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

Name:Lahiru Randika Index: 210527J

Table of Contents

Introduction

This report outlines the steps taken to implement OAuth2 authentication for Pandora Company Limited's PHP-based login application. The goal was to allow users to log in using their Google accounts while securely managing sensitive credentials.

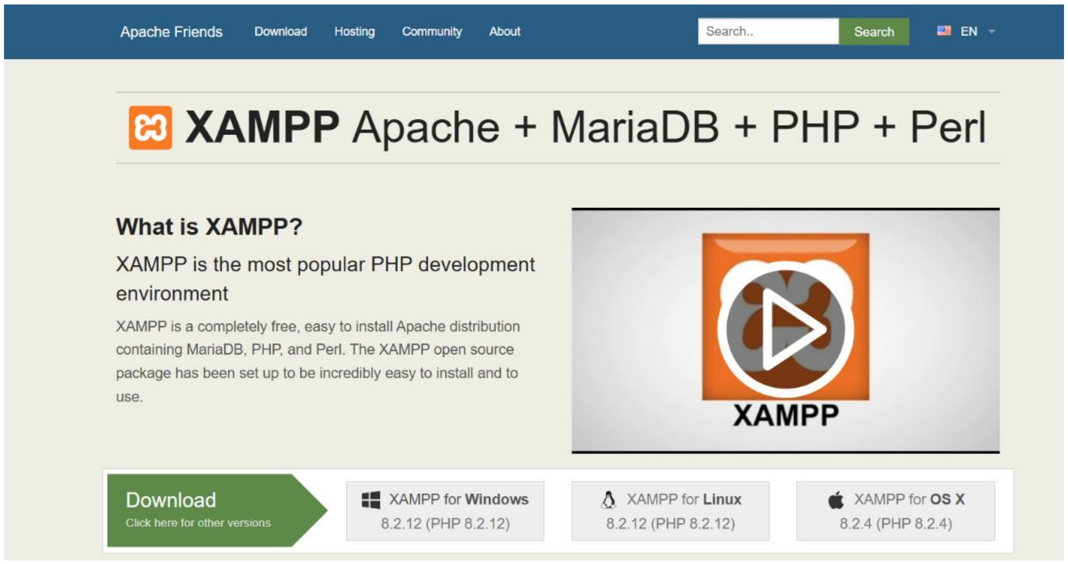
The process began with setting up the local development environment using XAMPP and installing required dependencies via Composer. I then configured Google Cloud to generate the necessary OAuth2 credentials. After that, I integrated the OAuth2 flow into the PHP application, ensuring proper session management and access to user data.

Challenges faced included configuring the Google Cloud setup, resolving OAuth client errors by creating a new project and client, handling constant refresh issues by adding a live server extension, and overcoming a lack of PHP knowledge through online resources. The result is a fully functional and secure Google login system for users to access a personalized dashboard.

Setting Up the Development Environment

Step 1: Installing XAMPP and Setting Up the Local Server

1. **Download and Install XAMPP:**
   I downloaded and installed XAMPP to set up a local web server environment from https://www.apachefriends.org/index.html. This package includes Apache, MySQL, and PHP, which are required for the application.

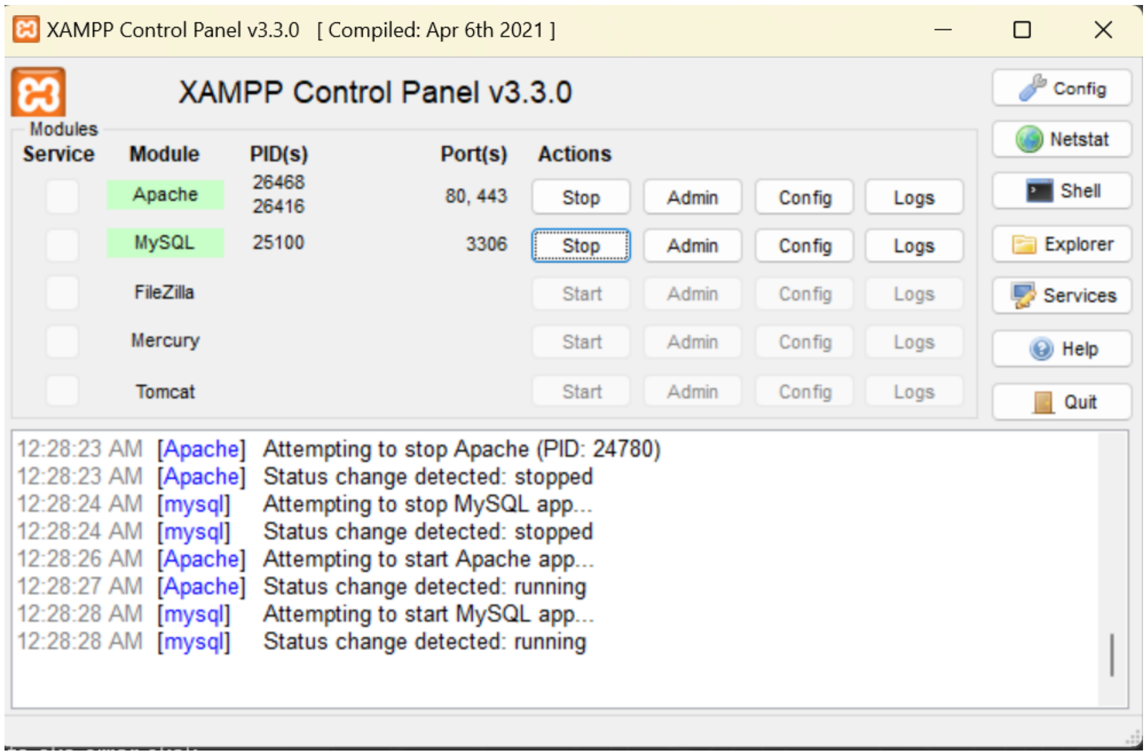

Apache Friends
Download
Hosting
Community
About
Search..
Search
EN
XAMPP Apache + MariaDB + PHP + Perl
XAMPP
Download
Click here for other versions
XAMPP for Windows 8.2.12 (PHP 8.2.12)
XAMPP for Linux 8.2.12 (PHP 8.2.12)
XAMPP for OS X 8.2.4 (PHP 8.2.4)

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.

