

Week Seven at Pandora Company Limited One Ring to Bring Them All

Implementing OAuth2

By I.M.C. Thisara (200652D)

Table of contents

Table of contents	2
Introduction	3
The Process	3
Steps to Create a Client Secret	3
Steps to implement the php website with OAuth2.0 login	5
Demonstration of the process.	9
Challenges Faced.....	10
Decisions Made.....	Error! Bookmark not defined.
References.	Error! Bookmark not defined.

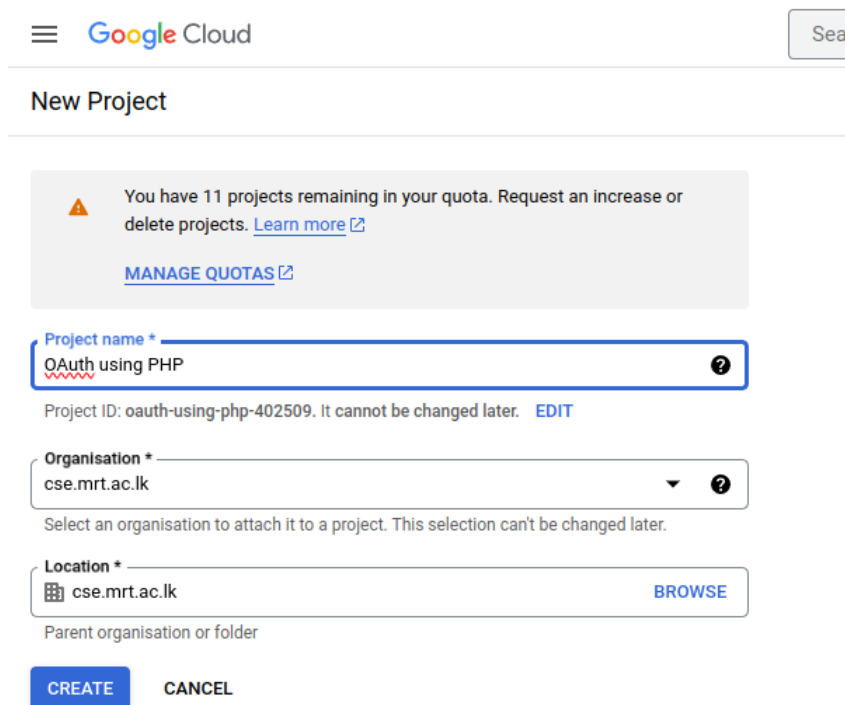
Introduction

This report outlines the steps taken to implement the OAuth2 authentication for the login system of Pandora Company Limited's PHP-based applications. The integration was carried out to enable streamlined user authentication using "Login with Google" functionality. This report documents the entire process, challenges met, and decisions made during the implementation.

The Process

Steps to Create a Client Secret (Google, 2023)

- Visit the Google Cloud Console at <https://console.cloud.google.com/> and log in with valid Google account credentials.
- Then create a new project specifically for the Pandora Company Limited application.



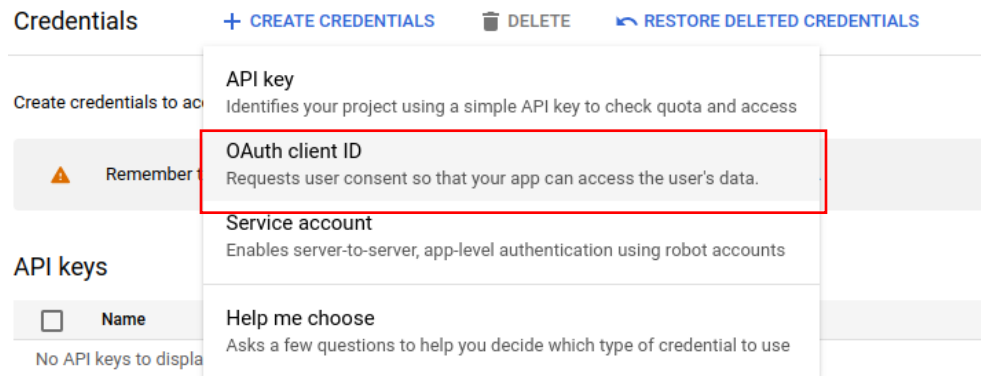
The screenshot shows the Google Cloud Console interface for creating a new project. At the top, there's a navigation bar with the Google Cloud logo and a search bar. Below it, the 'New Project' section is active. A warning box indicates that the user has 11 projects remaining in their quota and provides a link to 'MANAGE QUOTAS'. The main form contains three fields: 'Project name' (filled with 'OAuth using PHP'), 'Organisation' (filled with 'cse.mrt.ac.lk'), and 'Location' (filled with 'cse.mrt.ac.lk'). Below these fields, there are 'CREATE' and 'CANCEL' buttons.

