SOFTWARE SECURITY ASSIGNMENT – OAUTH 2.0

Group Members

Student Number	Student Name
MS21909528	Wimalasekera D.H
MS21909696	H. D. C Madushan
MS21909382	W.A.N.P Perera

Table of Contents

Table of Figures

OAuth 2.0 Framework

Introduction

Today, with a variety of internet applications and service, internet users are required to access their data via other applications. Therefore OAuth concept has gained popularity among web resource sharing.

OAuth is a protocol to enable secure authentication in web, mobile and desktop applications. In simple terms, OAuth allows a resource owner to grant limited access to his resource for a different web application. The process hides user's login credential and give access to the application based on the specified scope. OAuth 2.0 is the "industry-standard protocol for authorization" [1].

Basic OAuth2.0 terms are specified as;

- Resource Owner – An entity who is the owner of a protected resource and can delegate access to that resource. The resource owner interacts with a third party client application through which he needs to access his resources on a different server.
- Client Application – Application which make the request to access the protected resource on behalf of the resource owner. Should be registered with an authorization server to get credentials for the application
- Resource Server - Server which implements the protected resource. Resource server is not accessed by the client application until the authorization process is successful.
- Authorization Server – Server managing the access tokens and issue them to client applications when authorization received
- Scope – the level of access granted to the client application on the protected resource. Since granting full access is not acceptable at most levels, the scope defines the level of access given to the web application by the resource owner.

Roles in an OAuth Framework

Figure 1 gives a brief insight in to the roles involved in an OAuth 2.0 framework and their functioning.

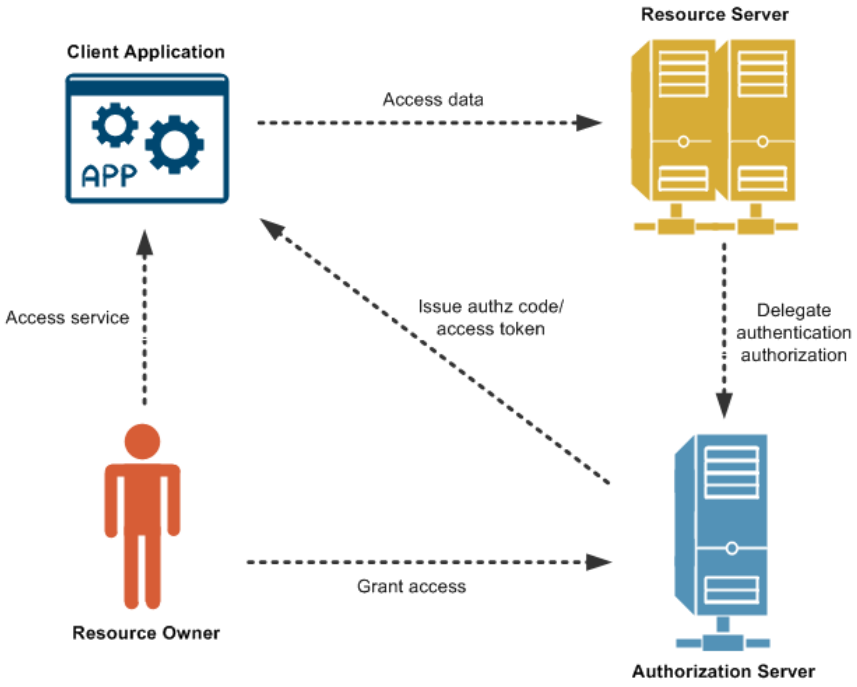


Figure 1:OAuth Framework Roles [2]

OAuth 2.0 Authorization Flow

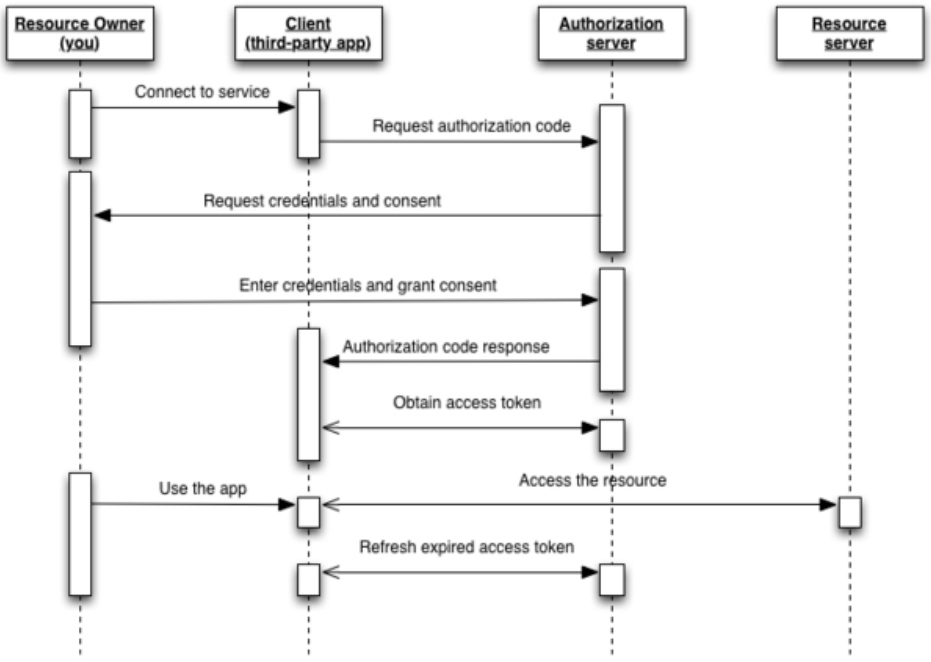


Figure 2:OAuth2.0 Authorization Flow [3]

OAuth 2.0 authorization flow is briefed in the figure 2. The user credentials to login to the resource server is hidden from the client application and once authorized, a token is given to access the resource.

Prototype Web Application

The application will depict the client application requiring to upload files to the end user's Google Drive. The client application will ask the client to use google login and upload files to the Google Drive.

The message flow of the web application process is described in the diagram.

