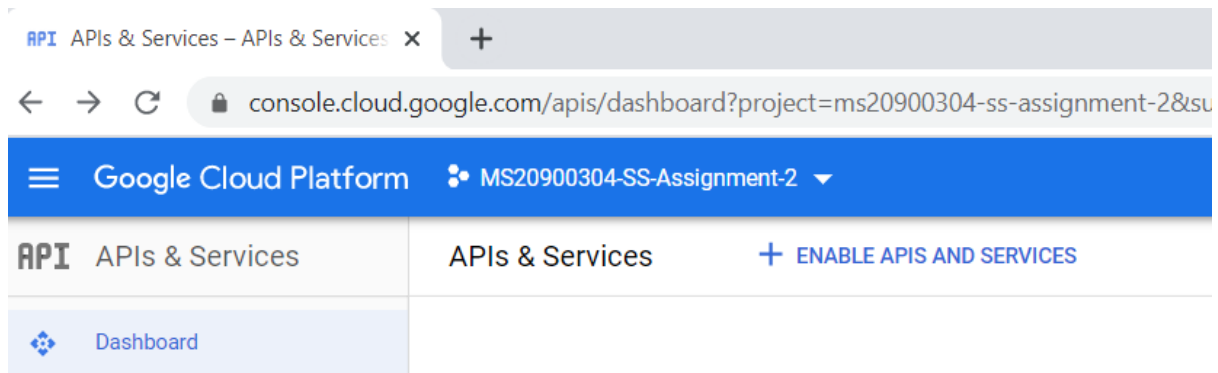


Refer the contents of JSON file in Appendix: - **/resources/client/client_secrets.json**

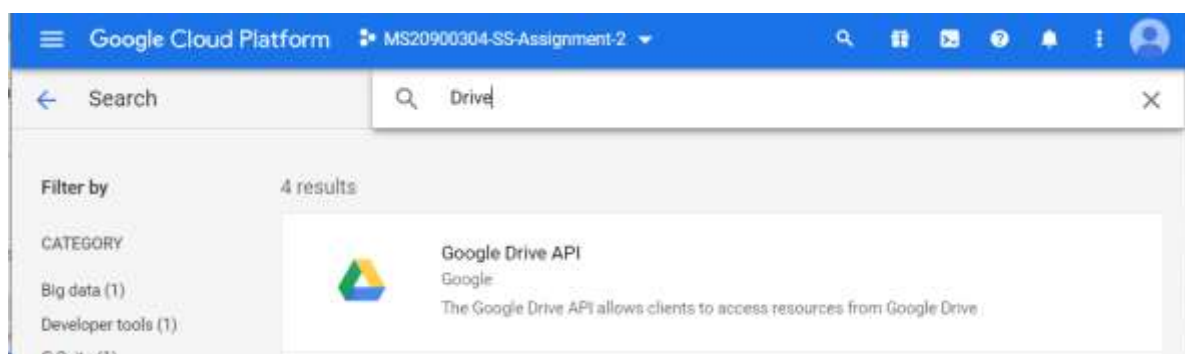
2.2 Configure Resource Server

For this assignment, Google Drive was used as the resource server.

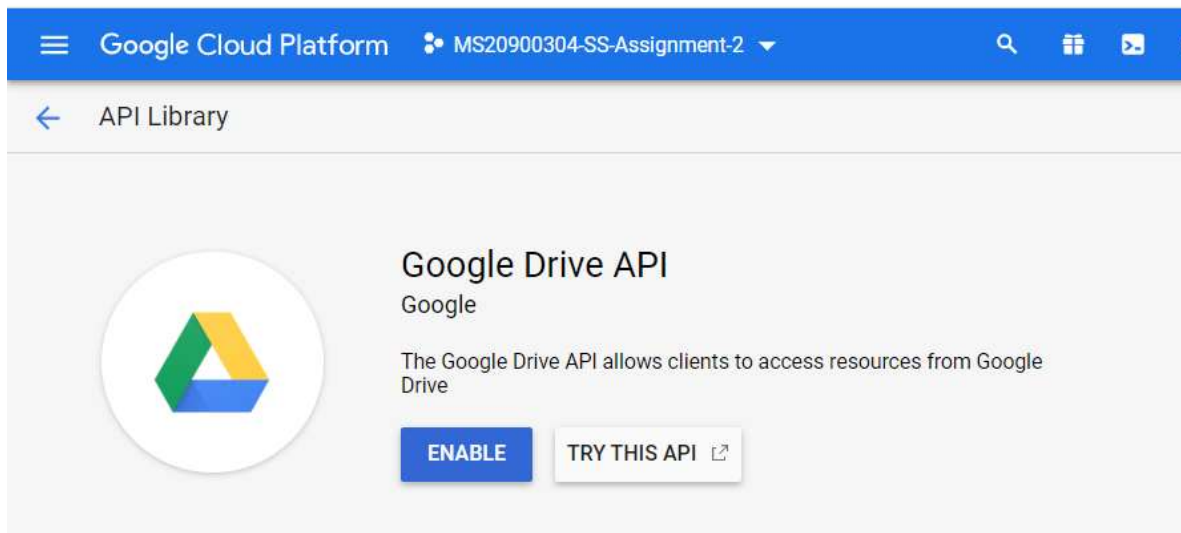
Navigate to **APIs & Services** -> **Dashboard** -> Click **+ ENABLE APIS AND SERVICES**



