

# Week Seven at Pandora Company Limited

## One Ring to Bring Them All

Name :- Nonis S.S.L

Index No :- 210439P

### 1. Introduction

Pandora Company Limited aims to enhance application security and user experience by integrating its PHP-based applications with a centralized user management system using OAuth2. This project focuses on implementing "Login with Google" functionality to streamline authentication. The application has three components: a login page, a redirect handler, and a dashboard to greet authenticated users. This report outlines the implementation process, challenges faced, and key decisions made to achieve a secure and efficient login solution.

### 2. Tools and technologies used :-

- php-8.4.1
- Google API PHP Client Library
- Composer
- Web server

### 3. Implementation Steps

#### • Install php

Go to the [official PHP website](#) and download the latest "Non-Thread Safe" ZIP package for your Windows version. Extract the ZIP file and install it in a directory. Open **System Properties > Environment Variables**. Edit the PATH variable. Install and set up the XAMPP server to generate php outputs. Use VS Code as the text editor with necessary extensions.

VS17 x64 Thread Safe (2024-Nov-20 11:18:26)

- [Zip](#) [32.33MB]  
sha256: 020d265aa9aa2759cd527c16ce4749cb7b8b7f70ec13cac9237f428da2eb6ee
- [Debug Pack](#) [31.03MB]  
sha256: 5fb9f657715931dabe055753eec99c71416b9d34ce20fe9e3424d497782bffd
- [Development package \(SDK to develop PHP extensions\)](#) [1.35MB]  
sha256: 30327f2e746de1530c547e3ac71078f4c5e607b85ac0f60a4b154481c14892f0

XAMPP Apache + MariaDB + PHP + Perl

#### What is XAMPP?

XAMPP is the most popular PHP development environment

XAMPP is a completely free, easy to install Apache distribution containing MariaDB, PHP, and Perl. The XAMPP open source package has been set up to be incredibly easy to install and to use.



Download

[Click here for other versions](#)

XAMPP for Windows  
8.2.12 (PHP 8.2.12)

XAMPP for Linux  
8.2.12 (PHP 8.2.12)

XAMPP for OS X  
8.2.4 (PHP 8.2.4)

#### • Setting Up Google Cloud Project

Log in to the Google Cloud Console and create a new project. Enable the "Google People API" or "OAuth Consent Screen" for the project. Navigate to "APIs & Services > Credentials". Create an OAuth2 Client ID, specifying the application type (Web Application). Configure the redirect URI to point to your application (e.g., <http://localhost/Assignment.php>). Note down the Client ID and Client Secret.

## Additional information

Client ID	17265268553- td63tlu6gj8hrubm8d1452vloskn2pvn.ap ps.googleusercontent.com
Creation date	1 December 2024 at 23:17:18 GMT+5

## Client secrets

If you are in the process of changing client secrets, you can manually rotate them without downtime. [Learn more](#)

Client secret	GOCSPX- 4QreR5sJ6xk0t7Cy3ciRPuQXk_-a
Creation date	1 December 2024 at 23:17:18 GMT+5
Status	Enabled

## Authorised redirect URIs [?](#)

For use with requests from a web server

URIs 1 \*  
http://localhost/Assignment.php

[+ ADD URI](#)

Note: It may take five minutes to a few hours for settings to take effect

- **Installing Dependencies**

First, ensure that **Composer**, a PHP dependency manager, is installed on your system. If not, download and install it from [getcomposer.org](#). Next, use Composer to add the **Google API Client Library** by running the command.

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

This will download and set up the necessary library for OAuth2 integration. Finally, include Composer's autoloader in your PHP files to make the library available for use by adding:

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

This ensures that all dependencies are properly loaded and ready for use in your project.

- **Creating PHP Files**

1. **login.php** - This file serves as the starting point of the application, displaying the "Login with Google" button. Initialize the Google Client object. Configure the client with the **Client ID**, **Client Secret**, and **Redirect URI**. Generate an OAuth2 authorization URL. Display the "Login with Google" button linking to the authorization URL.

```

Login.php X Dashboard.php 2 index.php redirect.php

Login.php > html > head > style > body
1  <?php
2  session_start();
3  require_once 'vendor/autoload.php'; // Include Composer autoload file
4
5  // Initialize Google OAuth configuration
6  $clientID = "17265268553-td63tlu6gj8hrubm8d1452vloskn2pvn.apps.googleusercontent.com";
7  $clientSecret = "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";
8  $redirectUri = "http://localhost/Assignment.php"; // Update with your redirect URI
9
10 // Create Google Client
11 $client = new Google_Client();
12 $client->setClientId($clientID);
13 $client->setClientSecret($clientSecret);
14 $client->setRedirectUri($redirectUri);
15 $client->addScope("email");
16 $client->addScope("profile");
17
18 // If user is already logged in, redirect to dashboard
19 if (isset($_SESSION['access_token']) && $_SESSION['access_token']) {
20     header("Location: dashboard.php");
21     exit;
22 }
23
24 // If not logged in, show the login page
25 ?>
26 <!doctype html>
27 <html lang="en">
28 <head>
29     <title>Login with Google</title>
30     <meta charset="utf-8">
31     <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