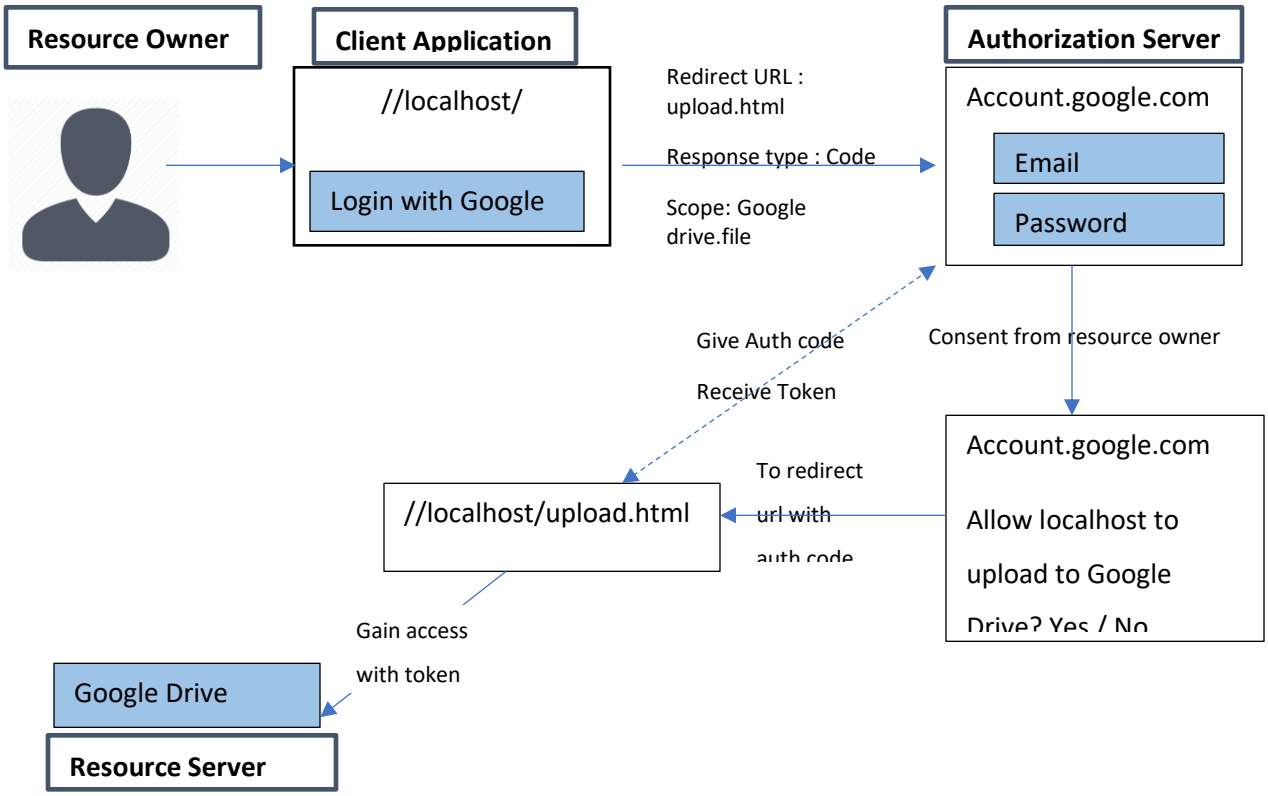


Figure 3: Message Flow of the Web Application

Project Development

A simple client application is developed with an index file and an upload page in html format with javascript. The index.html file is the home page where the web application request resource owner to log in to the google account.

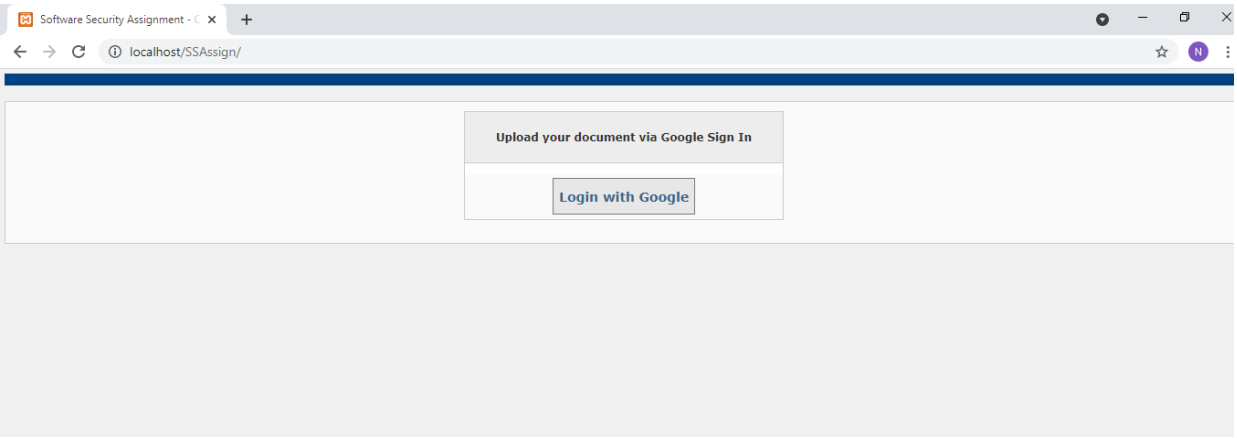


Figure 4: Index page

The upload.html page allows the user to upload files to the google drive after authentication. The upload.html is the redirect url where the authentication server redirects the user once successfully authenticated.