- When created some APIs will be enabled.

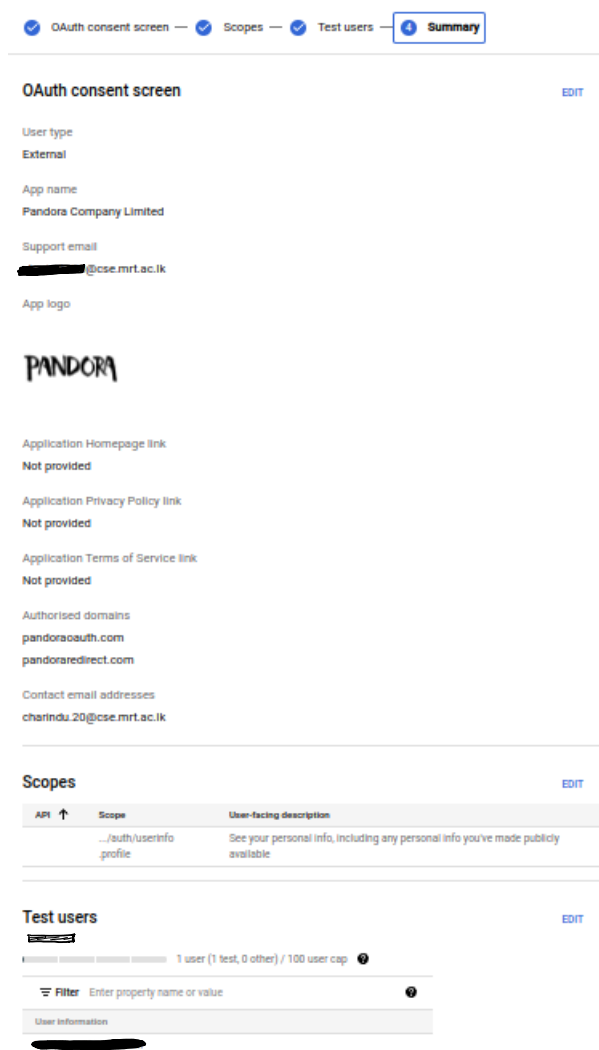


Name
BigQuery API
BigQuery Migration API
BigQuery Storage API
Cloud Datastore API
Cloud Logging API
Cloud Monitoring API
Cloud SQL
Cloud Storage
Cloud Storage API
Cloud Trace API
Google Cloud APIs
Google Cloud Storage JSON API
Service Management API
Service Usage API

- To create OAuth Client ID, go to the Credentials tab and click on "Create Credentials." And choose "OAuth client ID" as the credential type.

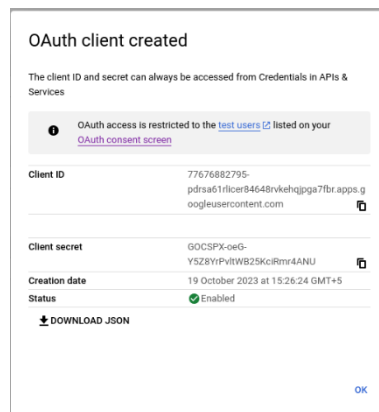


- Then configure the OAuth consent screen by supplying necessary details such as the application name, authorized domains, and developer contact information.



