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.

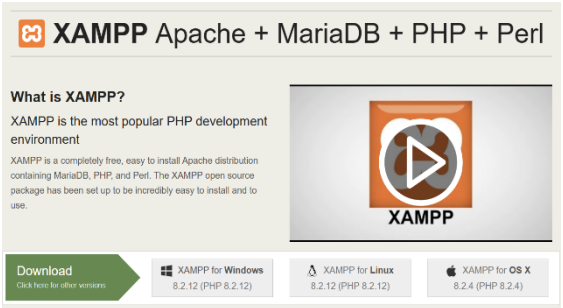
2. Tools and technologies used :-

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

3. Implementation Steps

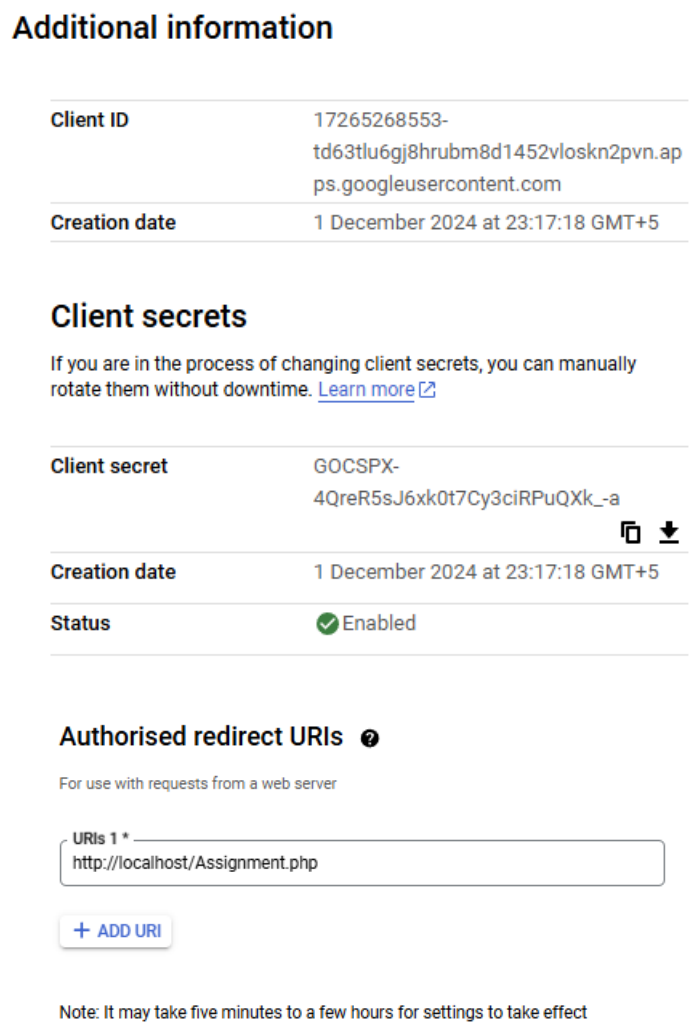
- **Install php**

  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.



- **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.



- 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.