2. **Starting the Server:**
   After installation, I launched the XAMPP Control Panel and started both Apache and MySQL services to set up the local server and database.


XAMPP Control Panel v3.3.0 [Compiled: Apr 6th 2021]
XAMPP Control Panel v3.3.0
Modules
Config
Netstat
Shell
Explorer
Services
Help
Quit

Service	Module	PID(s)	Port(s)	Actions			
	Apache	26468 26416	80, 443	Stop	Admin	Config	Logs
	MySQL	25100	3306	Stop	Admin	Config	Logs
	FileZilla			Start	Admin	Config	Logs
	Mercury			Start	Admin	Config	Logs
	Tomcat			Start	Admin	Config	Logs

```
12:28:23 AM  [Apache]  Attempting to stop Apache (PID: 24780)
12:28:23 AM  [Apache]  Status change detected: stopped
12:28:24 AM  [mysql]   Attempting to stop MySQL app...
12:28:24 AM  [mysql]   Status change detected: stopped
12:28:26 AM  [Apache]  Attempting to start Apache app...
12:28:27 AM  [Apache]  Status change detected: running
12:28:28 AM  [mysql]   Attempting to start MySQL app...
12:28:28 AM  [mysql]   Status change detected: running
```

3. **Creating the Project Folder:**
   I created a new project folder named website, inside the XAMPP htdocs directory (xampp\htdocs\website) and moved the project files there.

Step 2: Setting Up Composer and Installing Dependencies

1. **Installing Composer:**
   I installed Composer to manage PHP dependencies. Composer is a dependency manager that simplifies the process of installing and updating libraries.

Composer-Setup.exe	12/1/2024 3:21 PM	Application

2. **Creating the composer.json File:**
   I ran the following command to install the necessary libraries for Google OAuth authentication:

   PS C:\xampp\htdocs\website> composer require google/apiclient

   google/apiclient:

   - This is the official Google API Client Library for PHP. It provides a set of tools for interacting with Google APIs, such as OAuth2 authentication, accessing Google services (like Google Drive, Gmail, or Google Calendar), and more. It simplifies the process of integrating Google services into your PHP application.

Configuring Google OAuth2 Integration

Step 1: Setting Up Google Cloud Console

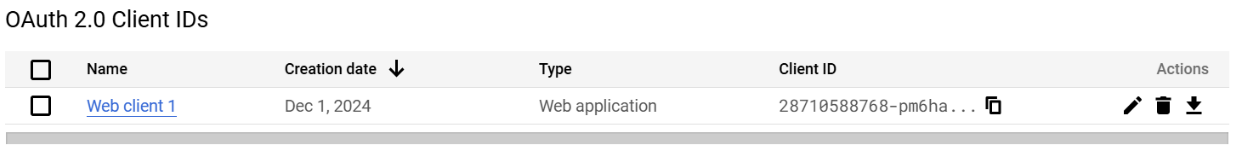
1. **Creating a Google Cloud Project:**
   I navigated to the Google Cloud Console and created a new project called PandoraLoginPage.

2. **Enabling Google APIs:**
   Under **APIs & Services**, I enabled the "Google+ API" and the OAuth2 API to authenticate users with Google accounts.

3. **Setting Up OAuth Consent Screen:**
   In the Google Cloud Console, I configured the OAuth consent screen with the following details:

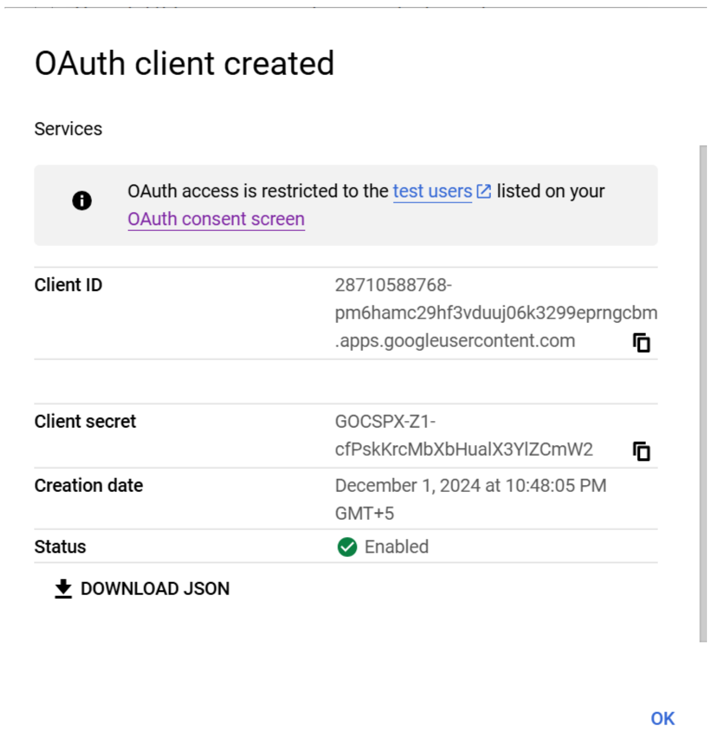
   - Application Name: *PandoraLoginPage*
   - Authorized redirect URI: http://localhost/website/redirect.php
   - Add other info like my Gmail etc.