- Then again go to create credentials to set Application Type and Authorized Redirect URIs.
 - Application type: "Web application"

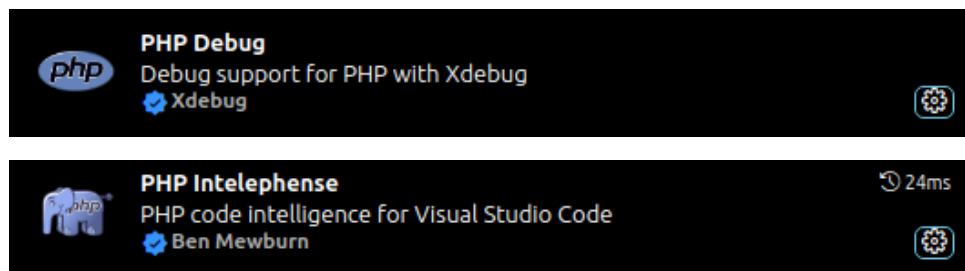
- Authorized redirect URIs: <http://localhost:3000/redirect.php>



- After that download, the JSON and we are ready to start creating the **php** login.

Steps to implement the php website with OAuth2.0 login

Here I am using VSCode with the following extensions for building and testing the APP.
(Google, google-api-php-client, 2023)



First Open a new folder and include the Downloaded JSON with client secret.

Then install composer by,

sudo apt install composer

Then create a file composer.json including the following details.

```
{
    "require": {
        "google/apiclient": "^2.15.0"
    },
    "scripts": {
        "pre-autoload-dump": "Google\\Task\\Composer::cleanup"
    }
}
```

To install required libraries, go to the directory having composer.json and,

composer install

To run the application, use the built in php server.

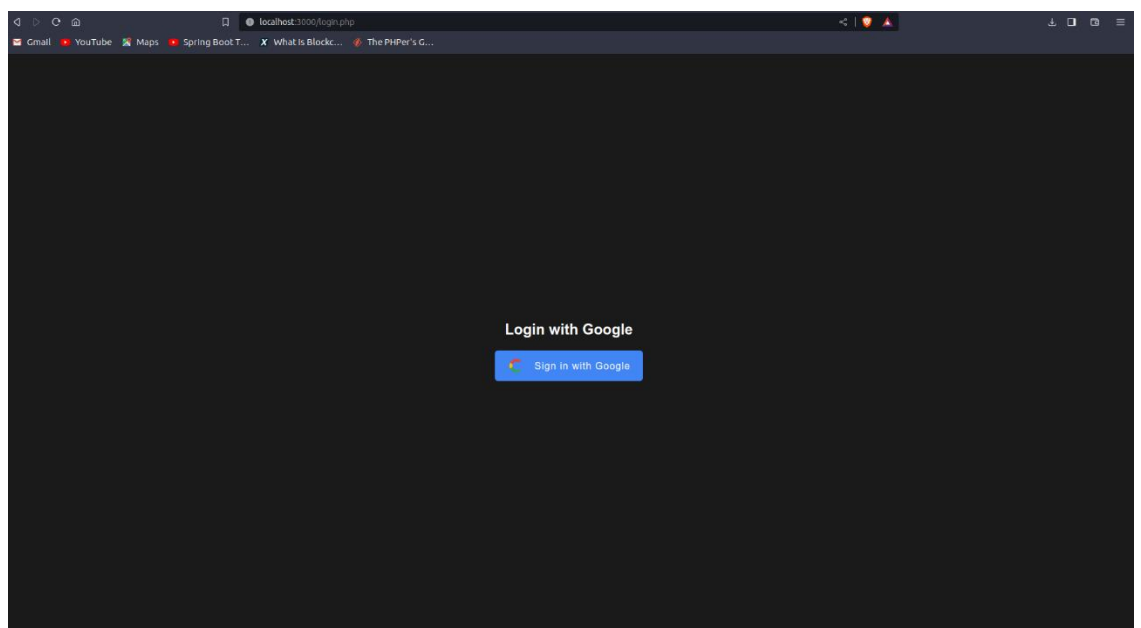
php -S localhost:3000 -t .