```

- 2. redirect.php** - This file handles the response from Google after the user authenticates. Retrieve the code parameter from Google's response. Exchange the code for an access token using the Google API Client library. Use the token to fetch user details.

```

redirect.php X

redirect.php > ...
1  <?php
2  session_start();
3  require_once 'vendor/autoload.php'; // Include Composer autoload file
4
5  // Initialize Google OAuth configuration
6  $clientID = "17265268553-td63tlu6gj8hrubm8d1452vloskn2pvn.apps.googleusercontent.com";
7  $clientSecret = "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";
8  $redirectUri = "http://localhost/Assignment.php"; // Must match with Google Developer Console
9
10 // Create Google Client
11 $client = new Google_Client();
12 $client->setClientId($clientID);
13 $client->setClientSecret($clientSecret);
14 $client->setRedirectUri($redirectUri);
15 $client->addScope("email");
16 $client->addScope("profile");
17
18 // Check if the code exists from the OAuth process
19 if (isset($_GET['code'])) {
20     try {
21         $token = $client->fetchAccessTokenWithAuthCode($_GET['code']);
22         if (isset($token['error'])) {
23             throw new Exception('Error fetching token: ' . $token['error_description']);
24         }
25
26         // Save the token to the session
27         $_SESSION['access_token'] = $token;
28
29         // Redirect to dashboard
30         header('Location: dashboard.php');
31         exit;
32     } catch (Exception $e) {
33         echo "Failed to log in: " . $e->getMessage();
34         exit;
35     }
36 }
37 } else {
38     // Redirect to login if the code is not present
39     header('Location: login.php');
40     exit;
41 }
42 ?>

```

- 3. dashboard.php** - This is the landing page after a successful login. Display a personalized greeting message using session variables. Ensure proper session handling for security.

```

1 <?php
2 session_start();
3 require_once 'vendor/autoload.php'; // Include Composer autoload file
4
5 // Initialize Google OAuth configuration
6 $clientID = "17265268553-td63tlu6gj8hrubm8d1452vloskn2pvn.apps.googleusercontent.com";
7 $clientSecret = "GOCSPX-4QreR5sJ6xk0t7Cy3ciRPuQXK_a";
8 $redirectUri = "http://localhost/Assignment.php"; // Update with your redirect URI
9
10 // Create Google Client
11 $client = new Google_Client();
12 $client->setClientId($clientID);
13 $client->setClientSecret($clientSecret);
14 $client->setRedirectUri($redirectUri);
15 $client->addScope("email");
16 $client->addScope("profile");
17
18 // Handle Logout
19 if (isset($_GET['logout'])) {
20     session_destroy();
21     header("Location: login.php");
22     exit;
23 }
24
25 // Handle logged-in user
26 if (isset($_SESSION['access_token']) && $_SESSION['access_token']) {
27     $client->setAccessToken($_SESSION['access_token']);
28
29     // Fetch user profile info
30     $google_oauth = new Google_Service_Oauth2($client);