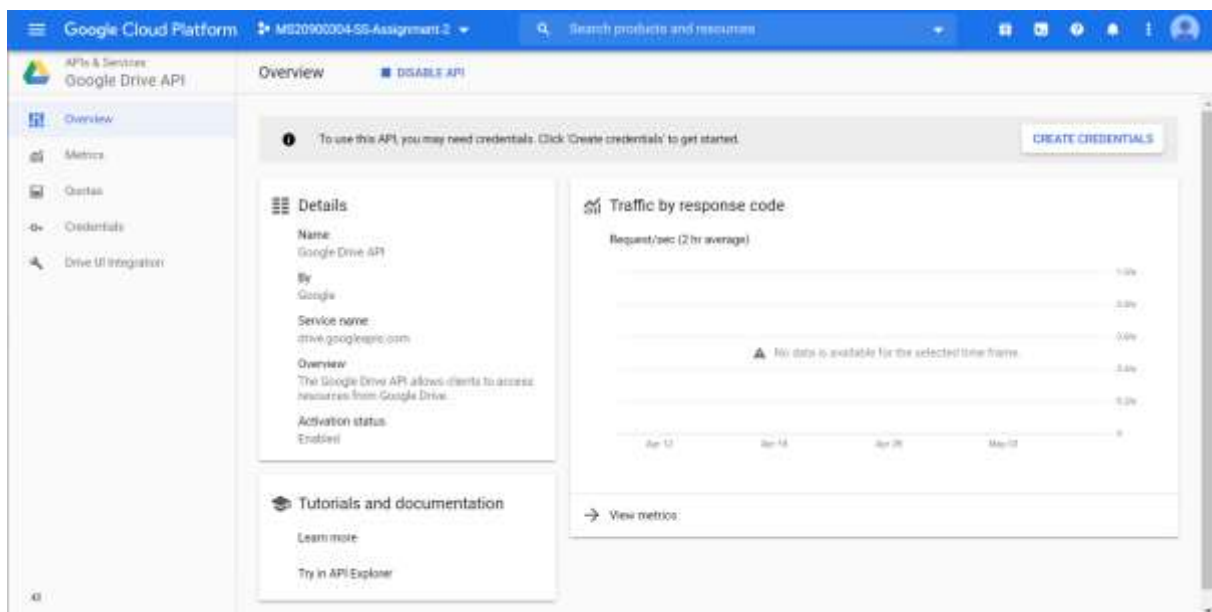
Search for Google Drive API -> Navigate to Google Drive API



Click **ENABLE**



After enabling, the user will be redirected to the API's dashboard



3 Deploy and Run the Web Application

This web application was developed using the following technologies:

Item	Description	Link(s)
Operating System	Ubuntu 19.10 (eoan) [1]	https://releases.ubuntu.com/eoan/
Development IDE	PhpStorm 2020.1 Ultimate [2]	https://www.jetbrains.com/phpstorm/
Development Language	PHP 7.3 [3]	https://www.php.net/
Dependency Manager	Composer 1.10.6 [4]	https://getcomposer.org/
Authorization Server	Google OAuth2 [5]	https://developers.google.com/identity/protocols/oauth2
Resource Server	Google Drive API [6]	https://developers.google.com/drive

3.1 Deploying the Web Application

After configuring authorization server and resource servers as above, conduct the following steps to deploy the web application.

3.1.1 Download the contents as a ZIP file:

Navigate to the following link on GitHub website:

<https://github.com/pd-720/MS20900304-Assignment2>

Navigate to 'Clone or Download' -> 'Download Zip'.

3.1.2 Set up PHP environment in Ubuntu:

Execute the following command:

```
sudo apt install php7.3 php7.3-common php7.3-opcache php7.3-cli php7.3-gd php7.3-curl php7.3-mysql
```

Note: Use the following link for further information:

<https://linuxize.com/post/how-to-install-php-on-ubuntu-18-04/#installing-php-73-on-ubuntu-1804>

3.1.3 Configure PHP interpreter in PhpStorm:

Use the following link for further information:

<https://www.jetbrains.com/help/PhpStorm/configuring-local-interpreter.html>

3.1.4 Install Composer dependency manager:

Execute the following command in terminal to install Composer:

```
sudo apt install composer
```

3.1.5 Initialize Composer for the project:

Navigate to the project root folder.

Execute the following commands:

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php -r "if (hash_file('sha384', 'composer-setup.php') ===
'e0012edf3e80b6978849f5eff0d4b4e4c79ff1609dd1e613307e16318854d24ae64f26d17af3ef0bf
7c7b710ca74755a') { echo 'Installer verified'; } else { echo 'Installer corrupt';
unlink('composer-setup.php'); } echo PHP_EOL;"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

3.1.6 Resolve Google API client dependencies using Composer:

Execute the following command to import Google API client and resolve dependencies:

```
php composer.phar require google/apiclient:^2.0
```

Note: After completion, a folder named 'vendor' will be created.

3.1.7 Update composer.phar file:

Execute the following command to update the composer.phar file:

```
php composer.phar update
```

3.2 Running the Web Application

Run the application by using the following command:

```
php -S localhost:8080 | firefox http://localhost:8080/login.php
```

Note: Save the command lines to a shell script and reuse.

4 Webpage Flow

4.1 Authorization Flow

File list

```
[root_folder]
| --- login.php
| --- [pages]
|     --- index.php
|     --- oauth2callback.php
|     --- logout.php
| --- [resources]
|     --- [client]
|         --- client_secrets.json
|     --- [files]
|         --- style.css
```

4.1.1 login.php

Web application starts on this page and if the link is clicked, the user is redirected to the index.php page

4.1.2 index.php

If an access token is:

a. not generated:

the user will be redirected to oauth2callback.php page after adding necessary scopes to complete OAuth2 authentication (i.e., valid login credential details) from user using an object from Google_Client class.

b. generated:

get the consent from the user to access Google Drive to retrieve and print all contents, using an object from Google_Service_Drive class.