- Creating the "login.php" File

```
<body>
  <div class="login-container">
    <h2>Login with Google</h2>
    <a href="/redirect.php" class="login-btn">
      <div class="no_ku">
        <svg width="30" height="30" viewBox="0 0 24 24" fill="none" class="ah svg-icon" style="vertical-align: middle;">
          <g fill-rule="evenodd" clip-rule="evenodd">
            <path d="M20.64 12.2c0-.63-.06-1.25-.16-1.84H12v3.49h4.84a4.14 4.14 0 0 1-1.8 2.71v2.26h2.92a8.78 8.78 0 0 0 2.68-6.61z" fill="#4285F4"></path>
            <path d="M12 21a8.6 8.6 0 0 0 5.96-2.181-2.91-2.26a5.41 5.41 0 0 1-8.09-2.85h-3v2.33a9 9 0 0 12 21z" fill="#34A853"></path>
            <path d="M5.96 13.71a5.41 5.41 0 0 1 0-3.42v7.96h-3a9 9 0 0 0 8.0813-2.33z" fill="#FBBC05"></path>
            <path d="M12 6.58c1.32 0 2.545 3.44 1.3512 58-2.58A9 9 0 0 3.96 7.9613 2.33A5.36 5.36 0 0 1 12 6.6z" fill="#EA4335"></path>
          </g>
        </svg>
        <span class="button-text">Sign in with Google</span>
      </div>
    </a>
  </div>
</body>
```

This is a Simple login page with a redirect button to /redirect.php

This is how it appears.



- Implementing the "redirect.php" File

ChatGPT and instructions of some websites were used when creating this file.

```
<?php
require_once 'vendor/autoload.php'; // Load the Google API Client Library
session_start();

if (isset($_SESSION['user'])) {
    // If the 'user' key is already set in the session, redirect to the dashboard
    header('Location: dashboard.php');
    exit;
}

$client = new Google_Client();
$client->setAuthConfig('client_secret.json');
```

```

$client->addScope(Google_Service_Oauth2::USERINFO_PROFILE);

if (!isset($_GET['code'])) {
    // If no authorization code is present, initiate the Google login process
    $auth_url = $client->createAuthUrl();
    header('Location: ' . filter_var($auth_url, FILTER_SANITIZE_URL));
    exit;
} else {
    // Exchange the authorization code for an access token
    $client->fetchAccessTokenWithAuthCode($_GET['code']);
    $_SESSION['access_token'] = $client->getAccessToken();

    if (isset($_SESSION['access_token'])) {
        // If access token is available, fetch the user's information and store it in the session
        $client->setAccessToken($_SESSION['access_token']);
        $oauth2 = new Google_Service_Oauth2($client);
        $userInfo = $oauth2->userinfo->get();
        $_SESSION['user'] = $userInfo;

        // Redirect to the dashboard with the user's name
        header('Location: dashboard.php');
        exit;
    } else {
        // Handle errors or unsuccessful authentication
        $redirect_uri = 'http://' . $_SERVER['HTTP_HOST'] . '/redirect.php';
        header('Location: ' . filter_var($redirect_uri, FILTER_SANITIZE_URL));
        exit;
    }
}
?>

```

This script will act like this. It first checks if any user is already logged in. If so, it will redirect to the dashboard. Otherwise, it starts the OAuth process with google.

The client secret file will be used when setting up the connection with google.

```

$client->createAuthUrl();

```

Then the user will be redirected to the google server to get user consent for supplying profile information. Then google authentication server will do its process and send the response with the **Authentication code** with the redirect URI as mentioned (<http://localhost:3000/redirect.php>)

Then the same script will run, and it will detect the session with the name “**code**”. Then then it will connect google again directly to get the access token with “**Authentication Code**” and the “**Client secrets**”.

```
$client->fetchAccessTokenWithAuthCode($_GET['code']);
```

After that, if the “**Access token**” is received, client will request user information with that and save it under the cookie “user”. Else, it will redirect again to the [redirect.php](http://localhost:3000/redirect.php) and try the process from the beginning.

- Building the "dashboard.php" File

What this does is check for the cookie “**user**” and if it exists then extract **user profile information** to show.

