GAppv3 using Google Drive API and OAuth 2.0 Functionality for Authentication/Authorization.

FINAL PROJECT

SOFTWARE SECURITY

KUSHAN WIJAYASURIYA – MS19814148

HASNA M R N - MS19801100

AMIRTHALINGAM SHIVADHARCSHAN – MS19813776

I. INTRODUCTION

Applications interact with the resource owner and the resource server regularly, third party applications such as login through Facebook to access Pinterest resources. OAuth 2.0 is the successor of OAuth 1.0 which was less secure in terms of authentication and authorization.

OAuth 1.0 directly requests for the resource owner's credentials for the app (Facebook) where the client app will use those credentials on behalf of the resource owner, which in return can be used to access not only the requested resources but everything else as well.

On the contrary OAuth 2.0 rectifies this issue by directly requesting the resource owner to authenticate and then authorize certain resources to the client app, this is done by using an access token which will be sent to the client app and then the client app can use that access token to access the required resources only.

II. DEFINITIONS

Resource Owner – User

Client – GAppv3

Authorization Server – Google Servers

Resource Server - Google Drive

III. GAppv3

This app is a resource aggregator where it aggregates certain type of files from all the cloud storage platforms like Google Drive, OneDrive and places it in the server of the *GAppv3*, so it is easy for the user to access specific files in one place.

GAppv3 was built on Visual Studio Community Edition 2019 on top of Microsoft model view controller (MVC) framework. The app utilizes the latest Google drive API version 3 and will mainly highlight the use of OAuth 2.0 in the form of a simple file manager application.

The main features of the application will be to display an authenticated users Google drive files and give the user the ability to list and delete their files one at a time.

We are using Google REST API to allow authorization and communication of third-party applications with Google provided services, which in this case is Google Drive.

The authentication of the application with Google services is made via a (.jason) file (client_secret) which includes; Client ID (GAppv3 id), Client secret, Redirect URL, Authentication URL, project id, authentication URI, token URI, authentication provider x509 certificate URL.

Work Flow of the application and OAuth 2.0 is explained in the diagram below.

GDApp

[client_secret.json]

1. GAppv3 requests to access the files on resource owner's Google Drive

Resource Owner

2. Resource owner allows GAppv3 and GAppv3 redirects the request to Resource Owner's Google Authorization server

Resource Owner

GDApp

[client_secret.json]

3. Resource owner then authenticates himself on Google Drive/ Google Authorization Server

GDApp

[client_secret.json] +

[DriveServiceCredentials.json]

4. Google Drive/ Google Authorization Server sends an access key to the GAppv3

Authentication Server- Google

GDApp

[client_secret.json] +

[DriveServiceCredentials.json]

5. Using the access key, GAppv3 will now request fo the resources from Google Drive

GDApp

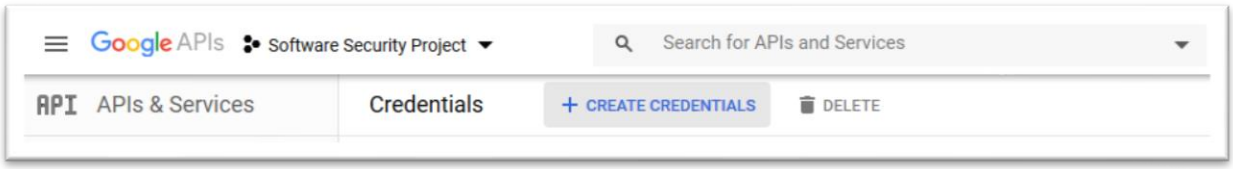
[client_secret.json] +

[DriveServiceCredentials.json]