4. **Creating OAuth2 Credentials:**
   I generated a new OAuth 2.0 client ID and secret from the Google Cloud Console, and saved the **Client ID** and **Client Secret** values.



OAuth 2.0 Client IDs

	Name	Creation date	Type	Client ID	Actions
☐	Web client 1	Dec 1, 2024	Web application	28710588768-pm6ha...	



OAuth client created

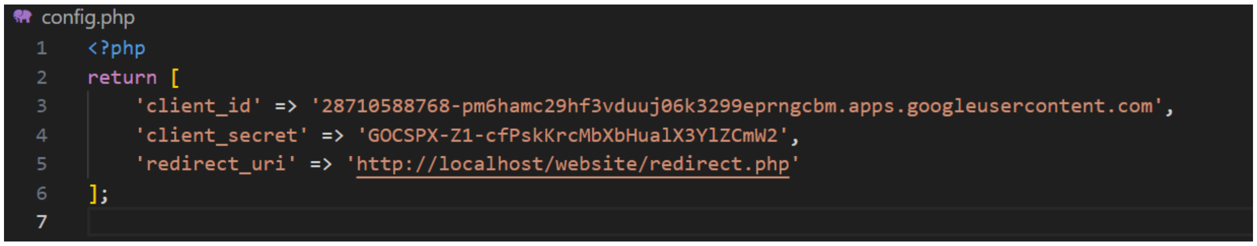
Services

OAuth access is restricted to the test users listed on your OAuth consent screen

Client ID	28710588768-pm6hamc29hf3vduuj06k3299eprngcbm.apps.googleusercontent.com
Client secret	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
Creation date	December 1, 2024 at 10:48:05 PM GMT+5
Status	Enabled

Step 2: Configuring the PHP Application for OAuth2

1. **Creating the config.php File:**
   Instead of using a .env file, I created a config.php file to store the Google OAuth credentials. This file is included in the application to securely load the credentials:

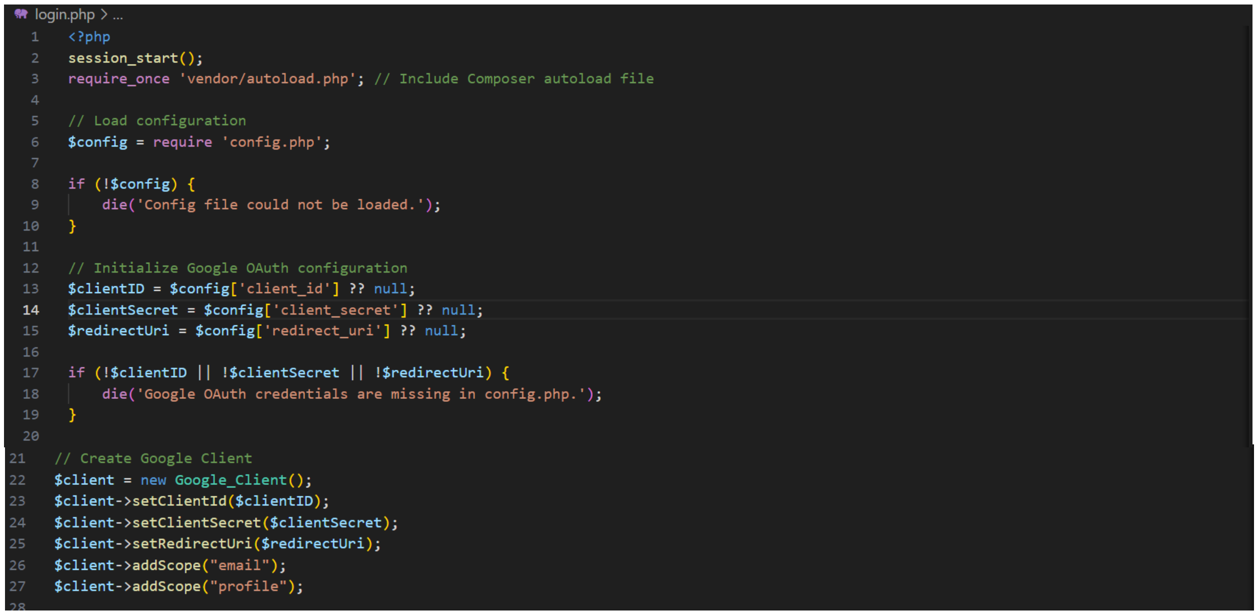


config.php

```php
<?php
return [
    'client_id' => '28710588768-pm6hamc29hf3vduuj06k3299eprngcbm.apps.googleusercontent.com',
    'client_secret' => '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',
    'redirect_uri' => 'http://localhost/website/redirect.php'
];

```

This configuration file allows me to load the credentials securely without hardcoding them into each script.