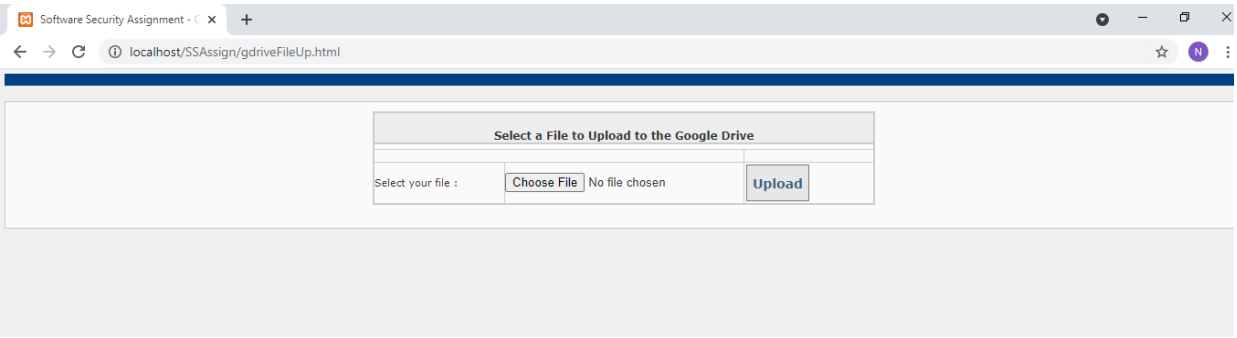


Figure 5: Upload page

Create a Project in Google Developer Console

Create a new project with a project name in google developer console

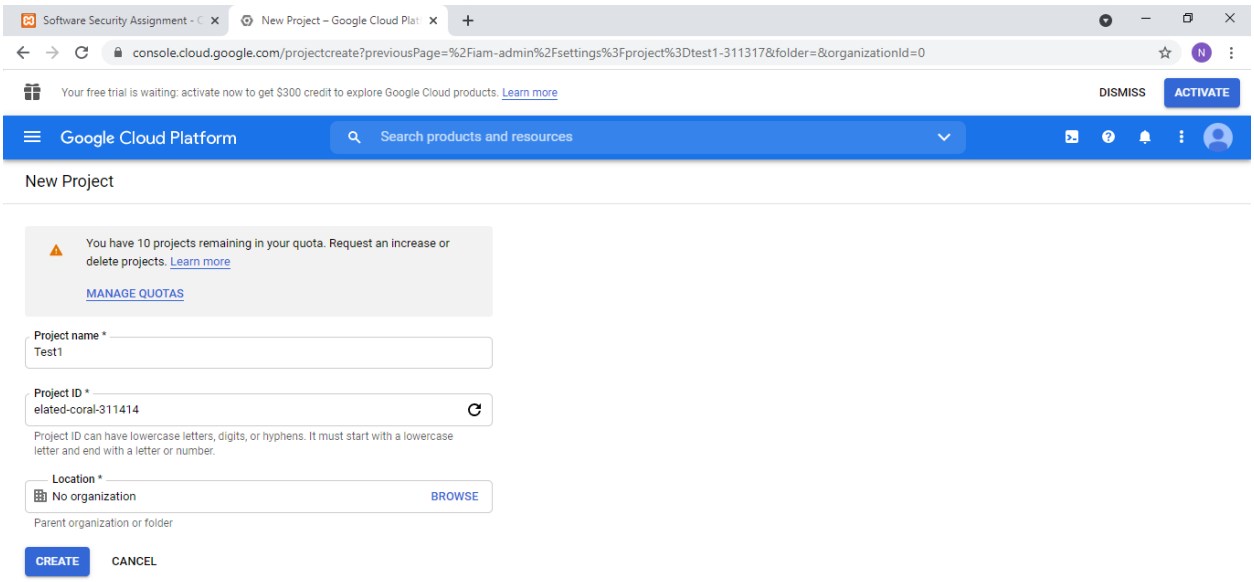


Figure 6: Create Project in Google Developer

Since the web application is required to upload files to the google drive, google drive APIs should also be enabled from google developer console.

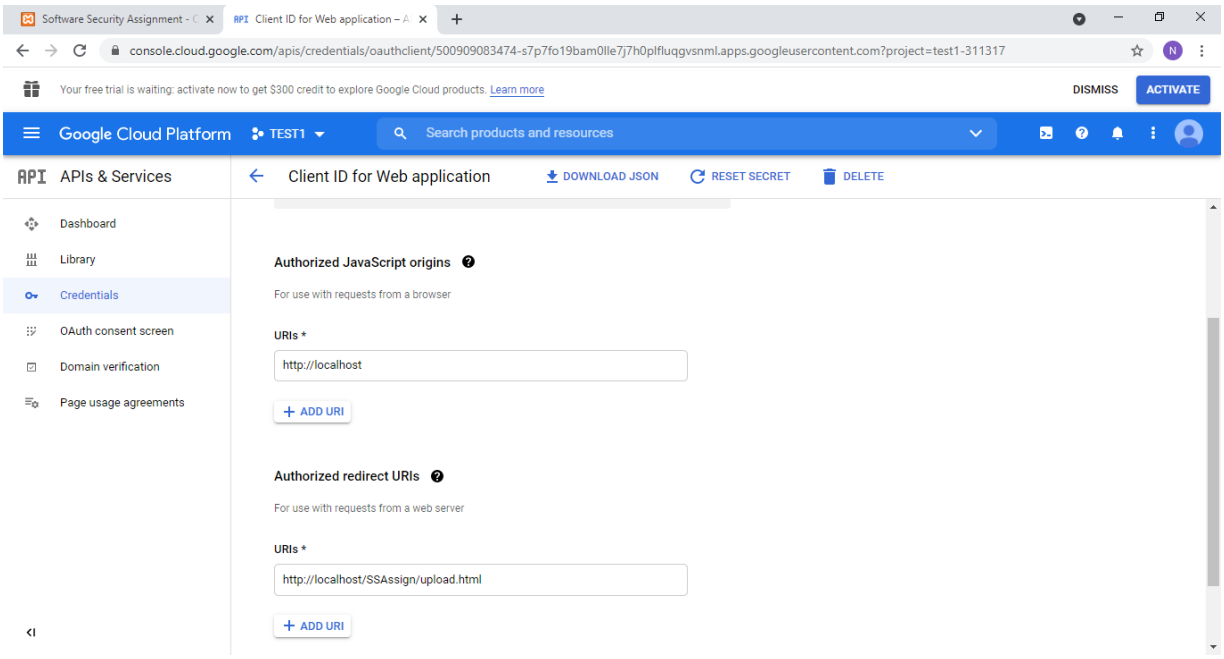


Figure 7: Create Credentials

Create credentials by selecting “OAuth Client ID”, which allows the request of user consent for the application to access his data. Under credential creation the URI of the application and the redirect URI for the client to be redirected should be specified. Once the credentials are created, a client ID and a

client secret are given to be included in the project. The client ID and the client Secret are unique to the application on the respective authorization server. In order for the client application to authenticate itself, it should send the client ID and the secret requesting from the authentication server.

The message flow of the process

Click on the "Login with Google" button in the home page.