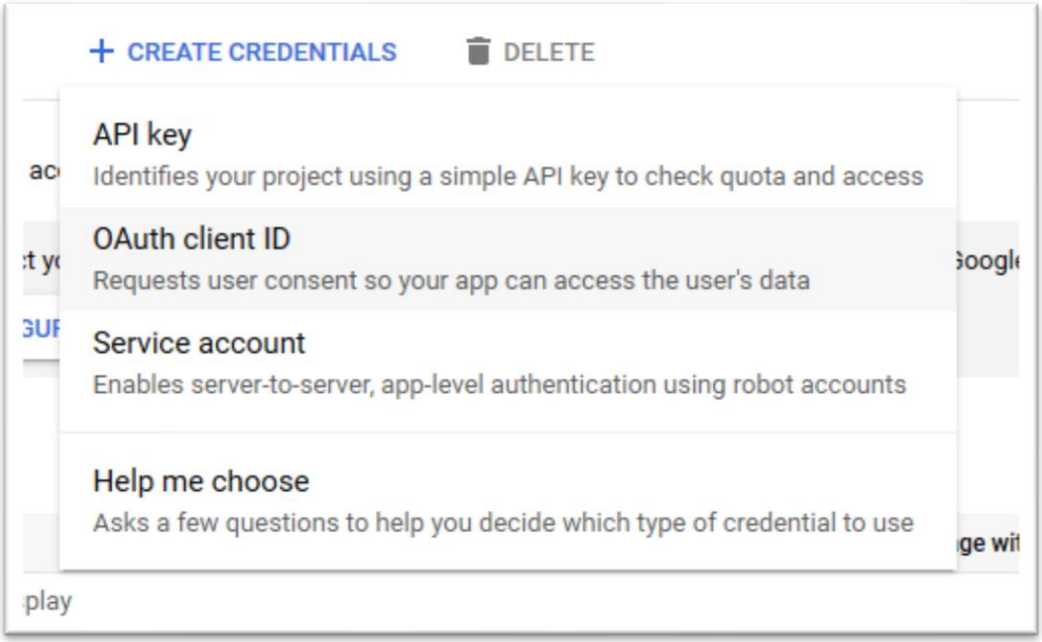
6. Google Drive allows GAppv3 to access the

IV. CREATING THE [authorization_token.json] FILE

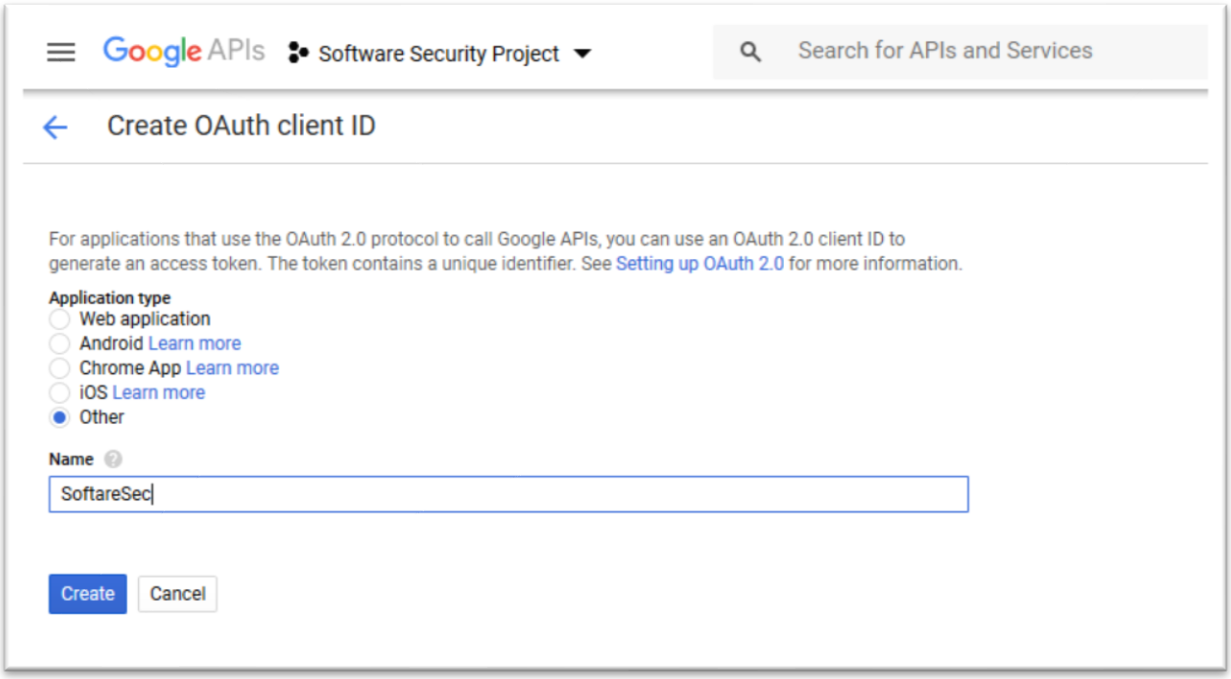
1. Under Google API Developer console click create credentials.