4.1.3 oauth2callback.php

When the access token is:

a. not generated:-

Create authorization URL to be sent to Google when redirecting for OAuth2 authentication to create the access token (with the refresh token at the

offline access method) from an object of Google_Client class by using the contents from client_secrets.json and scopes:

Contents from client_secrets.json:

```
client_id: 877467308433-  
voqussu1v2ok1ekqv5gmt0fd5sa3kq4j.apps.googleusercontent.com  
project_id: ms20900304-ss-assignment-2  
auth_uri: https://accounts.google.com/o/oauth2/auth  
token_uri: https://oauth2.googleapis.com/token  
auth_provider_x509_cert_url: https://www.googleapis.com/oauth2/v1/certs  
client_secret: QpWgQtVliS31TvBjxV2FV0TM  
redirect_uris: [http://localhost:8080/pages/oauth2callback.php]
```

Scopes:

OAuth2 permissions:

- Google_Service_Oauth2::USERINFO_EMAIL : View user's email address [8].
- Google_Service_Oauth2::USERINFO_PROFILE : View user's personal information, including any personal info that the user has made public [8].

Google Drive permissions:

Google_Service_Drive::DRIVE_METADATA : View and manage metadata of files in your Google Drive [9].

b. generated:-

Redirect the flow to index.php. Subsequently, based on the provided consent, create an object from Google_Service_Drive class to access user's files and folders and list them, based on the defined scope.

4.1.4 logout.php

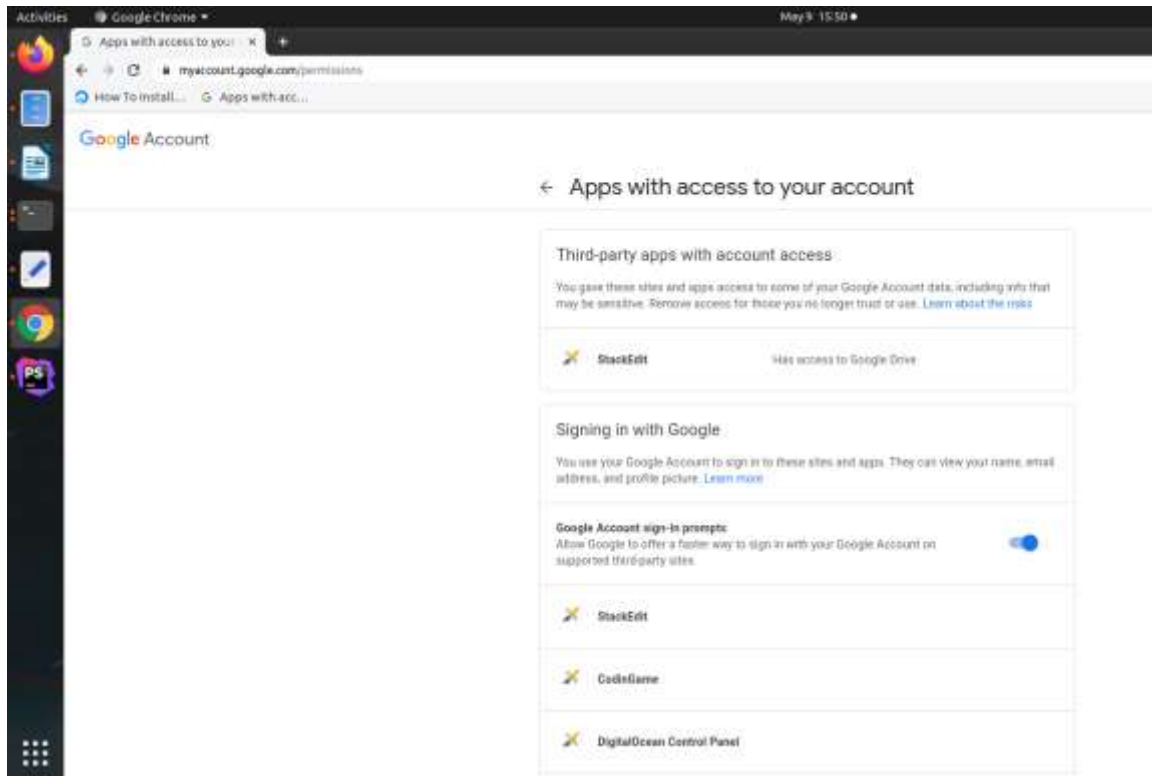
When redirected, the access token will be revoked using the revokeToken method.