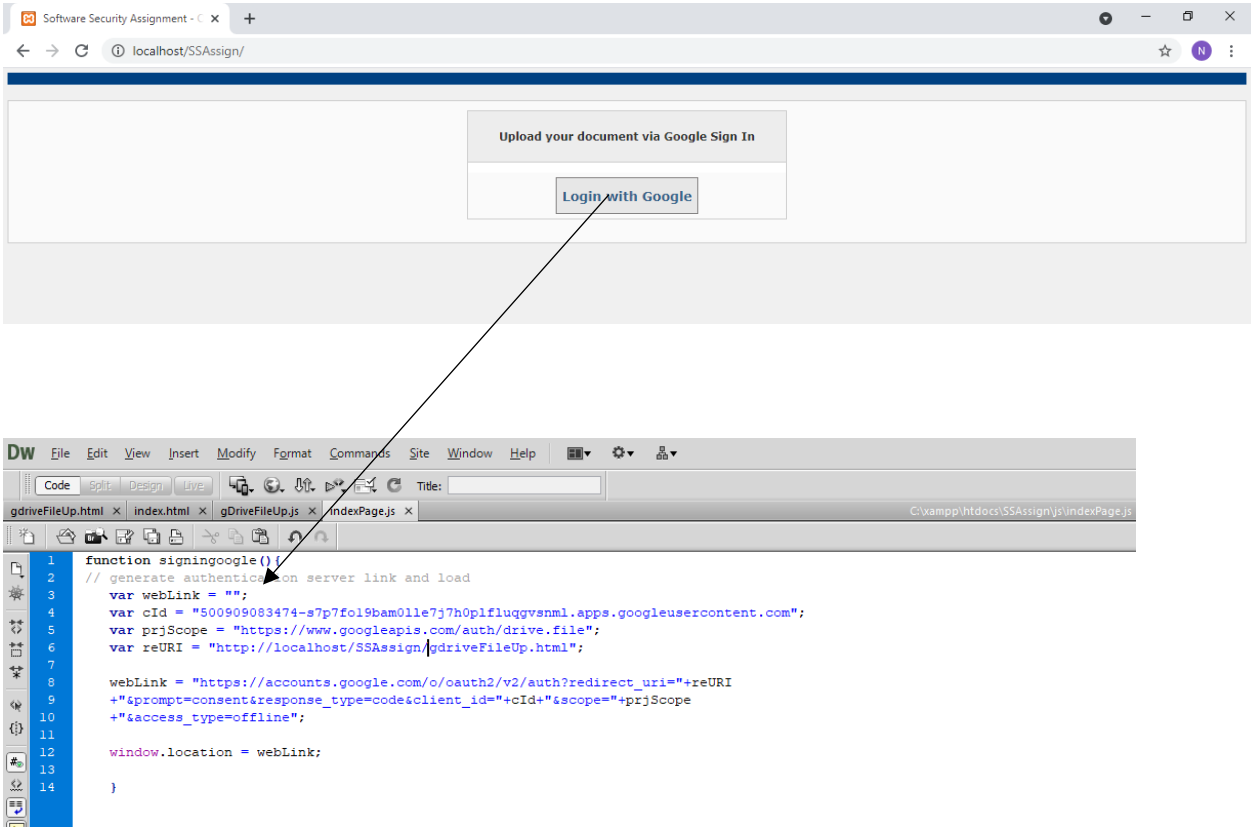


Figure 8 : Code snippet at Login with Google button

The button click will activate the following javascript function. It creates the URL to the authorization server with client ID, redirect URI and the project scope.

With the button click, the resource owner is redirected to the google user consent page hosted under accounts.google.com

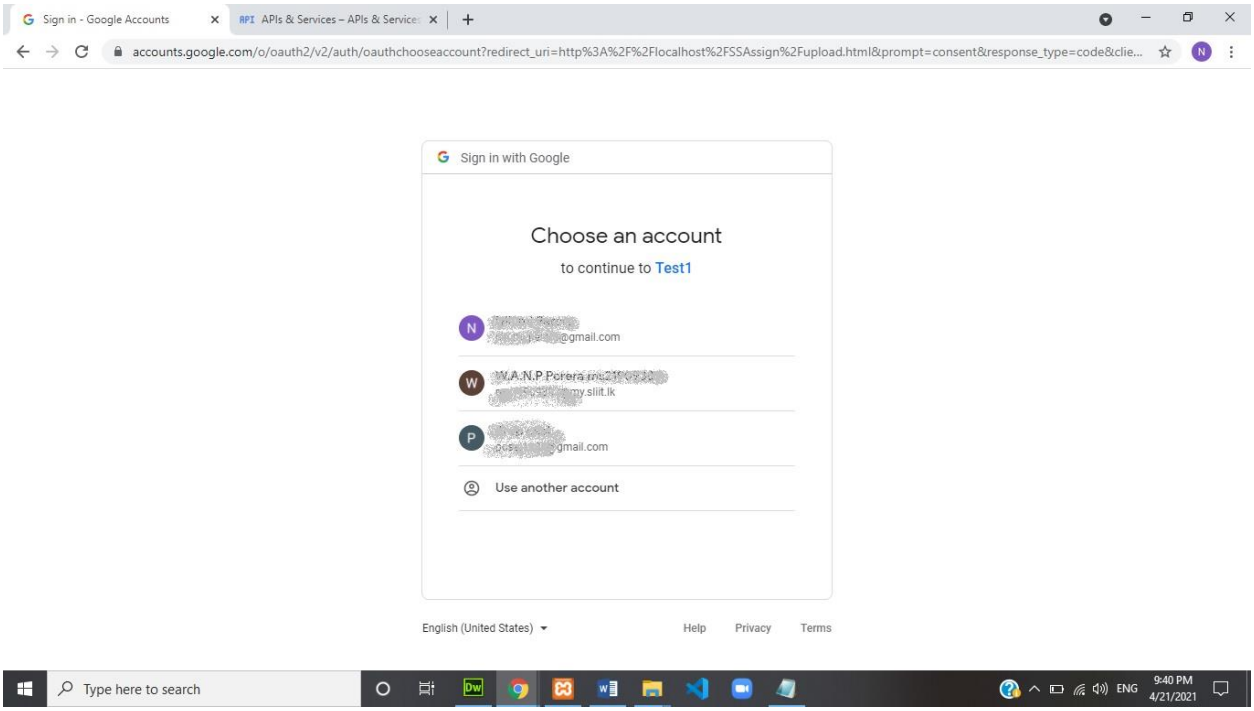


Figure 9: Login with Google account details

Here the resource owner can log in to the google account of which he need to upload files.

Once logged in, the authorization server ask again for confirmation on trusting the web application.