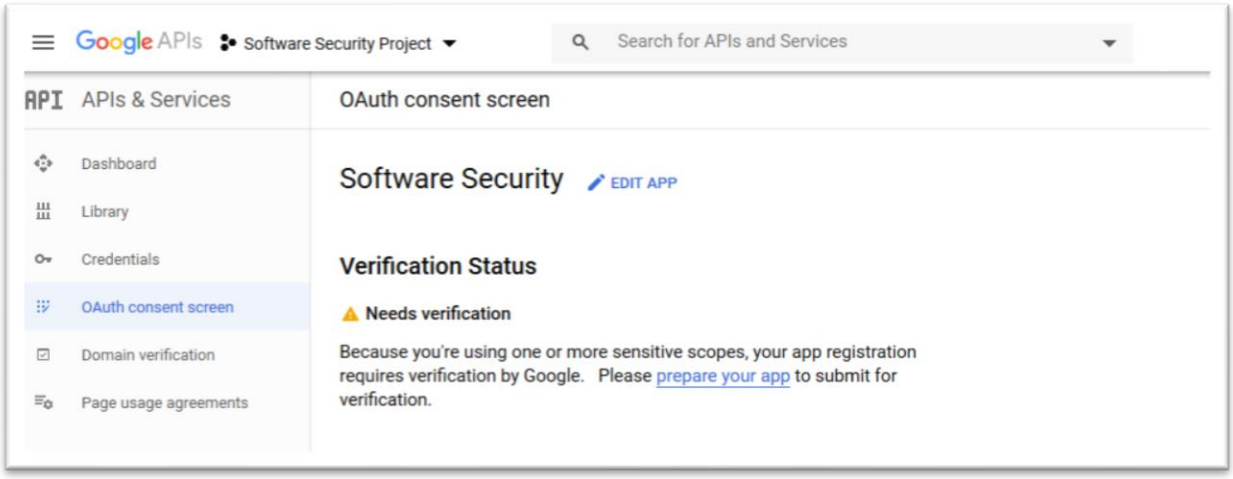
2. Then click on the OAuth Client ID.