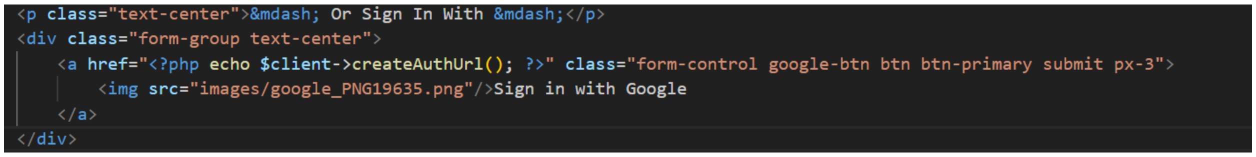
2. **Initializing Google OAuth Client in login.php:**
   In the login.php file, I set up the Google Client and integrated it with the config.php file:



login.php > ...

```php
<?php
session_start();
require_once 'vendor/autoload.php'; // Include Composer autoload file

// Load configuration
$config = require 'config.php';

if (!$config) {
    die('Config file could not be loaded.');
}

// Initialize Google OAuth configuration
$clientID = $config['client_id'] ?? null;
$clientSecret = $config['client_secret'] ?? null;
$redirectUri = $config['redirect_uri'] ?? null;

if (!$clientID || !$clientSecret || !$redirectUri) {
    die('Google OAuth credentials are missing in config.php.');
}

// Create Google Client
$client = new Google_Client();
$client->setClientId($clientID);
$client->setClientSecret($clientSecret);
$client->setRedirectUri($redirectUri);
$client->addScope("email");
$client->addScope("profile");
```

3. **Creating the Login Link:**
   I generated the Google OAuth login URL and displayed it as a login button:

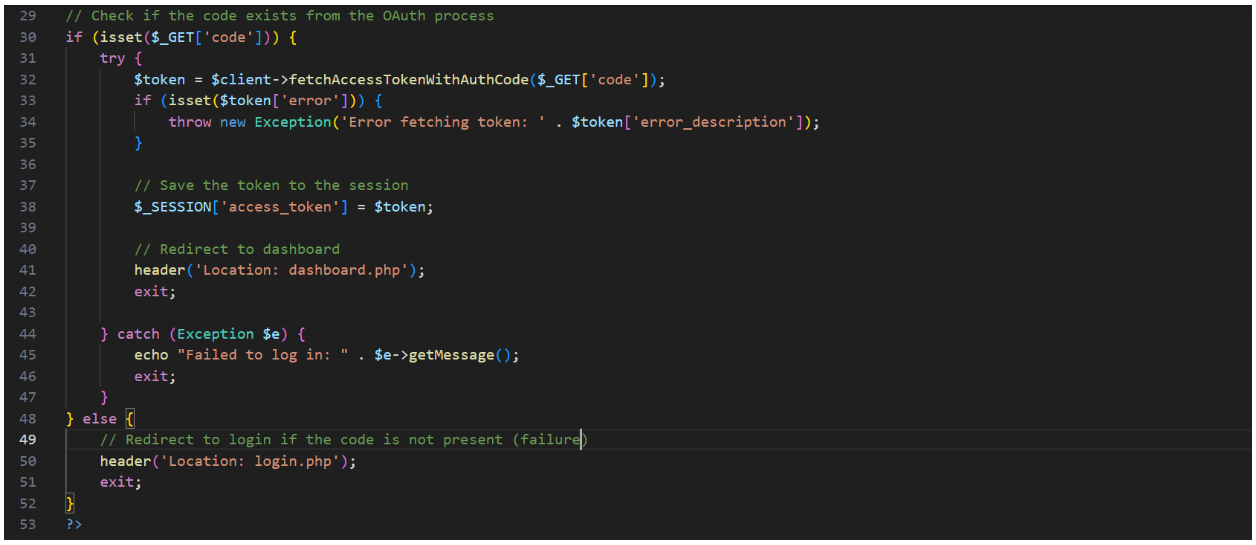


```html
<p class="text-center">&mdash; Or Sign In With &mdash;</p>
<div class="form-group text-center">
    <a href="<?php echo $client->createAuthUrl(); ?>" class="form-control google-btn btn btn-primary submit px-3">
        <img src="images/google_PNG19635.png"/>Sign in with Google
    </a>
</div>
```

Handling Authentication and User Data

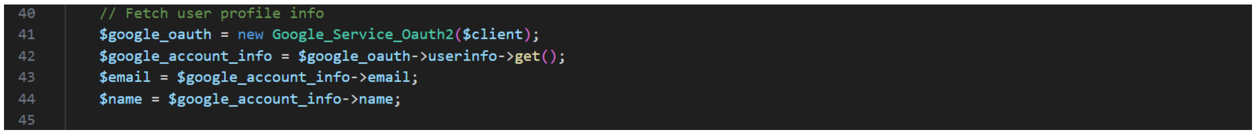
Step 1: Redirecting and Handling Callback

1. **Handling Callback in redirect.php:**
   After the user logs in via Google, they are redirected to the redirect.php page. This page captures the authorization code and exchanges it for an access token:



```php
// Check if the code exists from the OAuth process
if (isset($_GET['code'])) {
    try {
        $token = $client->fetchAccessTokenWithAuthCode($_GET['code']);
        if (isset($token['error'])) {
            throw new Exception('Error fetching token: ' . $token['error_description']);
        }

        // Save the token to the session
        $_SESSION['access_token'] = $token;

        // Redirect to dashboard
        header('Location: dashboard.php');
        exit;

    } catch (Exception $e) {
        echo "Failed to log in: " . $e->getMessage();
        exit;
    }
} else {
    // Redirect to login if the code is not present (failure)
    header('Location: login.php');
    exit;
}
?>
```