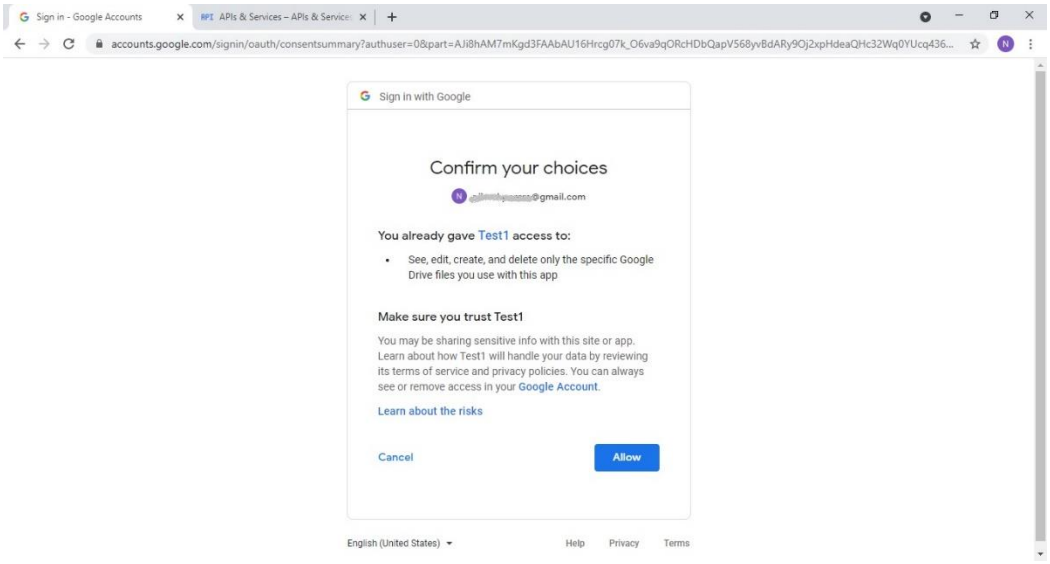


Figure 10 : Confirm Resource owner's choice

When the resource owner confirm his choice, the application request for a token from the Authorization server. For this request, the application must send the client ID, client secret, scope and other

parameters. Auth sever redirect back to the web application with a token which is stored internally by the javascript code.

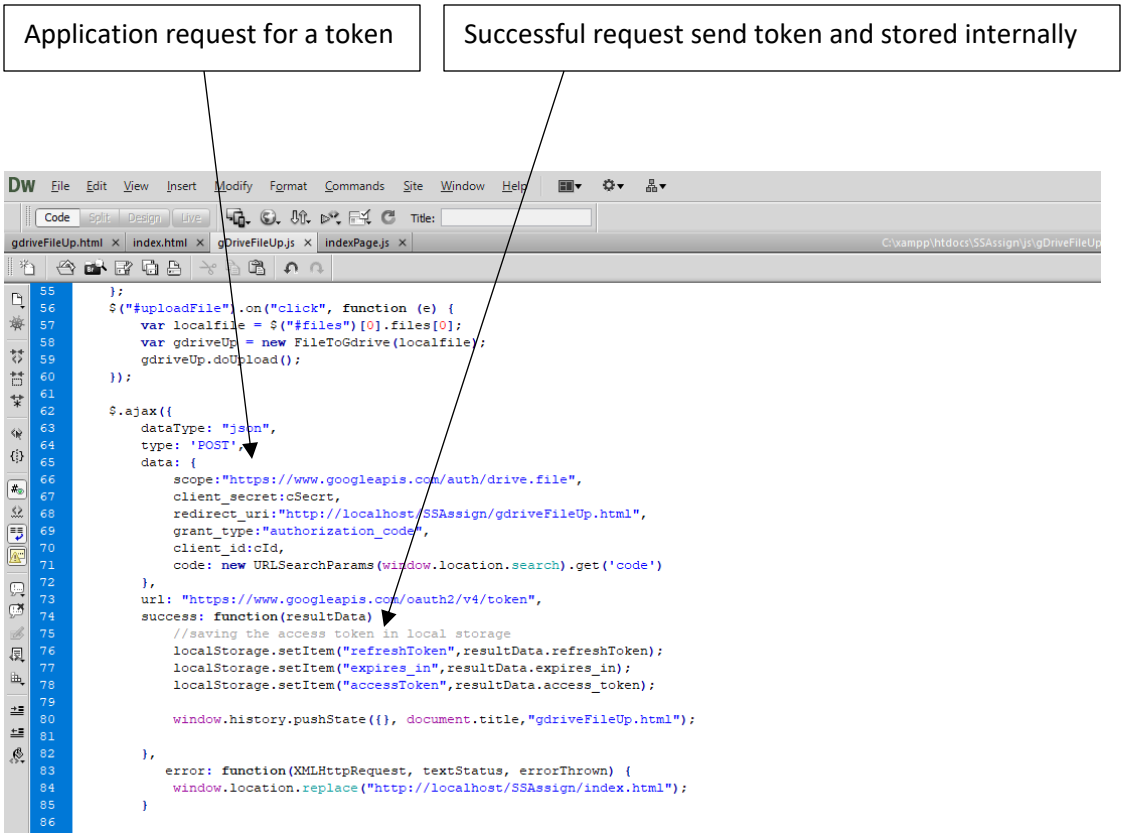


Figure 11: Token request and storing

When a file is chosen to be uploaded and the Upload button is clicked, the resource owner is allowed to upload the file to the own Google Drive.