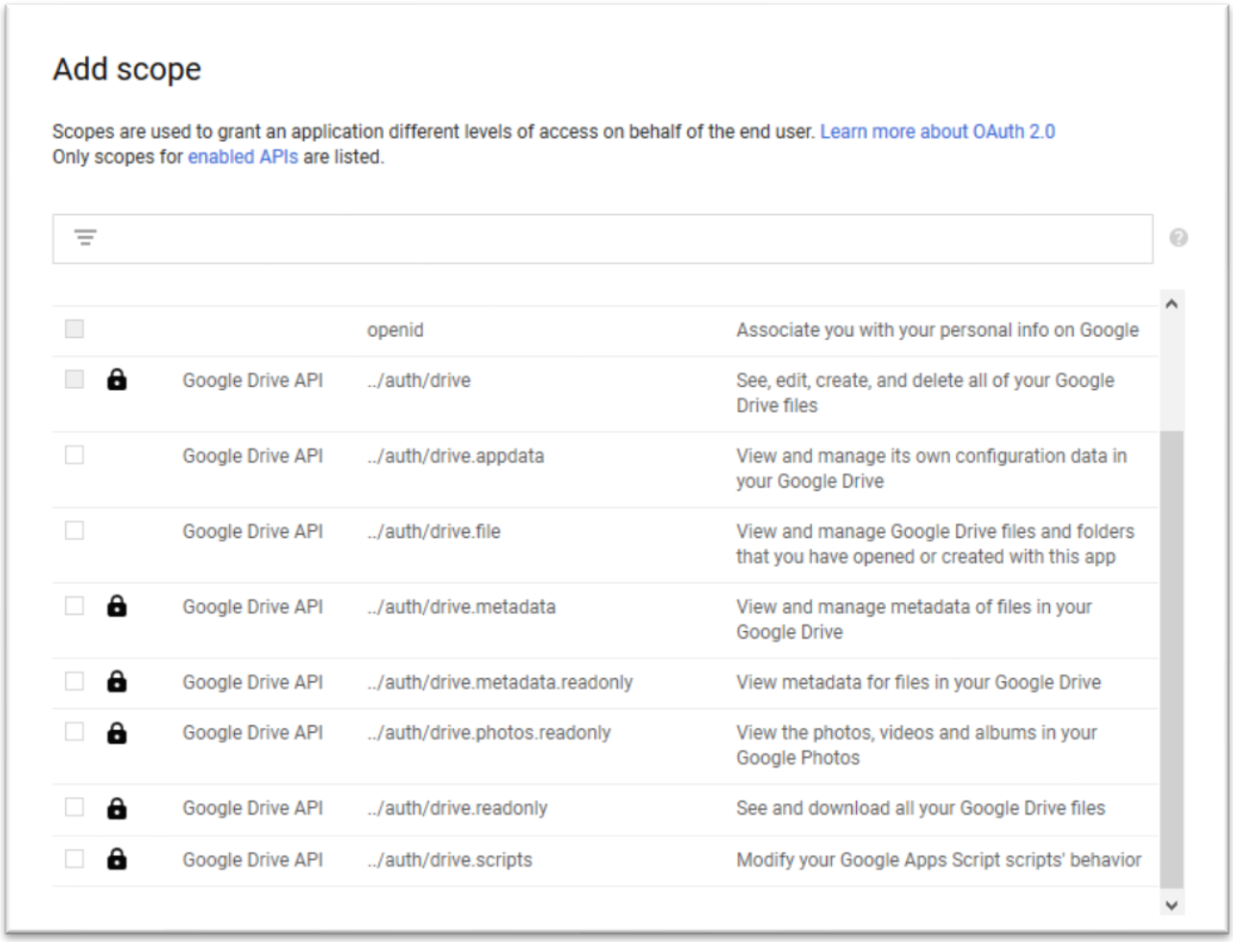


3. Choose and name and select the type “Other”.

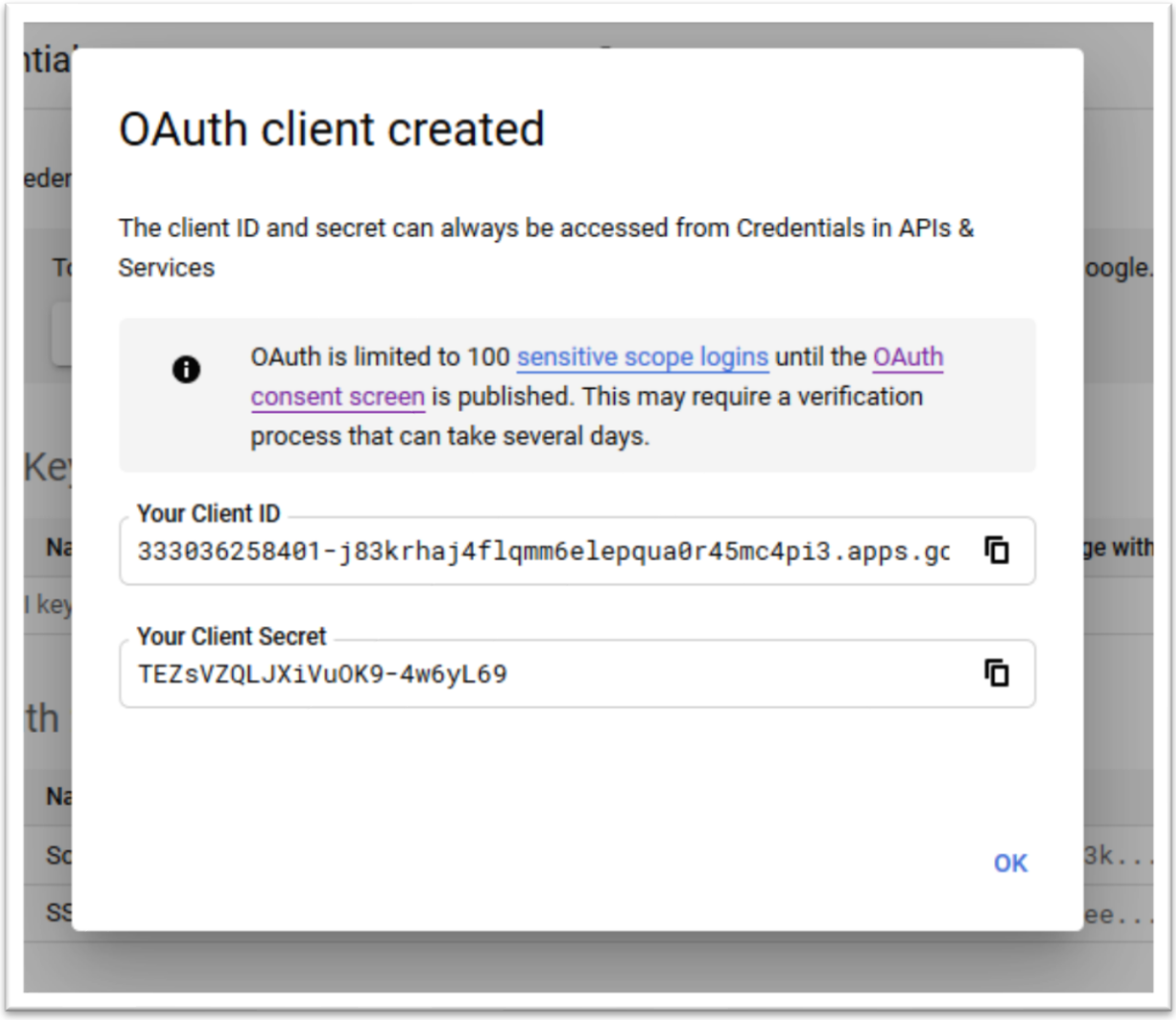


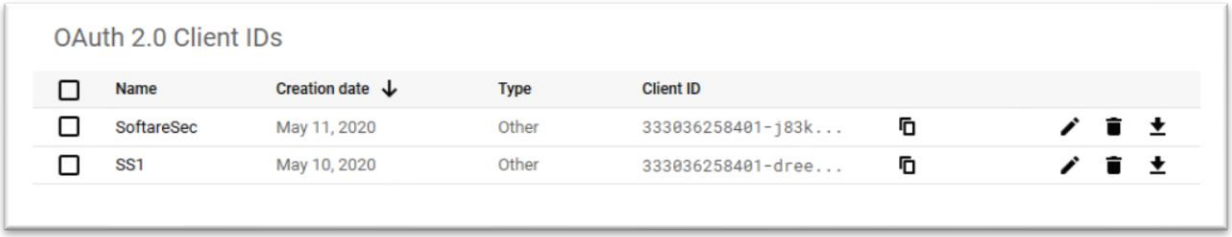
4. On the consent screen add the scope and save.





5. Get the client secret and the token





V. APPLICATION SOURCE CODE

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace GAppv3.Models
{
    public class GDAppFiles
    {
        public string ID { get; set; }
        public string FileName { get; set; }
        public long? FileSize { get; set; }
        public long? FileVersion { get; set; }
        public DateTime? CreatedTime { get; set; }
        public DateTime? ModifiedTime { get; set; }
    }
}

using Google.Apis.Auth.OAuth2;

using Google.Apis.Download;
using Google.Apis.Drive.v3;
using Google.Apis.Services;
using Google.Apis.Util.Store;
using System;
using System.Collections.Generic;
using System.IO;
using System.Threading;
using System.Web;

namespace GAppv3.Models
{
    public class GDAppRepo
    {

        public static string[] Scopes = { DriveService.Scope.Drive };


        public static DriveService GetService()
        {

            UserCredential authority;
            using (var stream = new FileStream(@"D:\authorization_token.json",
FileMode.Open, FileAccess.Read))
            {
                String FolderPath = @"D:\";
                String FilePath = Path.Combine(FolderPath, "GDriveServiceCreds.json");

                authority = GoogleWebAuthorizationBroker.AuthorizeAsync(
                    GoogleClientSecrets.Load(stream).Secrets,
                    Scopes,
                    "user",
                    CancellationToken.None,
                    new FileDataStore(FilePath, true)).Result;
            }