4.1.5 style.css

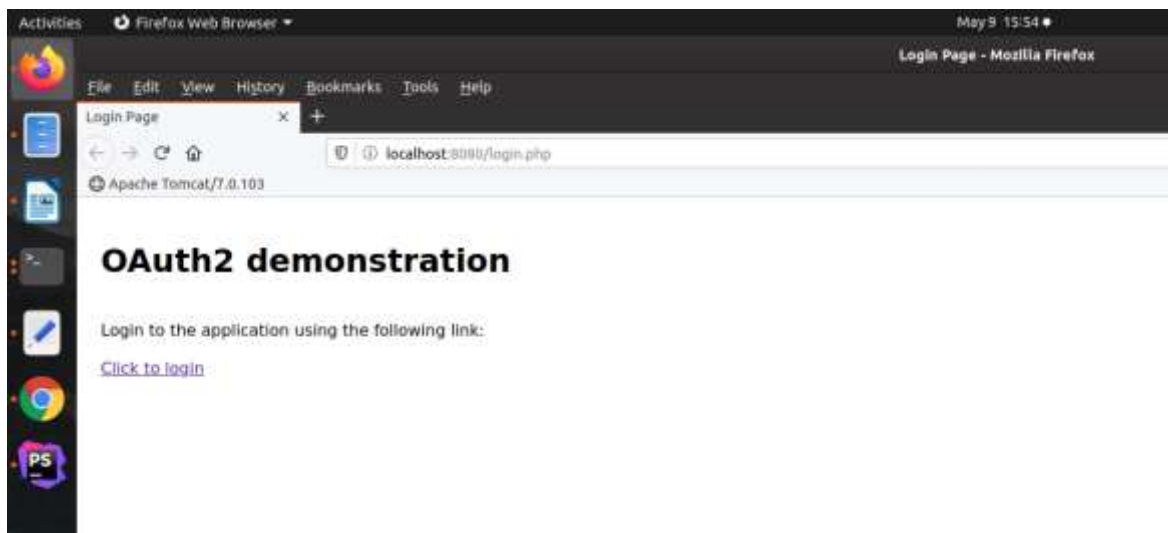
Sets the css layouts within the PHP web pages.

4.2 Web pages

4.2.1 Access not granted from Google Permissions



4.2.2 Web Application Login page



4.2.3 OAuth2 redirect (Sinhala)

https://accounts.google.com/signin/oauth/identifier?response_type=code&access_type=offline&client_id=877467308433-

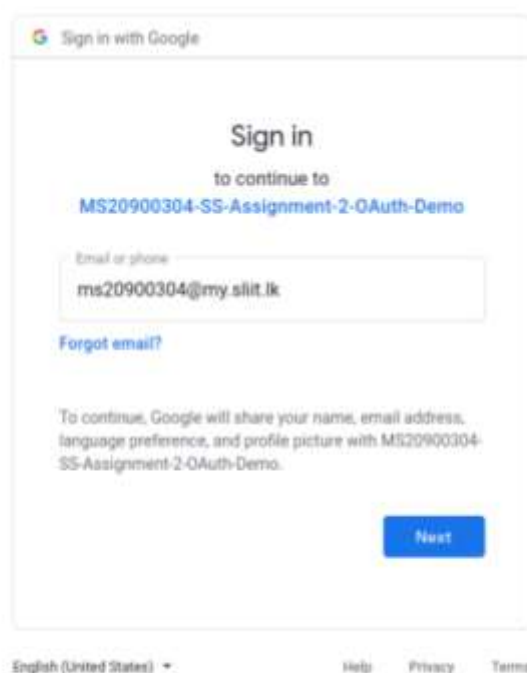
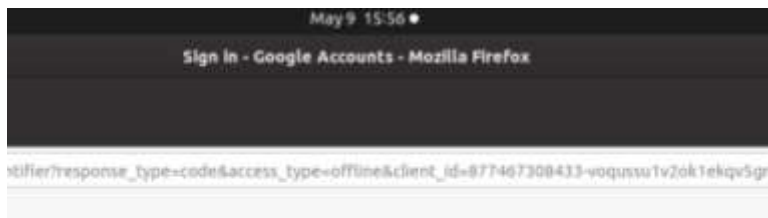
voqussu1v2ok1ekqv5gmt0fd5sa3kq4j.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fpages%2Foauth2callback.php&state&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.profile%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.metadata&approval_prompt=force&v=1&as=5h4NxwveTekoVwcnYyF_vw&flowName=GeneralOAuthFlow



4.2.4 OAuth2 redirect (English)

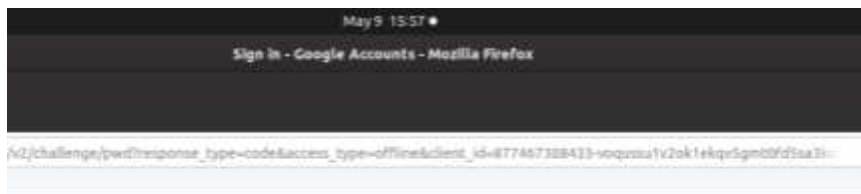
The original callback processes in locale (Sinhala). Therefore, for management purposes, the locale is changed to English

https://accounts.google.com/signin/oauth/identifier?response_type=code&access_type=offline&client_id=877467308433-voqussu1v2ok1ekqv5gmt0fd5sa3kq4j.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fpages%2Foauth2callback.php&state&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.profile%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.metadata&approval_prompt=force&as=5h4NxwveTekoVwcnYyF_vw&hl=en&flowName=GeneralOAuthFlow



4.2.5 Email address accepted

https://accounts.google.com/signin/v2/challenge/pwd?response_type=code&access_type=offline&client_id=877467308433-voqussu1v2ok1ekqv5gmt0fd5sa3kq4j.apps.googleusercontent.com&redirect_uri=http%3A%2F%2Flocalhost%3A8080%2Fpages%2Foauth2callback.php&state&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.profile%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fdrive.metadata&approval_prompt=force&o2v=1&as=5h4NxwveTekoVwcnYyF_vw&hl=en&flowName=GeneralOAuthFlow&cid=1&navigationDirection=forward&TL=AM3QAYaF5ESYBgPjrn0BYuFDSSszjMYUSxjFEy1fdtJF1gf1Zkg3D98Mm2pkWXeW