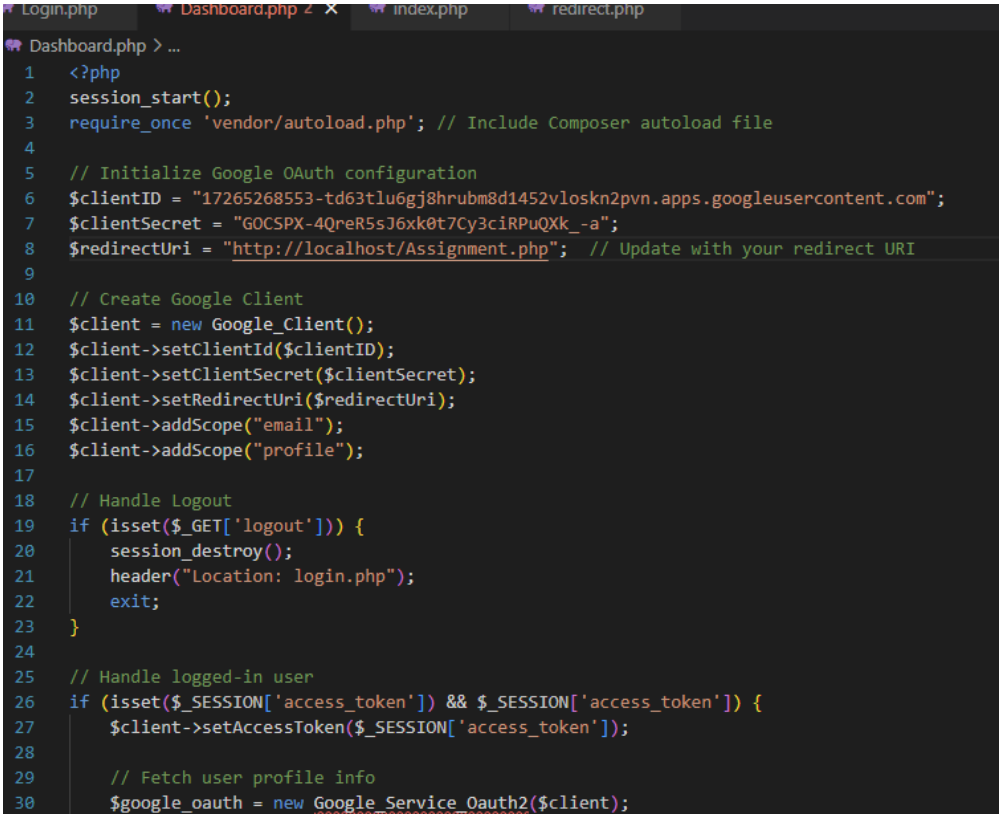
- Creating PHP Files
  1. **login.php -** This file serves as the starting point of the application, displaying the "Login with Google" button. Initialize the Google Client object. Configure the client with the **Client ID**, **Client Secret**, and **Redirect URI**. Generate an OAuth2 authorization URL.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

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

// Initialize Google OAuth configuration
$clientID = "17265268553-td63tlu6gj8hrubm8d1452vloskn2pvn.apps.googleusercontent.com";
$clientSecret = "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";
$redirectUri = "http://localhost/Assignment.php";  // Update with your redirect URI

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

// If user is already logged in, redirect to dashboard
if (isset($_SESSION['access_token']) && $_SESSION['access_token']) {
    header("Location: dashboard.php");
    exit;
}

// If not logged in, show the login page
?>
<!doctype html>
<html lang="en">
<head>
    <title>Login with Google</title>
    <meta charset="utf-8">
    <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.

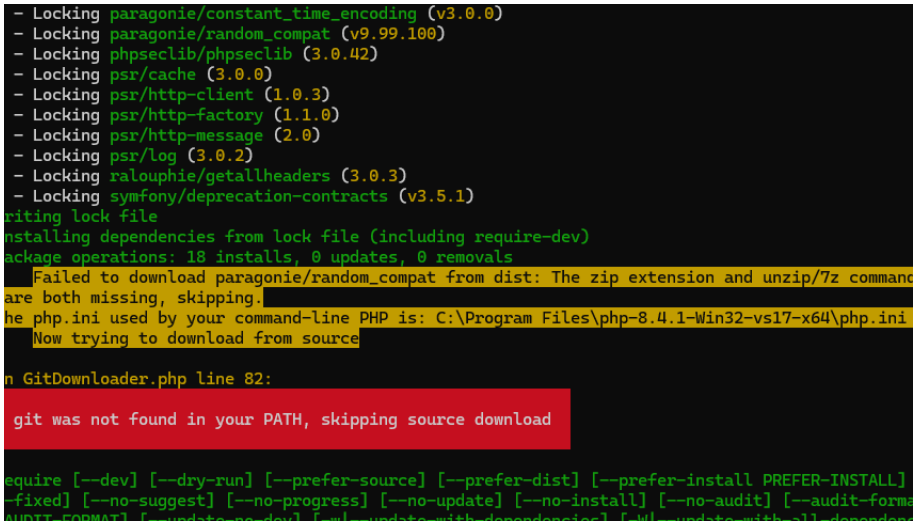
```php
<?php
session_start();
require_once 'vendor/autoload.php'; // Include Composer autoload file

// Initialize Google OAuth configuration
$clientID = "17265268553-td63tlu6gj8hrubm8d1452vloskn2pvn.apps.googleusercontent.com";
$clientSecret = "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";
$redirectUri = "http://localhost/Assignment.php"; // Must match with Google Developer Console

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

// 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
    header('Location: login.php');
    exit;
}
?>
```

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.