            DriveService service = new DriveService(new
BaseClientService.Initializer()
```

```
            {
                HttpClientInitializer = authority,
                ApplicationName = "GAppv3",
            });
            return service;
        }


        public static List<GDAppFiles> GetDriveFiles()
        {
            DriveService service = GetService();

            FilesResource.ListRequest FileListRequest = service.Files.List();



            FileListRequest.Fields = "nextPageToken, files(id, name, size, version,
createdTime, modifiedTime)";


            IList<Google.Apis.Drive.v3.Data.File> files =
FileListRequest.Execute().Files;
            List<GDAppFiles> FileList = new List<GDAppFiles>();

            if (files != null && files.Count > 0)
            {
                foreach (var file in files)
                {
                    GDAppFiles File = new GDAppFiles
                    {
                        ID = file.Id,
                        FileName = file.Name,
                        FileSize = file.Size,
                        FileVersion = file.Version,
                        CreatedTime = file.CreatedTime,
                        ModifiedTime = file.ModifiedTime
                    };
                    FileList.Add(File);
                }
            }
            return FileList;
        }
        public static void DeleteFile(GDAppFiles files)
        {
            DriveService service = GetService();
            try
            {
                // Initial validation.
                if (service == null)
                    throw new ArgumentNullException("service");

                if (files == null)
                    throw new ArgumentNullException(files.ID);

                // Make the request.
                service.Files.Delete(files.ID).Execute();
            }
            catch (Exception ex)
            {
                throw new Exception("Request Files.Delete failed.", ex);
            }
        }
```

```
    }
}

using GAppv3.Models;
using System.IO;
using System.Web;
using System.Web.Mvc;

namespace GAppv3.Controllers
{
    public class HomeController : Controller
    {
        [HttpGet]
        public ActionResult GetGoogleDriveFiles()
        {
            return View(GDAppRepo.GetDriveFiles());
        }

        [HttpPost]
        public ActionResult DeleteFile(GDAppFiles file)
        {
            GDAppRepo.DeleteFile(file);
            return RedirectToAction("GetGoogleDriveFiles");
        }
    }
}

@model IEnumerable<GAppv3.Models.GDAppFiles>
@{
    ViewBag.Title = "Software Security - OAuth Simple File Manager Application";
}

    <h2>Software Security - OAuth Simple File Manager Application</h2>
<script src="https://code.jquery.com/jquery-3.4.0.min.js"></script>

<style type="text/css">
    #header {
        width: 100%;
        background-color: #accae3;
        text-align: center;
    }

    #layouttable {
        border: 0px;
        width: 100%;
        font-family: 'Cocogoose Pro';
        background-color: #d3edf0;
    }

        #layouttable td.col1 {
            width: 20%;
            vertical-align: top;
        }

        #layouttable td.col2 {
            width: 60%;
            vertical-align: top;
            background-color: #c3d9db;
        }

        #layouttable td.col3 {
            width: 20%;
```

```
            vertical-align: top;
        }
</style>

<center>


    <table class="table" border="1">
        <tr id="header">
            <th>
                @Html.DisplayNameFor(model => model.FileName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.FileSize)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.FileVersion)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.CreatedTime)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ModifiedTime)
            </th>
            <th>
                Delete File
            </th>
        </tr>

        @if (Model.Count() > 0)
        {
            foreach (var item in Model)
            {
                <tr id="layouttable">
                    <td>
                        @Html.DisplayFor(modelItem => item.FileName)
                    </td>
                    <td>
                        @{
                            long? KiloByte = @item.FileSize / 1024;
                            string NewSize = KiloByte + " KB";
                        }
                        @NewSize
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.FileVersion)
                    </td>
                    <td>
                        @string.Format("{0: MM/dd/yyyy}",
Convert.ToDateTime(Html.DisplayFor(modelItem => item.CreatedTime).ToString()))
                    </td>
                     <td>
                        @string.Format("{0: MM/dd/yyyy}",
Convert.ToDateTime(Html.DisplayFor(modelItem => item.ModifiedTime).ToString()))
                    </td>
                    <td>
                        @using (Html.BeginForm("DeleteFile", "Home", FormMethod.Post,
new { enctype = "multipart/form-data" }))
                        {
                            <input type="hidden" name=Id value="@item.ID">
                            <input type="submit" class="DeleteFile" value="Delete"
style="align-content:center" />
```