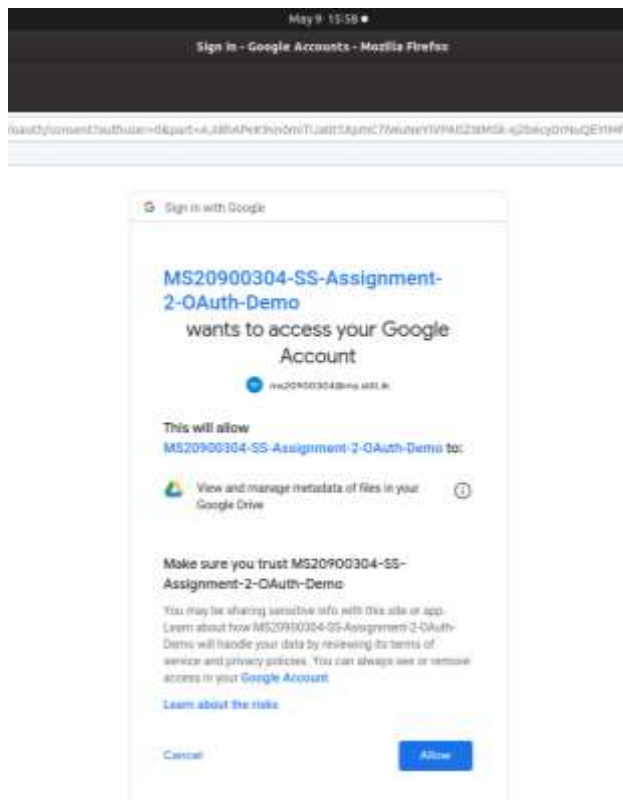


4.2.6 Password accepted, Google Drive permissions request

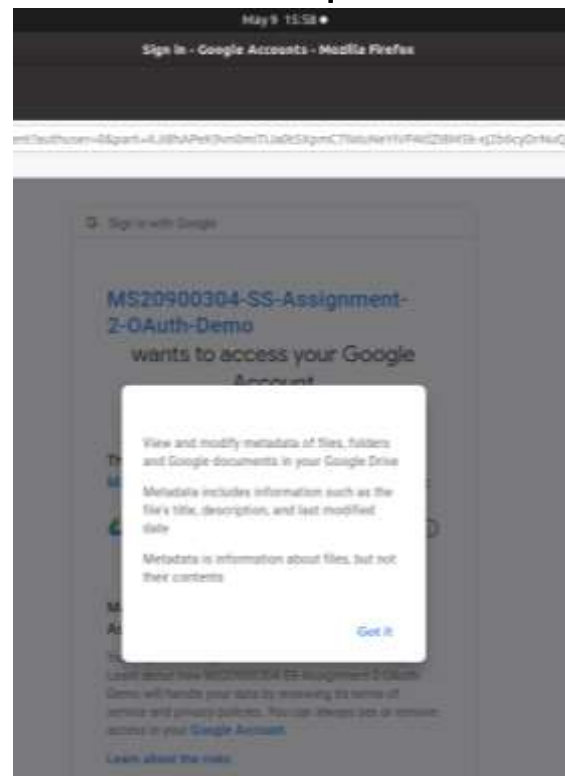
https://accounts.google.com/signin/oauth/consent?authuser=0&part=Aji8hAPeK9vn0miTiJa0t5Xp mC7IWuNeYiVPAtlZI8M5k-xj2b6cyDrNuQEYIMF0HBm_ob-

[bYHBwkjV6g35neRkVL2_p7ltetHmyygsYDFLKi1x91dzqzh6l3fg7za_pOv_SoEizBpmMgkMizPaF
Cyd9PxEyK0NY5I3rR-
jW_6rLBhvEetHOvgseV_UeF_PJTUxZR13xD3ZQEVIm2uy_PfAQTSymvmLwimOoX_vOSdHZZsY
Dris6ZOwwKYXqDUz_wxzQcRIS6BDELL4lrBvuaUq2IVE2Ba8EaUDvSCSTBUxFYfh4YoWYXZqVT
5kSpptBnQeHVKBkdINLyOC2CY-
PIRxGOBBNh6vXoMZhDBKj3CaaZE3TDegnev9zwCIUWYjIVY2KOeAFsBDOhBHjWr-
p7j9Sj1H8EO_Jz-
66ZoyT6pd2oaBWzCgmeM&hl=en&as=5h4NxxveTekoVwcnYyF_vw&rapt=AEjHL4Naz6hHfT
aYCl65Y74P51ToccOmwrUN141dgUi_RZbDVzTU9GV4s2pI0G7E5Ri9A7MEn49g1rBtA7JtHghA
64w3xvojWA](#)

Google Drive Consent Screen



Consent Requests



4.2.7 index.php loaded with file list (Callback completed)