Hello <username>. Welcome to Pandora Company Limited.

```
<?php
require_once 'vendor/autoload.php';

session_start();

if (isset($_SESSION['user'])) {
    $userInfo = $_SESSION['user'];
    $userName = $userInfo->name;

    echo '<h1>Hello, ' . $userName . '!</h1>';
    echo '<p>Welcome to Pandora Company Limited.</p>';
    echo '<form action="" method="post">';
    echo '<input type="submit" class="logout-btn" name="logout" value="Logout">';
    echo '</form>';

    // Handle logout
    if (isset($_POST['logout'])) {
        // Unset all of the session variables
        $_SESSION = array();

        // Destroy the session
        session_destroy();

        // Redirect to the login page or any other appropriate location after logout
        header("Location: login.php");

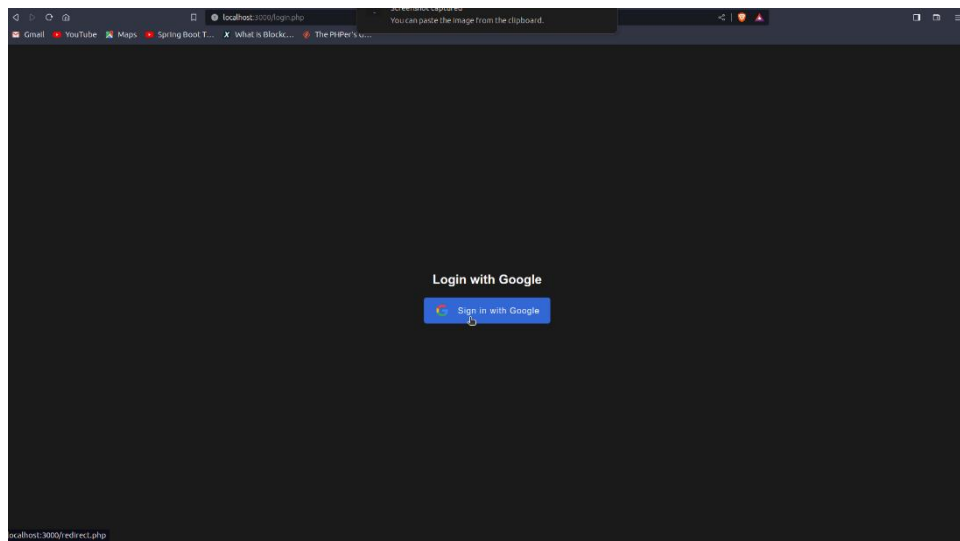
        exit;
    }
} else {
    echo '<p>User name not found.</p>';
}

?>
```

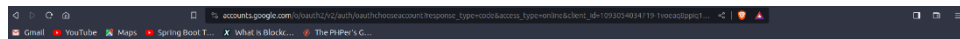
Additionally, I thought of adding a logout button as well. Because in **php** all the cookies are stored in the server. So, I couldn't remove the cookie “user” from the browser. Therefore, a logout button was added to delete the session unsetting all variables under it.

Demonstration of the process.

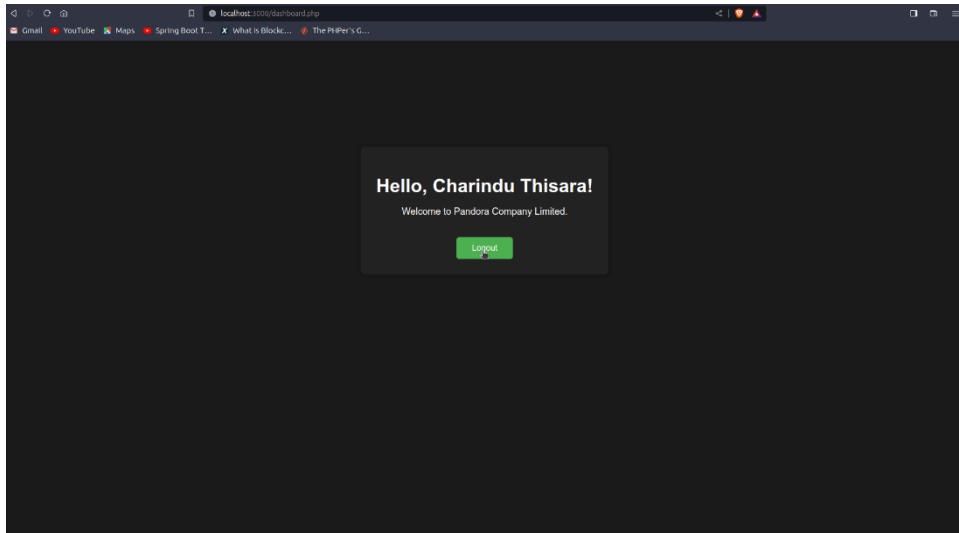
1)