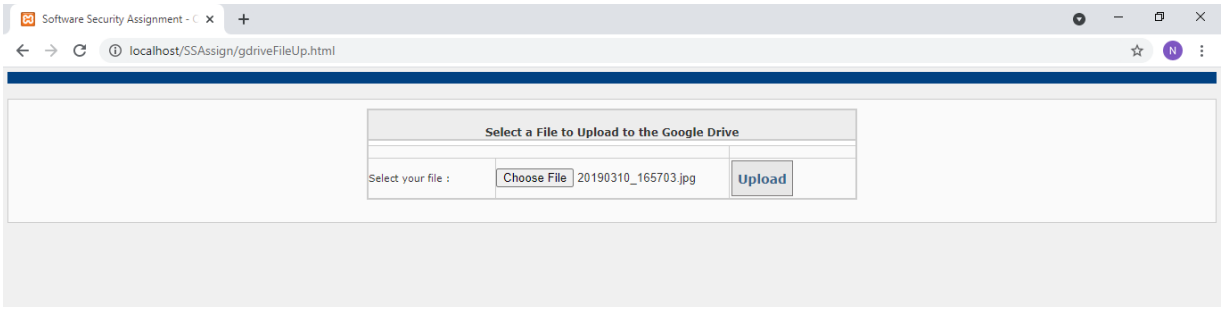
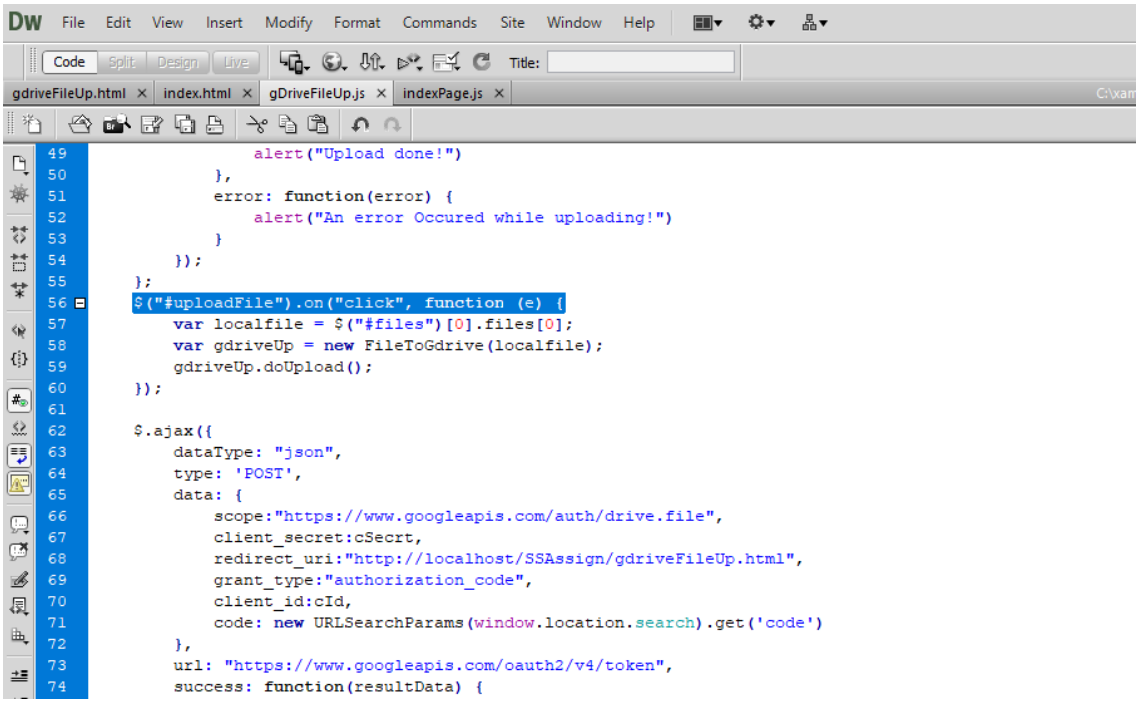


Figure 12: In progress upload

In this upload.html page,a javascript function runs at the button click to process file uploading request.

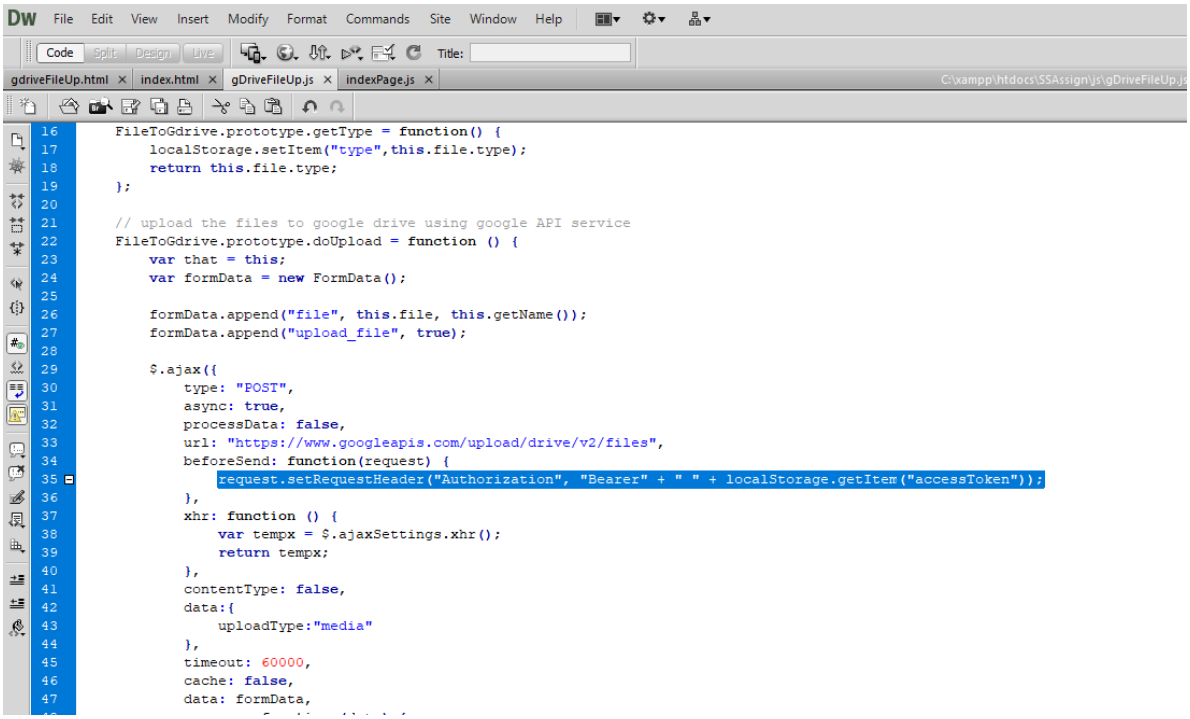


```
                alert("Upload done!")
            },
            error: function(error) {
                alert("An error Occured while uploading!")
            }
        });
    };
    $("#uploadFile").on("click", function (e) {
        var localfile = $("#files")[0].files[0];
        var gdriveUp = new FileToGdrive(localfile);
        gdriveUp.doUpload();
    });

    $.ajax({
        dataType: "json",
        type: 'POST',
        data: {
            scope:"https://www.googleapis.com/auth/drive.file",
            client_secret:cSecrt,
            redirect_uri:"http://localhost/SSAssign/gdriveFileUp.html",
            grant_type:"authorization_code",
            client_id:cId,
            code: new URLSearchParams(window.location.search).get('code')
        },
        url: "https://www.googleapis.com/oauth2/v4/token",
        success: function(resultData) {
```