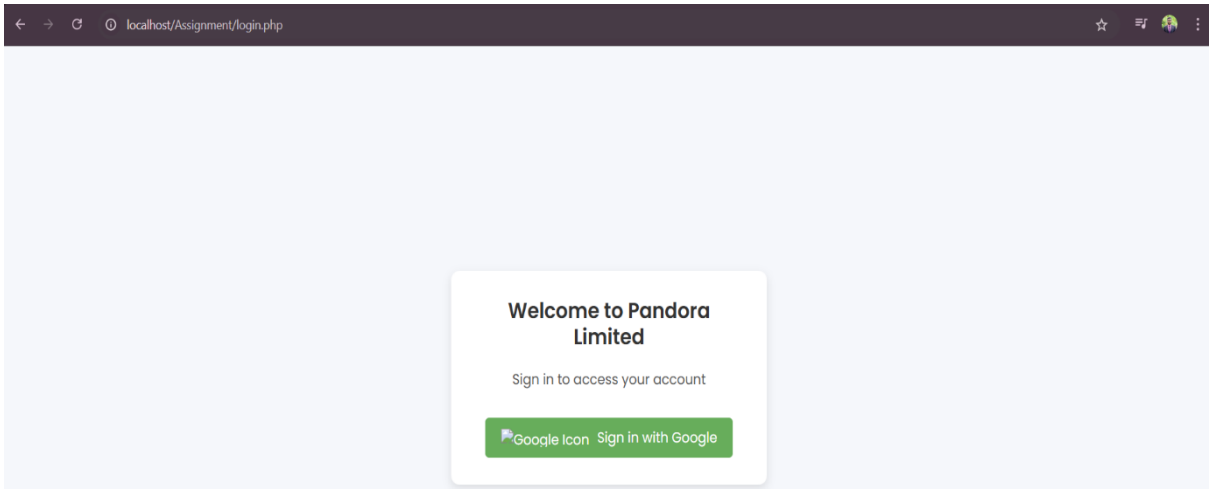
4. Challenges Faced

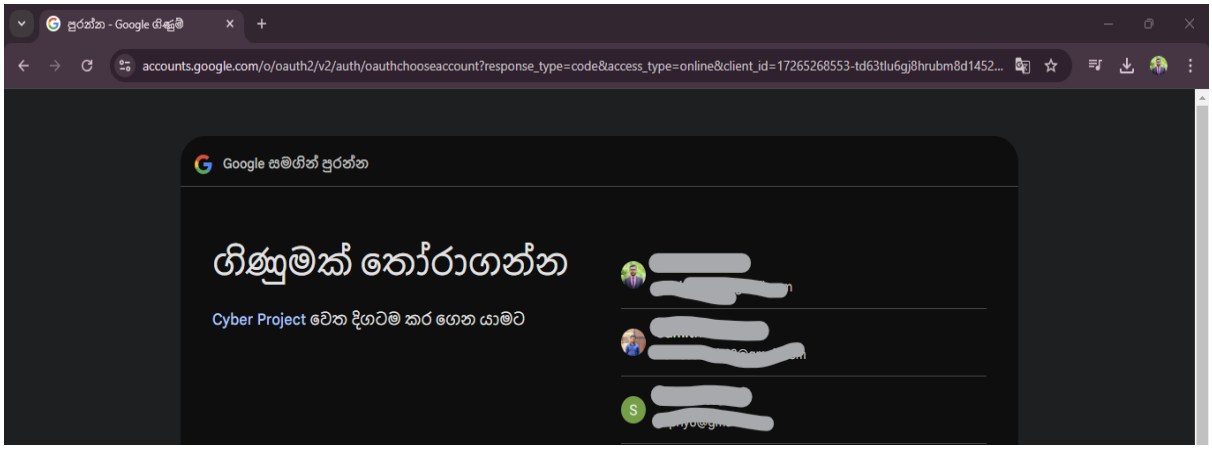
- Configuring the Redirect URI: Ensuring that the redirect URI specified in the Google Cloud Console matches the application endpoint exactly. Any mismatch (e.g., missing http://localhost or incorrect file path) results in authentication errors.
- Library or PHP Extension Issues: Encountered errors like 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.



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...