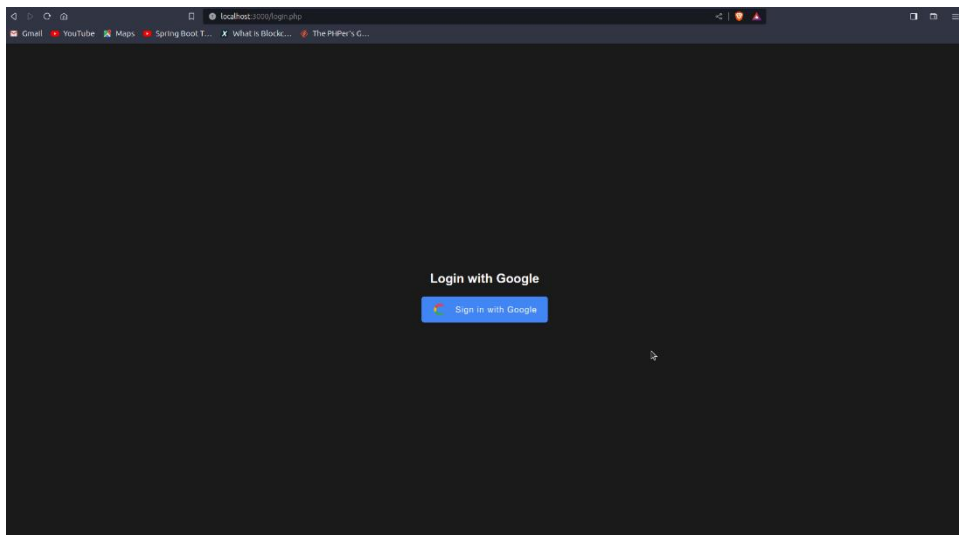
2)



3)



4) logout > back to login



Challenges Faced & Decisions Made

During the implementation process, the following challenges were met:

1. Understanding OAuth2 Workflow: It took a while to understand the intricacies of the OAuth2 workflow and the specific requirements for integrating Google OAuth.
2. Configuring Redirect URIs: Ensuring the correct configuration of authorized redirect URIs was challenging due to the requirement for precise matching between the redirect URI in the code and the one set in the Google Cloud Console.
3. PHP was not a familiar language to me. So, it took a while to figure out how to run the files under a server.

To address the challenges faced during implementation, the following decisions were made:

1. Thorough Research: Extensive research was conducted to gain a comprehensive understanding of the OAuth2 authentication process and its integration with Google services.
2. Debugging and Testing: Testing and debugging were made easier by using the built-in **PHP development** server instead of the **Apache server**. (PHP server can be run in the current directory without configuration)
3. Used ChatGPT to generate and describe the meaning of the PHP codes.

Conclusion

This report provides an overview of the OAuth2 implementation process for Pandora Company Limited's PHP-based application, along with the steps to create a client secret. It also highlights the challenges met and the decisions made to successfully integrate the "Login with Google" functionality.

References

- Google. (2023). *google-api-php-client*. Retrieved from Google APIs:
<https://github.com/googleapis/google-api-php-client>
- Google. (2023, 10 18). *Using OAuth 2.0 to Access Google APIs*. Retrieved from Google Identity:
<https://developers.google.com/identity/protocols/oauth2>