Figure 13: Code snippet Upload button

Prior to sending the upload, the doUpload function prepares the request headers with the token received by the client application.



```
FileToGdrive.prototype.getType = function() {
    localStorage.setItem("type",this.file.type);
    return this.file.type;
};

// upload the files to google drive using google API service
FileToGdrive.prototype.doUpload = function () {
    var that = this;
    var formData = new FormData();

    formData.append("file", this.file, this.getName());
    formData.append("upload_file", true);

    $.ajax({
        type: "POST",
        async: true,
        processData: false,
        url: "https://www.googleapis.com/upload/drive/v2/files",
        beforeSend: function(request) {
            request.setRequestHeader("Authorization", "Bearer" + " " + localStorage.getItem("accessToken"));
        },
        xhr: function () {
            var tempx = $.ajaxSettings.xhr();
            return tempx;
        },
        contentType: false,
        data:{
            uploadType:"media"
        },
        timeout: 60000,
        cache: false,
        data: formData,
```

Figure 14: Code Snippet on request headers

Once the token is valdated by the resource server and the upload is complete, the client application will display the success message and the resource owner can observe the uploaded file in the Google Drive.

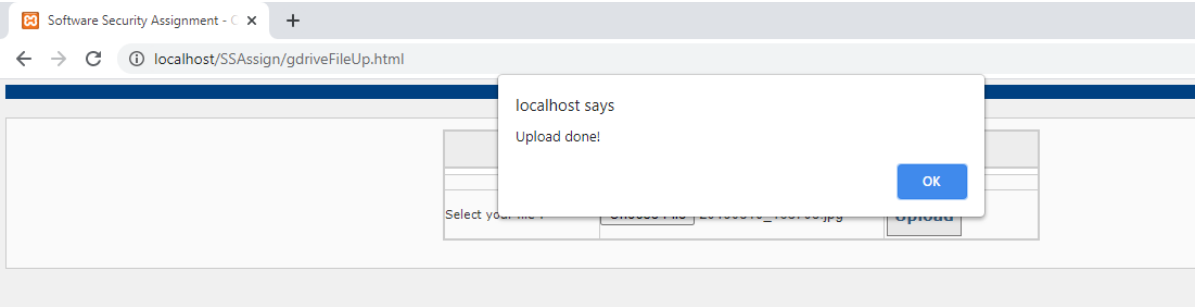


Figure 15: Success Message

Developers have restricted users directly accessing the upload page by capturing the error and redirecting the users back to the main page.

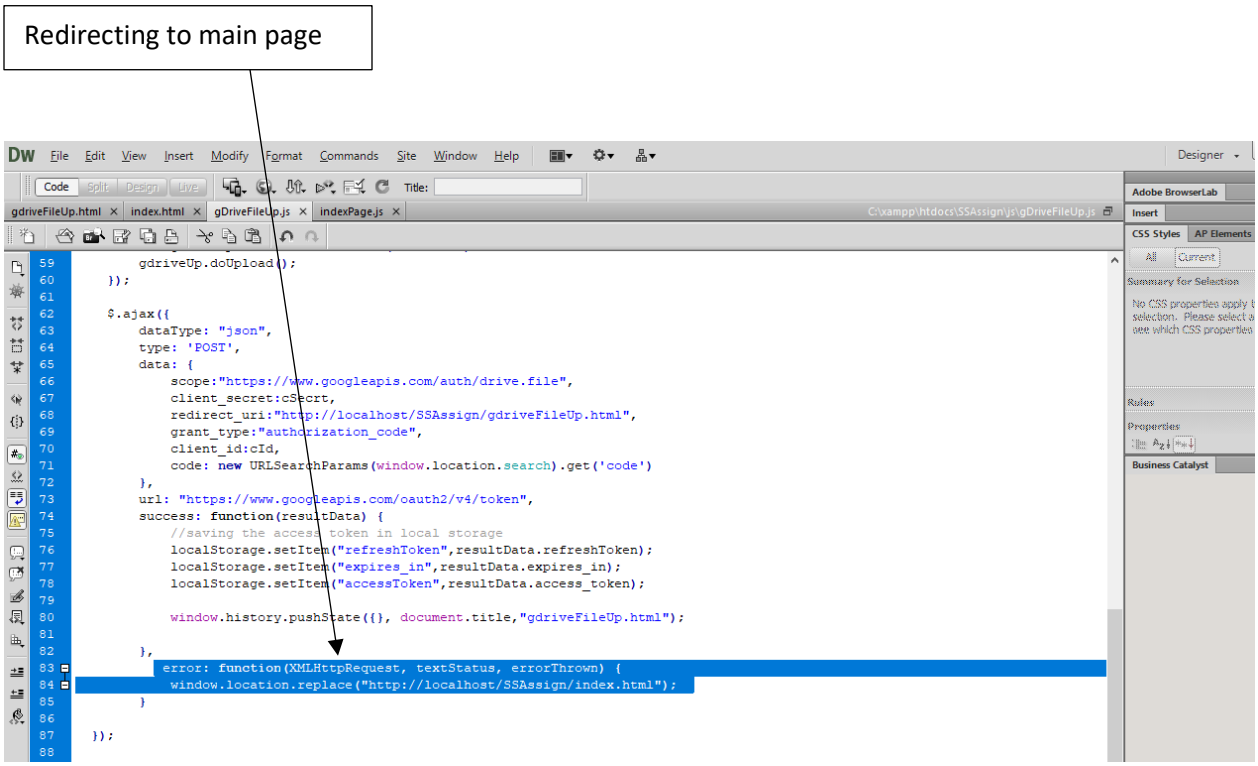


Figure 16: Code Snippet on redirecting back to main page

APPENDIX

Index.html code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
<meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Software Security Assignment - OAuth Project</title>

  <link rel="stylesheet" href="css/css1.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
  <script src="js/indexPage.js"></script>
</head>
<body>
<div align="left">
<table width="100%" cellpadding="0" cellspacing="0">
<tr>
 <td bgcolor="004182"> </td></tr>
</table>
</div>
<p> </p>
        <table width="100%" border="0" cellspacing="0" cellpadding="10" style="border: solid #CCCCCC 1px;">
         <tr>
                <td bgcolor="#FAFAFA">
                 <form name="login" method="post">
<table width="350px" border="0" cellspacing="0" cellpadding="1" align="center">
<tr>
```