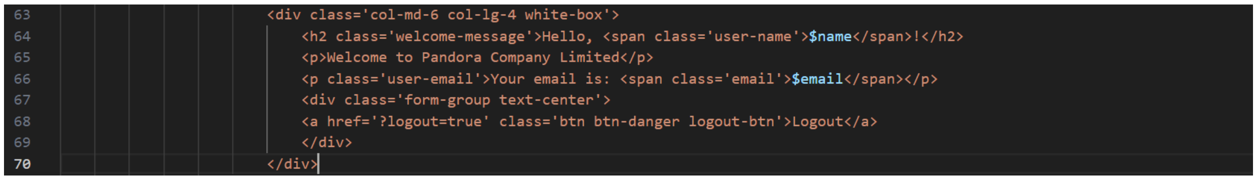
2. **Fetching User Information:**
   Once the access token is retrieved, I used it to fetch the user's profile information (name and email):



```php
    // Fetch user profile info
    $google_oauth = new Google_Service_Oauth2($client);
    $google_account_info = $google_oauth->userinfo->get();
    $email = $google_account_info->email;
    $name = $google_account_info->name;

```

Step 2: Designing the Dashboard

1. **Displaying User Information on the Dashboard:**
   After successful authentication, the user is redirected to the dashboard.php page, where their details are displayed:



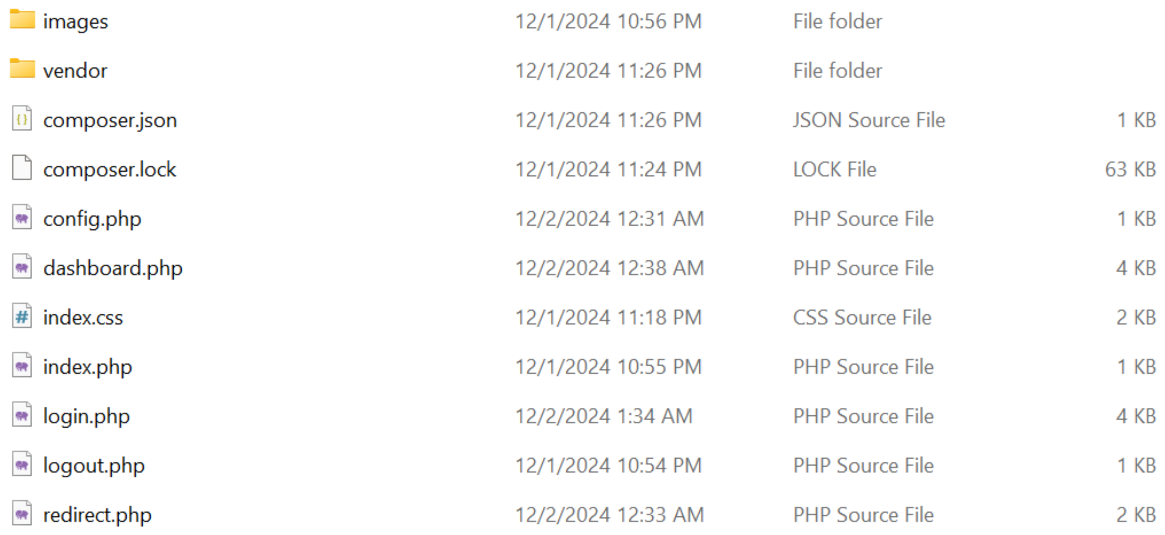
```html
<div class='col-md-6 col-lg-4 white-box'>
    <h2 class='welcome-message'>Hello, <span class='user-name'>$name</span>!</h2>
    <p>Welcome to Pandora Company Limited</p>
    <p class='user-email'>Your email is: <span class='email'>$email</span></p>
    <div class='form-group text-center'>
    <a href='?logout=true' class='btn btn-danger logout-btn'>Logout</a>
    </div>
</div>
```

2. **Styling the Dashboard:**
   I applied basic CSS to style the dashboard and make it visually appealing:

All Codes and ScreenShots

1. Folder Structure

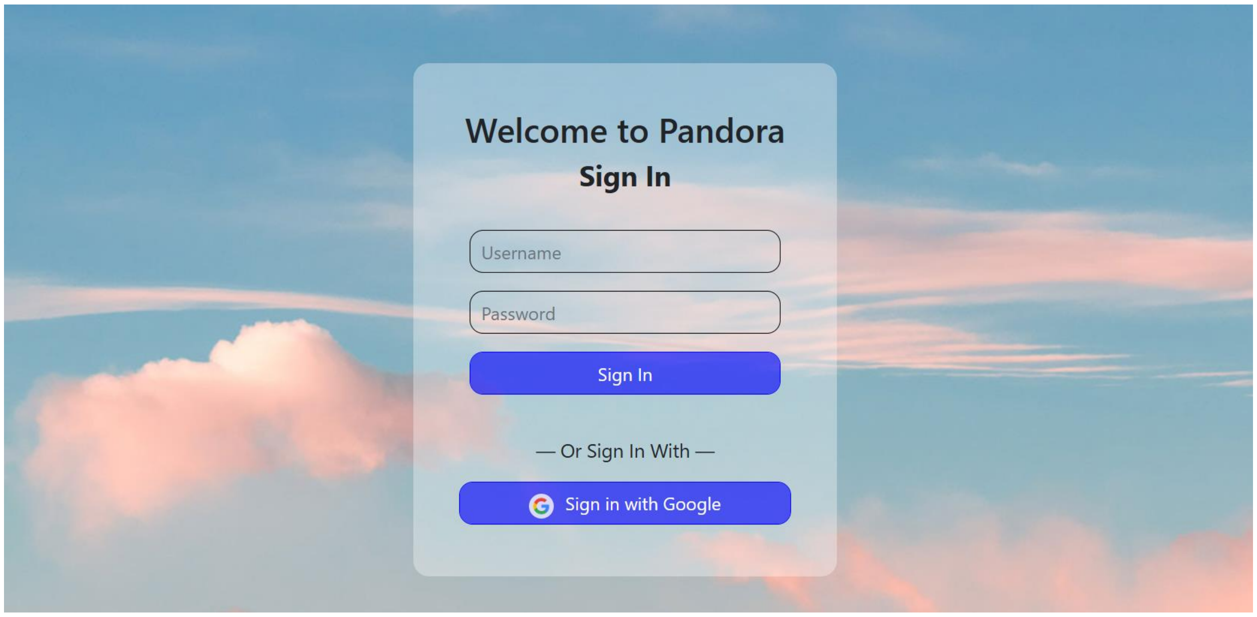
C:/xampp/htdocs/website



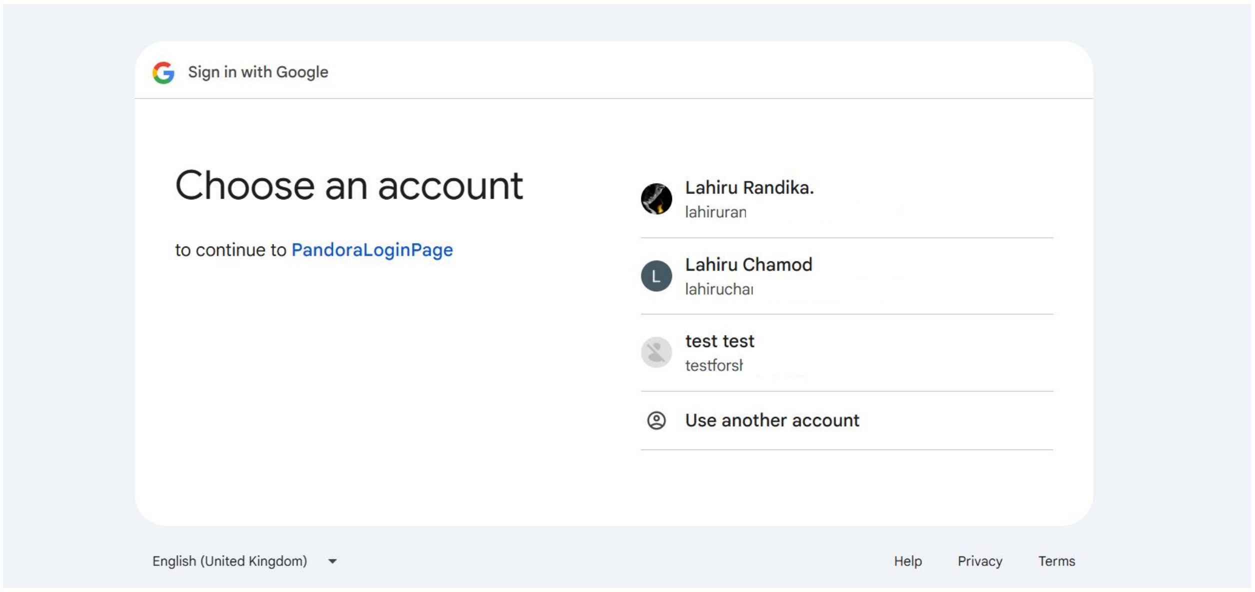
images	12/1/2024 10:56 PM	File folder	
vendor	12/1/2024 11:26 PM	File folder	
composer.json	12/1/2024 11:26 PM	JSON Source File	1 KB
composer.lock	12/1/2024 11:24 PM	LOCK File	63 KB
config.php	12/2/2024 12:31 AM	PHP Source File	1 KB
dashboard.php	12/2/2024 12:38 AM	PHP Source File	4 KB
index.css	12/1/2024 11:18 PM	CSS Source File	2 KB
index.php	12/1/2024 10:55 PM	PHP Source File	1 KB
login.php	12/2/2024 1:34 AM	PHP Source File	4 KB
logout.php	12/1/2024 10:54 PM	PHP Source File	1 KB
redirect.php	12/2/2024 12:33 AM	PHP Source File	2 KB