<http://localhost:8080/pages/index.php>

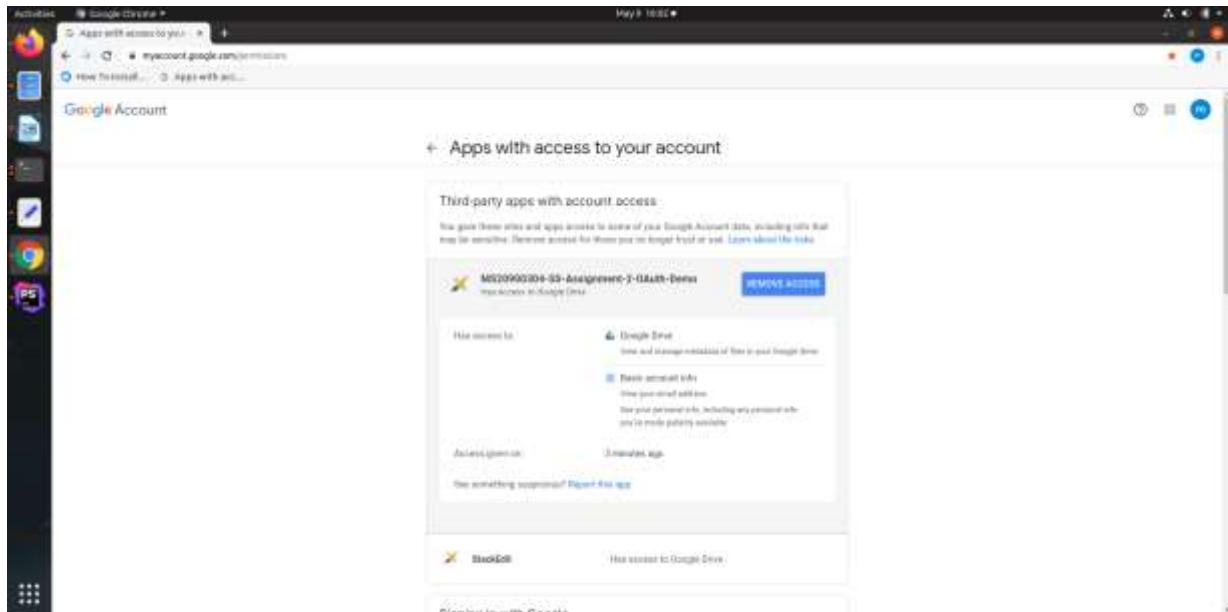


24

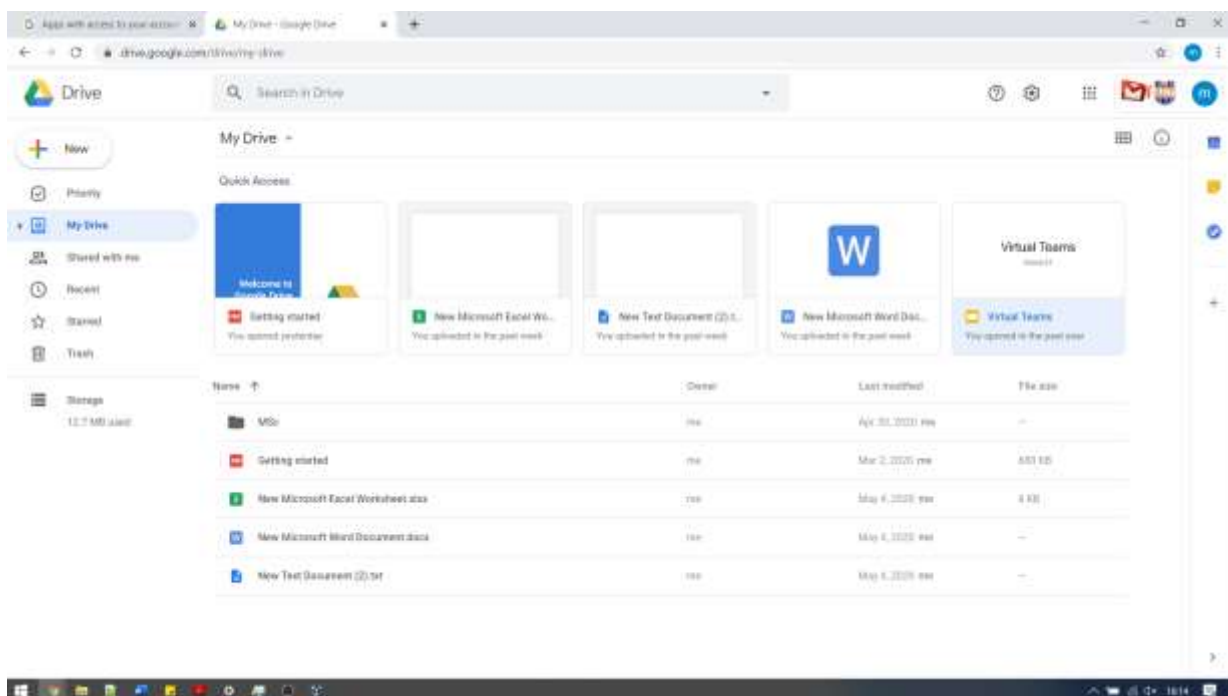
4.2.9 Logout page



4.2.10 Access granted on Google Permissions



4.2.11 List of files on Google Drive



References

- [1] Ubuntu 19.10 (eoan): <https://releases.ubuntu.com/eoan/>
- [2] PhpStorm 2020.1 Ultimate: <https://www.jetbrains.com/phpstorm/>
- [3] PHP 7.3: <https://www.php.net/>
- [4] Composer 1.10.6: <https://getcomposer.org/>
- [5] Google OAuth2: <https://developers.google.com/identity/protocols/oauth2>
- [6] Google Drive API: <https://developers.google.com/drive/>
- [7] Creating Google Cloud Platform Projects: <https://console.developers.google.com/>
- [8] Google OAuth2 API permissions: <https://developers.google.com/resources/api-libraries/documentation/oauth2/v2/java/latest/com/google/api/services/oauth2/Oauth2Scopes.html>
- [9] Google Drive Permissions: <https://developers.google.com/resources/api-libraries/documentation/drive/v3/java/latest/com/google/api/services/drive/DriveScopes.html>

Appendix - Artefacts

File list

```
[root_folder]
| --- login.php
| --- [pages]
|     | --- index.php
|     | --- oauth2callback.php
|     | --- logout.php
| --- [resources]
|     | --- [client]
|         | --- client_secrets.json
|     | --- [files]
|         | --- style.css
```