```
 <td bgcolor="#CCCCCC">
        <table width="100%" border="0" cellspacing="0" cellpadding="3">
         <tr bgcolor="#EDEDED">
                <td style="border-bottom: solid 1px #CCCCCC;">
                 <p> </p>
                 <h5 align="center">Upload your document via Google Sign In</h5>
                 <p align="center"> </p>
                 </td>
         </tr>
         <tr bgcolor="#FFFFFF">
          <td height="5"></td>
          </tr>
         <tr bgcolor="#FAFAFA">
                <td  align="center"><p><input type="button" name="login2" value="Login with Google"
class="button" onclick="signingoogle()" /></p></td>
                </tr>
        </table>
 </td>
</tr>
</table><br>
</p>
</form>
                </td>
         </tr>
        </table>
</body>
        </html>
```

indexPage.js Code

```
function signingoogle(){
 // generate authentication server link and load
        var webLink = "";
```

```
        var cId = "500909083474-s7p7fo19bam0lle7j7h0plfluqgvsnml.apps.googleusercontent.com";
        var prjScope = "https://www.googleapis.com/auth/drive.file";
        var reURI = "http://localhost/SSAssign/gdriveFileUp.html";
        webLink = "https://accounts.google.com/o/oauth2/v2/auth?redirect_uri="+reURI
        +"&prompt=consent&response_type=code&client_id="+cId+"&scope="+prjScope
        +"&access_type=offline";

        window.location = webLink;
  }
```

gdriveFileUp.html code

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
        <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>Software Security Assignment - OAuth Project</title>

  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>

  <link rel="stylesheet" href="css/css1.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script src="js/gDriveFileUp.js"></script>
</head>
<body>
<div align="left">
<table width="100%" cellpadding="0" cellspacing="0">
<tr>
 <td bgcolor="004182"> </td></tr>
```

```
</table>
</div>
<p> </p>
        <table width="100%" border="0" cellspacing="0" cellpadding="10" style="border: solid #CCCCCC 1px;">
         <tr>
                <td bgcolor="#FAFAFA">
<table width="550px" border="0" cellspacing="0" cellpadding="1" align="center">
<tr>
 <td bgcolor="#CCCCCC">
        <table width="100%" border="0" cellspacing="1" cellpadding="0">
         <tr bgcolor="#EDEDED">
                <td colspan="3" style="border-bottom: solid 1px #CCCCCC;">
                 <p> </p>
                 <h5 align="center">Select a File to Upload to the Google Drive</h5>
                 </td>
         </tr>
         <tr bgcolor="#FFFFFF">
          <td colspan="3" height="5"></td>
          </tr>
         <tr bgcolor="#FAFAFA">
          <td colspan="2" align="center"> </td>
          <td > </td>
          </tr>
         <tr bgcolor="#FAFAFA">
                <td>Select your file :   </td>
                <td width="48%"><input type="file" id="files" name="files[]" /></td>
                <td width="26%" ><p><button id="uploadFile" class="button">Upload</button>
                 </p></td>
         </tr>
        </table>
 </td>
</tr>
```

```
</table><br>
</p>
                    </td>
             </tr>
            </table>
</body>
            </html>
```

gDriveFileUp.js code

```
$(document).ready(function(){
   //set variables and constants to make the auth server link
   var cId = "500909083474-s7p7fo19bam0lle7j7h0plfluqgvsnml.apps.googleusercontent.com";
   const cSecrt = "T0Y3aDpGPqZAe2g3dXASb_9B";
   var access_token= ""; //this will be dynamically populated
   var FileToGdrive = function (file) {
      this.file = file;
   };
   FileToGdrive.prototype.getName = function() {
      return this.file.name;
   };
   FileToGdrive.prototype.getSize = function() {
//get the file size
      localStorage.setItem("size",this.file.size);
      return this.file.size;
   };
   FileToGdrive.prototype.getType = function() {
//type of the file
      localStorage.setItem("type",this.file.type);
      return this.file.type;
   };

   // upload the files to google drive using google API service
   FileToGdrive.prototype.doUpload = function () {
      var that = this;
      var formData = new FormData();

      formData.append("file", this.file, this.getName());
      formData.append("upload_file", true);

      $.ajax({
         type: "POST",
         async: true,
         processData: false,
         url: "https://www.googleapis.com/upload/drive/v2/files",
         beforeSend: function(request) {
            //set request headers
            request.setRequestHeader("Authorization", "Bearer" + " " + localStorage.getItem("accessToken"));
```

```
        },
        xhr: function () {
           var tempx = $.ajaxSettings.xhr();
           return tempx;
        },
        contentType: false,
        data:{
           uploadType:"media"
        },
        timeout: 60000,
        cache: false,
        data: formData,
        success: function (data) {
           alert("Upload done!")
        },
        error: function(error) {
           alert("An error Occured while uploading!")
        }
     });
};
$("#uploadFile").on("click", function (e) {
   var localfile = $("#files")[0].files[0];
   var gdriveUp = new FileToGdrive(localfile);
   gdriveUp.doUpload();
});

$.ajax({
   dataType: "json",
   type: 'POST',
   data: {
      scope:"https://www.googleapis.com/auth/drive.file",
      client_secret:cSecrt,
      redirect_uri:"http://localhost/SSAssign/gdriveFileUp.html",
      grant_type:"authorization_code",
      client_id:cId,
      code: new URLSearchParams(window.location.search).get('code')
   },
   url: "https://www.googleapis.com/oauth2/v4/token",
   success: function(resultData) {
      //saving the access token in local storage
      localStorage.setItem("refreshToken",resultData.refreshToken);
         //token expiry
      localStorage.setItem("expires_in",resultData.expires_in);
         //access token
      localStorage.setItem("accessToken",resultData.access_token);

      window.history.pushState({}, document.title,"gdriveFileUp.html");

   },
     error: function(XMLHttpRequest, textStatus, errorThrown) {
      window.location.replace("http://localhost/SSAssign/index.html");
   }
```

```
  });

});
```

REFERENCES

[1] Oauth.net, "*OAuth 2.0*", oauth.net, Accessed On: Apr.19, 2021 [Online] Available: https://oauth.net/2/

[2] Oracle, "*Introduction to API Gateway OAuth 2.0*", oracle.com, Accessed On: Apr.20, 2021 [Online] Available: https://docs.oracle.com/cd/E50612_01/doc.11122/oauth_guide/content/oauth_intro.html

[3] O.Dulin,"*OAuth 2.0: The protocol at the center of the universe*", infoworld.com, Nov, 24th, 2015, Accessed On: Apr.19,2021 [Online] Available : https://www.infoworld.com/article/3007415/oauth-20-the-protocol-at-the-center-of-the-universe.html