2. Web Pages

http://localhost/website/login.php



Welcome to Pandora
Sign In


Sign in with Google

Choose an account

to continue to PandoraLoginPage

Lahiru Randika.
lahiruran

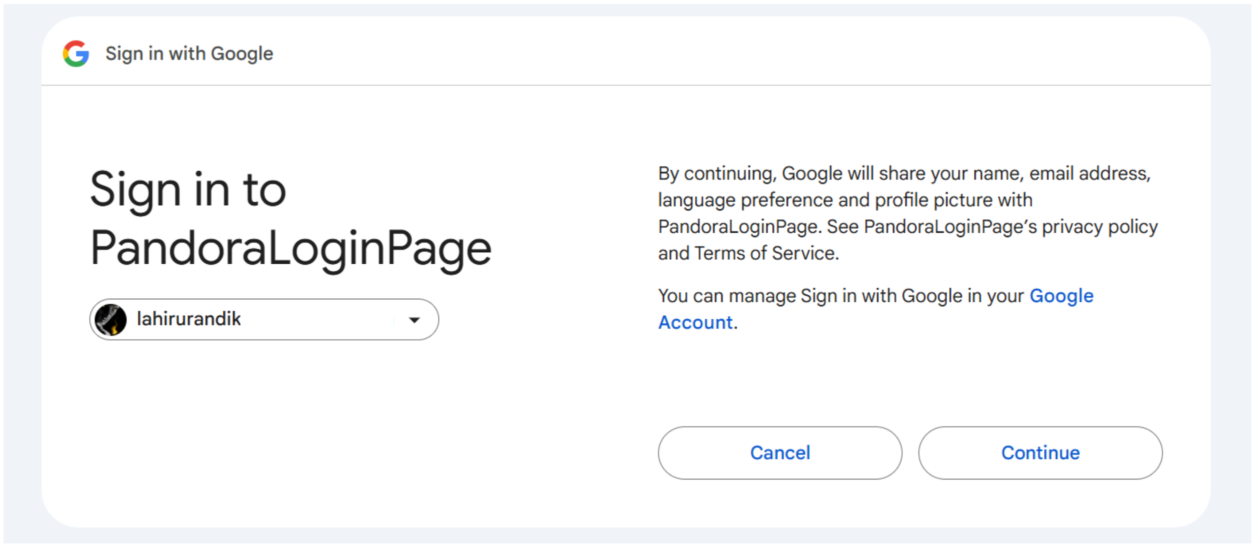
Lahiru Chamod
lahiruchai

test test
testforst

Use another account

English (United Kingdom)

Help Privacy Terms


Sign in with Google

Sign in to PandoraLoginPage

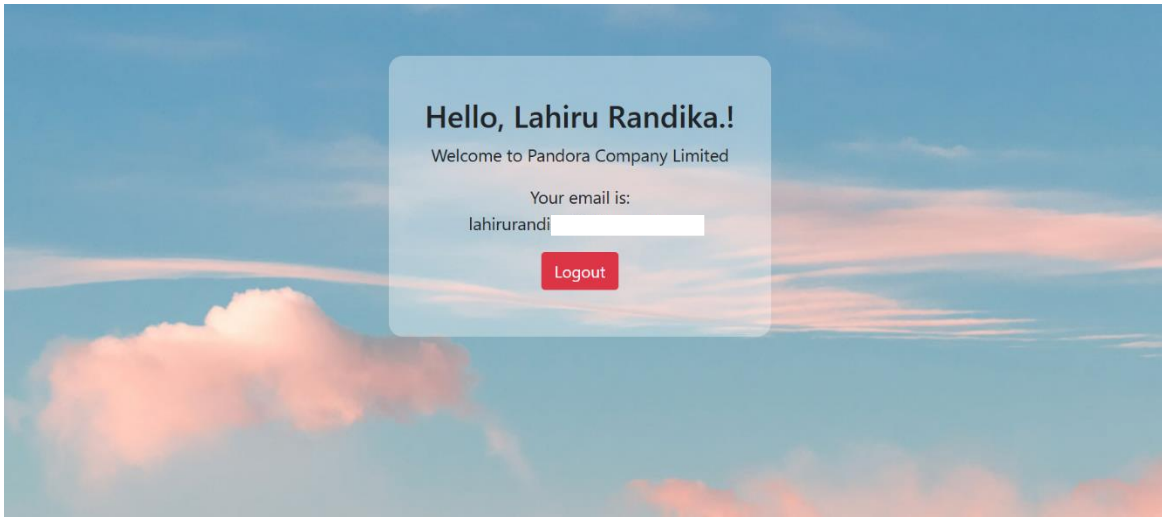
lahirurandik

By continuing, Google will share your name, email address, language preference and profile picture with PandoraLoginPage. See PandoraLoginPage's privacy policy and Terms of Service.

You can manage Sign in with Google in your Google Account.

Cancel Continue

http://localhost/website/dashboard.php



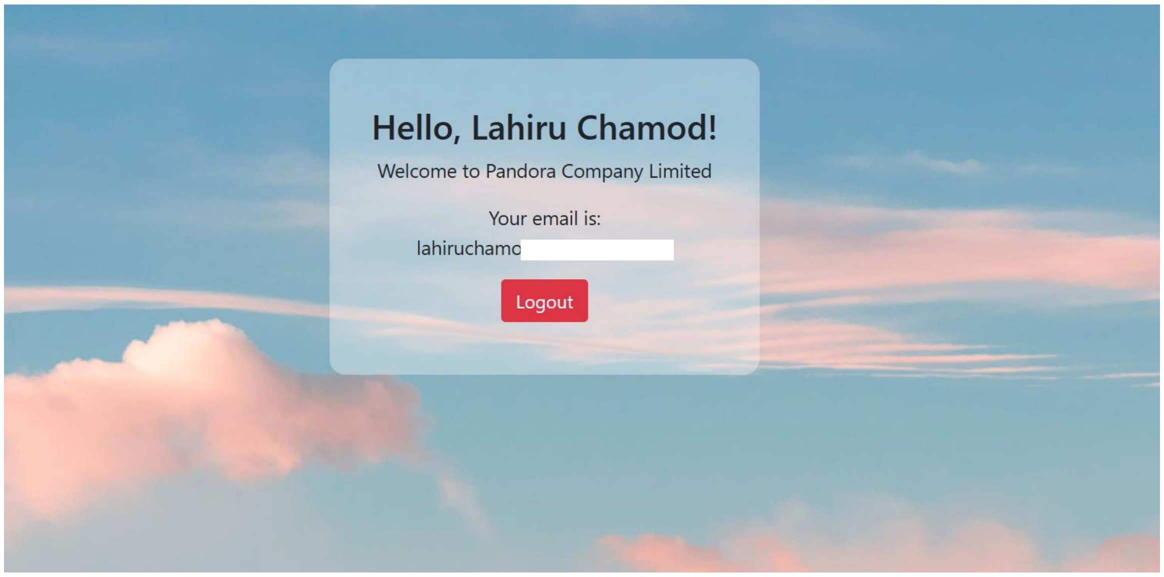
Hello, Lahiru Randika.!