/resources/client/client_secrets.json

```
{
  "web": {
    "client_id": "877467308433-voqussu1v2ok1ekqv5gmt0fd5sa3kq4j.apps.googleusercontent.com",
    "project_id": "ms20900304-ss-assignment-2",
    "auth_uri": "https://accounts.google.com/o/oauth2/auth",
    "token_uri": "https://oauth2.googleapis.com/token",
    "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
    "client_secret": "QpWgQtVliS31TvBjxV2FV0TM",
    "redirect_uris": [
      "http://localhost:8080/pages/oauth2callback.php"
    ]
  }
}
```

login.php

```
<?php

//CSS style sheet
echo '<link rel = "stylesheet" type = "text/css" href =
"./resources/files/style.css" />';

if (!isset($_SESSION['access_token'])) {
    echo '<div>';
    echo '<h1>' . 'OAuth2 demonstration' . '</h1>';
    echo '<br>';
    echo 'Login to the application using the following link: ' . '<br>';
    echo '<br>';
    echo '<a href="pages/index.php"> Click to login </a>';
    echo '</div>';
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login Page</title>
```

```
<script type="application/javascript">
    window.moveTo(0, 0);
    window.resizeTo(screen.width, screen.height);
    window.innerWidth = screen.width;
    window.innerHeight = screen.height;
    window.screenX = 0;
    window.screenY = 0;
    alwaysLowered = false;
</script>

</head>
<body>

</body>
</html>
```

/pages/index.php

```
<?php

    //Call the Google OAuth2 API client
    require_once dirname(getcwd(),1).'/vendor/autoload.php';

    //Import contents from client_secrets.json file containing the OAuth2 client
    details
    $client_secrets_file=
    dirname(getcwd(),1).'/resources/client/client_secrets.json';

    //CSS style sheet
    echo '<link rel = "stylesheet" type = "text/css" href =
    "/resources/files/style.css" />';

    //Start the PHP session
    session_start();

    //Create Google_Client object
    $client = new Google_Client();

    //Import OAuth2 client details from client_secrets.json file
    $client->setAuthConfigFile($client_secrets_file);

    //Add scopes to ask for the user permissions
        //OAuth2 permissions
    $client->addScope(Google_Service_Oauth2::USERINFO_EMAIL);
    $client->addScope(Google_Service_Oauth2::USERINFO_PROFILE);
        //Google Drive permissions
    $client->addScope(Google_Service_Drive::DRIVE_METADATA);

    // To get both an access and refresh token so that it can refresh the access
    token without user interaction.
    $client->setAccessType('offline');
    // Allows the web application to always receive a refresh token.
    $client->setApprovalPrompt('force');

    if (isset($_SESSION['access_token']) && $_SESSION['access_token']) {
```



```
//Set the session token with access token
$client->setAccessToken($_SESSION['access_token']);

echo '<H1>Index page</H1>';

//-----User details-----
$google_oauth = new Google_Service_Oauth2($client);
$google_account_info = $google_oauth->userinfo->get();

$user_name = $google_account_info->name;
$user_Email = $google_account_info->email;

//User details printing
echo '<p>' . 'You are logged in as: ' . '</p>';

echo '<div class="tg-wrap">';
echo '<table >';
    echo '<tr>';
        echo '<td><b>' . "Username:" . '</b></td>';
        echo '<td>' . $user_name . '</td>';
    echo '</tr>';
    echo '<tr>';
        echo '<td><b>' . "Email:" . '</b></td>';
        echo '<td>' . $user_Email . '</td>';
    echo '</tr>';
echo '</table>';
echo '</div>';

//Logout page link
echo '<br><a href="logout.php"> Click to Logout </a><br>';

//-----Print Token details-----
echo "<div class='div-vdmp'>";

echo '<p><H2>Token details</H2></p>';
$access_token = $client->getAccessToken();
echo '<pre>'.json_encode($access_token,JSON_PRETTY_PRINT |
JSON_UNESCAPED_SLASHES).'</pre>';

echo "</div>";

//-----Print file list-----
echo '<div>';

//Create Google_Service_Drive object
$drive_service = new Google_Service_Drive($client);

echo '<p><H2>' . 'Google Drive Contents:-' . '</H2></p>';

//Array to determine file parameters taken from the Google_Service_Drive
Object
$file_params = array(
    'pageSize' => 100,
    'fields' =>
'nextPageToken,files(name,createdTime,fileExtension,mimeType)'
);
//Extract the list of files (and folders)
$item_list = $drive_service->files->listFiles($file_params)->getFiles();
```

```
//Get the item count
$all_item_count = count($item_list);

//Item count output
echo '<Strong>Number of Items (files and folders) in Drive:- </strong>' .
$all_item_count . '<br>';
echo '<br>';

//Print the items list in Google Drive
if ($all_item_count == 0) {
    echo "No files found ..." . '<br>';
} else {

    /* ----Determining file parameters in table---- */
    //values taken from the $file_params array
    $array_fields_val = '';
    foreach ($file_params as $key => $val) {
        $array_fields_val = $val;
    }
    //remove files and brackets
    $file_attr_m = str_replace(array('(', ')', ' '), '',
str_replace('files', '', $array_fields_val));

    //generate array
    $array_file_params = explode(',', $file_attr_m);

    //remove 'nextPageToken' parameter
    unset($array_file_params[0]);

    //Print the item list
    echo '<div class="tg-wrap">';
    echo '<table class="tg">';

        echo '<thead>';

            echo '<tr>';
            echo '<th class="tg-lboi">'.<b>' . '#' . '</b>'.</th>';
            echo '<th class="tg-lboi">'.<b>' . 'File Name' .
'</b>'.</th>';
            echo '<th class="tg-lboi">'.<b>' . 'Created Time' .
'</b>'.</th>';
            echo '<th class="tg-lboi">'.<b>' . 'Extension' .
'</b>'.</th>';
            echo '<th class="tg-lboi">'.<b>' . 'mime type' .
'</b>'.</th>';
            echo '</tr>';

        echo '<thead>';
        echo '<tbody>';
        $i=0;
        foreach ($item_list as $file) {
            echo '<tr>';
            echo '<td class="tg-0pky">' . ++$i . '</td>';
            echo '<td class="tg-0pky">' . $file->getName() . '</td>';
            echo '<td class="tg-0pky">' . $file->getCreatedTime() .
'</td>';
            echo '<td class="tg-0pky">' . $file->getFileExtension() .
'</td>';
```

```
                echo '<td class="tg-0pky">' . $file->getMimeType() .
'</td>';
                echo '</tr>';
            }
            echo '</tbody>';
            echo '</table>';
            echo '</div>';
        }
    } else {
        //If the user is not logged in, redirect the user to the call back URL
        $redirect_uri = 'http://' . $_SERVER['HTTP_HOST'] .
'/pages/oauth2callback.php';
        header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));
    }
?>
```