```
                        }

                    </td>

                </tr>
            }
        }
        else
        {
            <td colspan="6">The List is Empty</td>
        }

    </table>

</center>

<script>
    $(document).on('click', '.DownloadFile', function () {
        debugger;
        var fileId = $(this).attr("data-key");
        window.location.href = '/Home/DownloadFile/' + fileId;
    });
</script>

@model IEnumerable<GAppv3.Models.GDAppFiles>
@{
    ViewBag.Title = "Software Security - OAuth Simple File Manager Application";
}

    <h2>Software Security - OAuth Simple File Manager Application</h2>
<script src="https://code.jquery.com/jquery-3.4.0.min.js"></script>

<style type="text/css">
    #header {
        width: 100%;
        background-color: #accae3;
        text-align: center;
    }

    #layouttable {
        border: 0px;
        width: 100%;
        font-family: 'Cocogoose Pro';
        background-color: #d3edf0;
    }

        #layouttable td.col1 {
            width: 20%;
            vertical-align: top;
        }

        #layouttable td.col2 {
            width: 60%;
            vertical-align: top;
            background-color: #c3d9db;
        }

        #layouttable td.col3 {
            width: 20%;
            vertical-align: top;
        }
</style>
```

```
<center>


    <table class="table" border="1">
        <tr id="header">
            <th>
                @Html.DisplayNameFor(model => model.FileName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.FileSize)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.FileVersion)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.CreatedTime)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.ModifiedTime)
            </th>
            <th>
                Delete File
            </th>
        </tr>

        @if (Model.Count() > 0)
        {
            foreach (var item in Model)
            {
                <tr id="layouttable">
                    <td>
                        @Html.DisplayFor(modelItem => item.FileName)
                    </td>
                    <td>
                        @{
                            long? KiloByte = @item.FileSize / 1024;
                            string NewSize = KiloByte + " KB";
                        }
                        @NewSize
                    </td>
                    <td>
                        @Html.DisplayFor(modelItem => item.FileVersion)
                    </td>
                    <td>
                        @string.Format("{0: MM/dd/yyyy}",
Convert.ToDateTime(Html.DisplayFor(modelItem => item.CreatedTime).ToString()))
                    </td>
                     <td>
                        @string.Format("{0: MM/dd/yyyy}",
Convert.ToDateTime(Html.DisplayFor(modelItem => item.ModifiedTime).ToString()))
                    </td>
                    <td>
                        @using (Html.BeginForm("DeleteFile", "Home", FormMethod.Post,
new { enctype = "multipart/form-data" }))
                        {
                            <input type="hidden" name=Id value="@item.ID">
                            <input type="submit" class="DeleteFile" value="Delete"
style="align-content:center" />
                        }

                    </td>
```

```
                </tr>
            }
        }
        else
        {
            <td colspan="6">The List is Empty</td>
        }

    </table>

</center>

<script>
    $(document).on('click', '.DownloadFile', function () {
        debugger;
        var fileId = $(this).attr("data-key");
        window.location.href = '/Home/DownloadFile/' + fileId;
    });
</script>
```

VI. References

[1] C. Bihis, Mastering OAuth 2.0, Birmingham, USA, Packt Publishing Ltd. 2015.

[2] OAuth 2.0 for Client-side Web Applications. Google Identity Platform. N.d. [Online]. Available: https://developers.google.com/identity/protocols/oauth2/javascript-implicit-flow#prerequisites

[3] The OAuth 2.0 Authorization Framework. Internet Engineering Task Force. October 2012. [Online]. Available: https://tools.ietf.org/html/rfc6749

[4] Google Drive API. Uploading, Viewing, Downloading, & Deleting Files. September 2017. [Online]. Available: https://www.youtube.com/watch?v=aTv5t7oH6X8