Welcome to Pandora Company Limited

Your email is:

lahirurandi[redacted]

with another account



Hello, Lahiru Chamod!

Welcome to Pandora Company Limited

Your email is:

lahiruchamo[redacted]

3. Codes

Dashboard.php : < https://github.com/Lahiru-Randika/Pandora-Project/blob/main/dashboard.php/ >
Index.php : < https://github.com/Lahiru-Randika/Pandora-Project/blob/main/index.php/
Index.css : < https://github.com/Lahiru-Randika/Pandora-Project/blob/main/index.css/ >
Login.php : < https://github.com/Lahiru-Randika/Pandora-Project/blob/main/login.php/ >

Logout.php : < https://github.com/Lahiru-Randika/Pandora-Project/blob/main/logout.php/ >
Redirect.php : < https://github.com/Lahiru-Randika/Pandora-Project/blob/main/redirect.php/ >

Challenges Faced and Solutions

Challenge	**Solution**
Google Cloud Setup Issues	I watched YouTube tutorials and searched Google for setup guides. Ensured that the project and client credentials were correctly configured.
Constant Page Refreshes	Installed the *Live Server* browser extension to enable real-time page reloads while testing locally.
OAuth Client Error (401 Invalid Client)	Created a new project and OAuth client in Google Cloud to ensure the correct details were used, matching the redirect URI.
Lack of PHP Knowledge	watched YouTube videos and PHP documentation to learn how to work with Google OAuth2 and handling sessions in PHP.
Dealing with Redirect Issues	Ensured the redirect URI in Google Cloud matched exactly with the one used in the application.
Debugging Authentication Flow	Used var_dump() and print_r() for debugging session data and OAuth2 responses during development.
Problem With the .env file	Make a config.php file instead

Conclusion

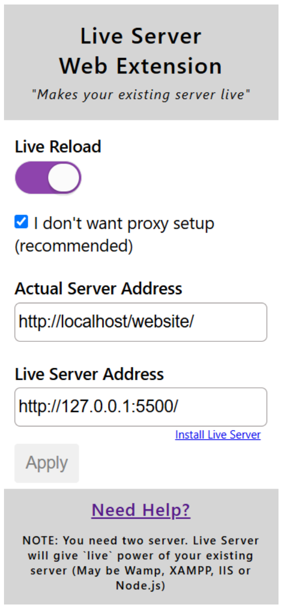
The implementation of OAuth2 authentication using Google for the Pandora Company Limited login system was a success, providing a secure and efficient way for users to log in with their Google credentials. The integration with the Google API Client Library ensures that user data is handled securely, without storing sensitive credentials directly.

Challenges such as setting up Google Cloud credentials, managing OAuth2 client errors, and troubleshooting page refresh issues were addressed through research, tutorials, and debugging tools. Despite limited PHP knowledge, online resources helped overcome obstacles and streamline the process.

In conclusion, the OAuth2 integration has improved both the security and user experience of the login system, demonstrating the effectiveness of third-party authentication in web applications.

References

- Youtube videos : < https://www.youtube.com/watch?v=OpvcXKn20Os />
  < https://www.youtube.com/watch?v=zZ6vybT1HQs&t=13777s />
  < https://www.youtube.com/watch?v=yi2b9U1kQyc />
  < https://www.youtube.com/watch?v=OKMgyF5ezFs/ >
- Login Page idea : < https://colorlib.com/wp/template/login-form-20/ >
- Google Developers : < https://developers.google.com/identity/protocols/oauth2/ >
- Medium.com : < https://medium.com/@tony.infisical/guide-to-using-oauth-2-0-to-access-google-apis-dead94d6866d/ >
- Composer : < https://github.com/MusabDev/php-google-login/ >
  < https://getcomposer.org/download/ >
- TechnoDecv : < https://www.thetechnodev.com/what-is-xampp-and-how-its-work-complete-guide/ >
- Images : < https://www.bing.com/images/search?view=detailV2&ccid=flV%2FHAhk&id=0278AB65C8D706270671B01ADE26B91346B1891B&thid=OIP.flV_HAhkgpxUwwDRW-5p9AHaHa&mediaurl=https%3A%2F%2Fpngimg.com%2Fuploads%2Fgoogle%2Fgoogle_PNG19635.png&cdnurl=https%3A%2F%2Fth.bing.com%2Fth%2Fid%2FR.7e557f1c0864829c54c300d15bee69f4%3Frik%3DG4mxRhO5Jt4asA%26pid%3DImgRaw%26r%3D0&exph=1125&expw=1125&q=google+pic+png&simid=607989433863461212&FORM=IRPRST&ck=BA32609A92FEC60EB37775C88F471677&selectedIndex=0&itb=1&cw=1152&ch=496&ajaxhist=0&ajaxserp=0 />
- Live server Web extension : < https://chromewebstore.google.com/detail/live-server-web-extension/fiegdmejfepffgpnejdinekhfieaogmj/ >



[✓] I don't want proxy setup (recommended)

Actual Server Address: http://localhost/website/

Live Server Address: http://127.0.0.1:5500/

NOTE: You need two server. Live Server will give `live` power of your existing server (May be Wamp, XAMPP, IIS or Node.js)