Script.

**Introduction**

For assignment 2 in this module, I decided to develop an application demonstrating an advanced technique using a web development framework and evaluate how well I feel that framework fitted to the needs of the assignment.

The web framework that I have decided to use is Phalcon. Phalcon is an open source, full stack framework for PHP 5 written as a C-extension, with its functionality exposed as PHP classes.

The application that I decided to develop demonstrates an Authentication and Authorization standards when accessing a web service. This standard is known as the OAuth2 protocol, which replaced the original OAuth protocol in 2006.

OAuth2 is a protocol that lets external applications request authorization to private details of a user without getting their password by requesting access in the form of an access token. This is preferred over Basic Authentication because tokens have what is known as ‘scope’. Scope means that tokens are limited to specific pieces of data, which can be revoked by users at any time.

**Going through the Application**

In this application, I have developed 3 small websites along with a simple web service to demonstrate how OAuth2 is used. The websites include a simple client application, the authentication server and finally a user account backend to control access to authorized applications. I should also say at this time that all of the visual websites have used twitter bootstrap for the styling.

So to demonstrate the application flow that is used in order to request access to private data we start on the client applications website which as you can see just contains a simple button. When we press this button the user is then redirected to the authentication server at ‘/oauth/authorize’ with a query string with the initial details regarding this request. The details that have been send are the client id, which is the unique identifier for the client requesting access, scope, which is the data that the client is requesting, a state variable, which is an un-guessable random string which is used to protect against cross-site request forgery attacks, and finally a redirect URI, which is the website that we should redirect to after authentication is completed i.e the user has accepted or declined access.

I should say at this point that in a live system a client will have to sign up as a developer in order to gain a client id and secret in order to make requests and during this process they will have to provide a default redirect URI which will be used if one is not supplied or the one supplied is not valid. The supplied redirect URI in the initial request has to have the same hostname as the default one or the default one will be used. This is to prevent callbacks being made to another website.

If all of the provided information is valid this log in form will be displayed, otherwise the server will send a 403 Forbidden status.

At this point the user will log in with his normal credentials, but this is done not done on the client server meaning that the client never has access to the users log in credentials.

After the user successfully logs in, the authentication will display all of the permissions that the client is asking for and we can accept a certain set of permissions or decline the request outright.

If we decline the request, an error message is passed back to the client and this has to be deal with accordingly, in this case it just redirects back to the initial page.

If the user accepts the request however, a temporary code is passed back to the client with the original state string. The code can be then exchanged for an access token with the accepted scopes attached to it. The access token is then used in any web service requests to get information on the user that authorized us access, in this case it calls the ‘/user’ api which returns the users profile information.

I have also developed a simple backend for a user which allows users to alter permissions given to client applications. This initial page uses the same login information to log in to the backend.

This then takes us to the users backend and at the bottom of the page we can see the applications that this user has given access to. We can then alter the permissions this client has or revoke permissions entirely, meaning they will have to re-request for permission if they try to access this users profile using the access token gained in a previous authentication request.

**Talk about the code behind the application.**

**Conclusion.**