```

#### 4. Challenges Faced

- Configuring the Redirect URI: Ensuring that the redirect URI specified in the Google Cloud Console matches the application endpoint exactly. Any mismatch (e.g., missing http://localhost or incorrect file path) results in authentication errors.
- Library or PHP Extension Issues: Encountered errors like Undefined type 'Google\Client' or Google\Service\Oauth2. These were caused by missing dependencies or improperly installed PHP libraries, which required troubleshooting the Composer setup and verifying the presence of required extensions.
- Managing Session Tokens Securely: Handling user session tokens securely to maintain user authentication across pages without exposing sensitive information or risking session hijacking. This required careful use of PHP sessions and storage mechanisms.

```

- Locking paragonie/constant_time_encoding (v3.0.0)
- Locking paragonie/random_compat (v9.99.100)
- Locking phpseclib/phpseclib (3.0.42)
- Locking psr/cache (3.0.0)
- Locking psr/http-client (1.0.3)
- Locking psr/http-factory (1.1.0)
- Locking psr/http-message (2.0)
- Locking psr/log (3.0.2)
- Locking ralouphie/getallheaders (3.0.3)
- Locking symfony/deprecation-contracts (v3.5.1)
writing lock file
installing dependencies from lock file (including require-dev)
package operations: 18 installs, 0 updates, 0 removals
Failed to download paragonie/random_compat from dist: The zip extension and unzip/7z command
are both missing, skipping.
the php.ini used by your command-line PHP is: C:\Program Files\php-8.4.1-Win32-vs17-x64\php.ini
Now trying to download from source

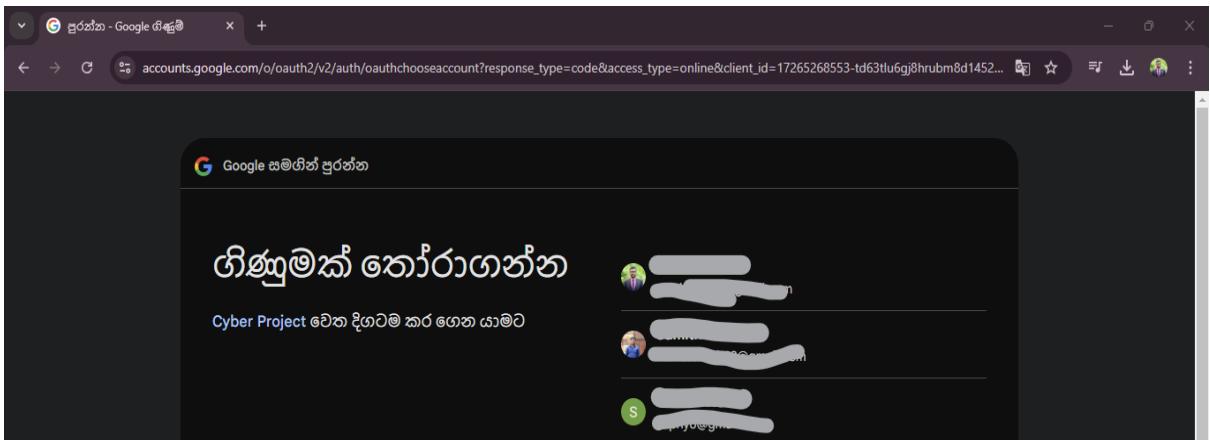
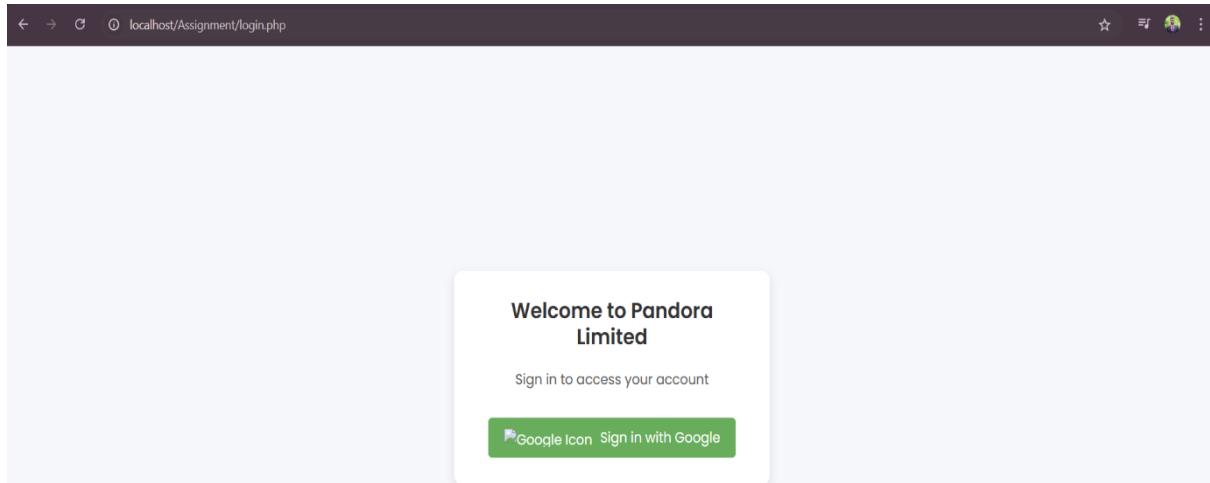
n GitDownloader.php line 82:
git was not found in your PATH, skipping source download

require [--dev] [--dry-run] [--prefer-source] [--prefer-dist] [--prefer-install PREFER-INSTALL]
--fixed] [--no-suggest] [--no-progress] [--no-update] [--no-install] [--no-audit] [--audit-forma
AUDIT_FORMAT] [--update-newer] [--update-with-dependencies] [--update-with-all-dependencies]

```

## 5. Conclusion –

The integration of the "Login with Google" functionality using OAuth2 was successfully implemented, providing a seamless and secure user authentication mechanism for Pandora Company Limited's applications. Through the creation of the login.php, redirect.php, and dashboard.php files, users can now authenticate via Google and access a personalized dashboard. The project highlighted the importance of understanding OAuth2 workflows, dependency management, and secure session handling in web applications. This implementation not only enhances the user experience by providing single sign-on capabilities but also lays the foundation for further improvements in user management and application security.



## 6. References –

- Official PHP download : <https://windows.php.net/download/>
- Setting Up Google Cloud Project : <https://console.cloud.google.com/>
- Composer : <https://getcomposer.org/download/>
- <https://www.youtube.com/watch?v=zZ6vybT1HQs&t=13777s>
- <https://www.youtube.com/watch?v=OpvcXKn20Os>
- Example Source Codes: <https://techareatutorials.com/login-w...>