/pages/oauth2callback.php

```
<?php
//Call the Google OAuth2 API client
require_once dirname(getcwd(),1).'/vendor/autoload.php';

//Import contents from client_secrets.json file containing the OAuth2 client
details
$client_secrets_file=
dirname(getcwd(),1).'/resources/client/client_secrets.json';

//Start the PHP session
session_start();

//Create Google_Client object
$client = new Google_Client();

//Import OAuth2 client details from client_secrets.json file
$client->setAuthConfigFile($client_secrets_file);

//Set redirectUri value
$client->setRedirectUri('http://' . $_SERVER['HTTP_HOST'] .
'/pages/oauth2callback.php');

//Add scopes to ask for the user permissions
//OAuth2 permissions
$client->addScope(Google_Service_Oauth2::USERINFO_EMAIL);
$client->addScope(Google_Service_Oauth2::USERINFO_PROFILE);
//Google Drive permissions
$client->addScope(Google_Service_Drive::DRIVE_METADATA);

// To get both an access and refresh token so that it can refresh the access
token without user interaction.
$client->setAccessType('offline');
// Allows the web application to always receive a refresh token.
$client->setApprovalPrompt('force');

if (! isset($_GET['code'])) {
```

```
// Create authorization URL to be sent to Google when redirecting for
permissions
$auth_url = $client->createAuthUrl();
//Refresh the page to go to the initial page
header('Location: ' . filter_var($auth_url, FILTER_SANITIZE_URL));
} else {
    //If the authorization is provided, set the access token for the session
    and redirect the URI to initial page
    $client->authenticate($_GET['code']);
    $_SESSION['access_token'] = $client->getAccessToken();
    $redirect_uri = 'http://' . $_SERVER['HTTP_HOST'] . '/pages/index.php';
    header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));
}

?>
```

/pages/logout.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Logout page</title>
</head>
<body>

<H1>Logout page</H1>
<br>

    <?php
        echo '<link rel = "stylesheet" type = "text/css" href =
"./../resources/files/style.css" />';
        if (isset($_SESSION['access_token'])) {
            $client->revokeToken();
            echo '<h6>You are logged out ...</h6>'.<br>';
        } else {
            echo 'Please ' . '<a href="./../login.php">login</a>' . ' to continue
...'.<br>';
        }

    ?>

</body>
</html>
```

/resources/files/style.css

```
html
body
{
    font-family:Arial Helvetica, sans-serif;
    border-radius: 1em;
    padding: 1em;
```

```
    position: absolute;
    /*top: 50%;*/
    /*left: 50%;*/
    /*margin-right: -50%;*/
    /*transform: translate(-50%, -50%) ;*/
}

/*Table css*/
.tg {border-color:#ccc;border-spacing:0.1px;align-content: center}
.tg td{background-color:#fff;border-color:#ccc;border-style:solid;border-
width:1px;color:#333;
    font-family:Arial, sans-serif;font-size:14px;overflow:hidden;padding:10px
5px;word-break:normal;}
.tg th{background-color:#f0f0f0;border-color:#ccc;border-style:solid;border-
width:1px;color:#333; table-layout: auto;
    font-family:Arial, sans-serif;font-size:14px;font-
weight:normal;overflow:hidden;padding:10px 5px;word-break:normal;}
.tg .tg-lboi{border-color:inherit;text-align:left;vertical-align:middle}
.tg .tg-0pky{border-color:inherit;text-align:left;vertical-align:top; border-
style: ridge; }
@media screen and (max-width: 767px) {.tg {width: auto !important;}.tg col {width:
auto !important;}.tg-wrap {overflow-x: auto;-webkit-overflow-scrolling: touch;}}

div.div-vdmp{
    /*var_dump*/
    width: content-box;
    background-color: #f6f6f6;
}

p {
    font-family: Helvetica, sans-serif;
}

h1,h2,h3,h4,h5,h6 {
    font-family:Arial Helvetica, sans-serif;
    color: #000000;
    font-weight: bold;
    font-style: normal;
}

strong
{
    font-family:Arial Helvetica, sans-serif;
    font-weight: bolder;
}
```