Software Security

Assignment 2

Web Application using OAuth Framework 2.0

Student IDs –

MS19810256

MS19814384

MS19814216

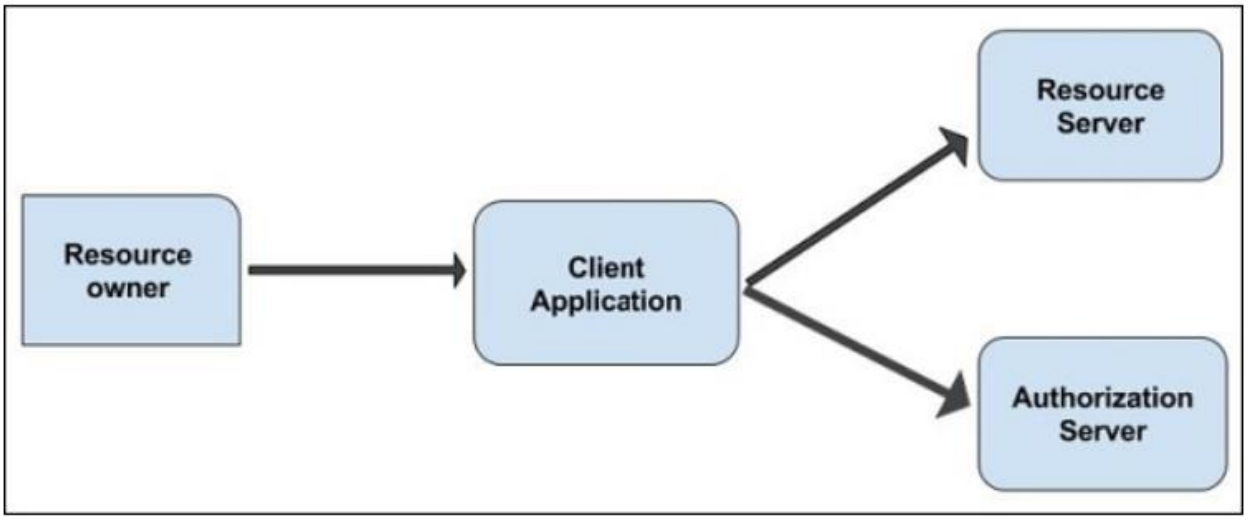
M.Sc. in IT
Sri Lanka Institute of Information Technology

Table of Contents

Introduction

OAuth is a mainly authorization framework which allows to access and communicate between two difference applications. Which means OAuth will provide function to logging to third party web or mobile application without not exchange username and password. When user logging into the third part application this framework interferes with particular application targeted platform. After that application ask from logging credentials, but user giving credential only communicate vis OAuth framework. This authorization mechanism allows by different type of vendors like Facebook, google and other cloud service providers.

Mainly there are four roles in OAuth framework.



Resource server – this means server API are hosted and can be accessed by users to exchange their username and password.

Resource owner – resource owner is which is the main application trying to get access by using resource server.

Client application – client application is what is the method, platform is accessed by user to connect with resource owner. Either could be mobile application, web application.

Authentication server – getting access token and other resources from resource owner and provide permission to connect with resource owner.

Apart that there is some other concept are also available in this framework. Like web server, user agent, and native application. But mainly four concepts are actively participating in OAuth framework. This open standard method has some versions and currently using OAuth 2.0.

Workflow

There is way how this framework performs with third party application and resource owner application model.

Step 1 – as a first step, one user access resource server using their client application. Client application with direct user to resource owner.

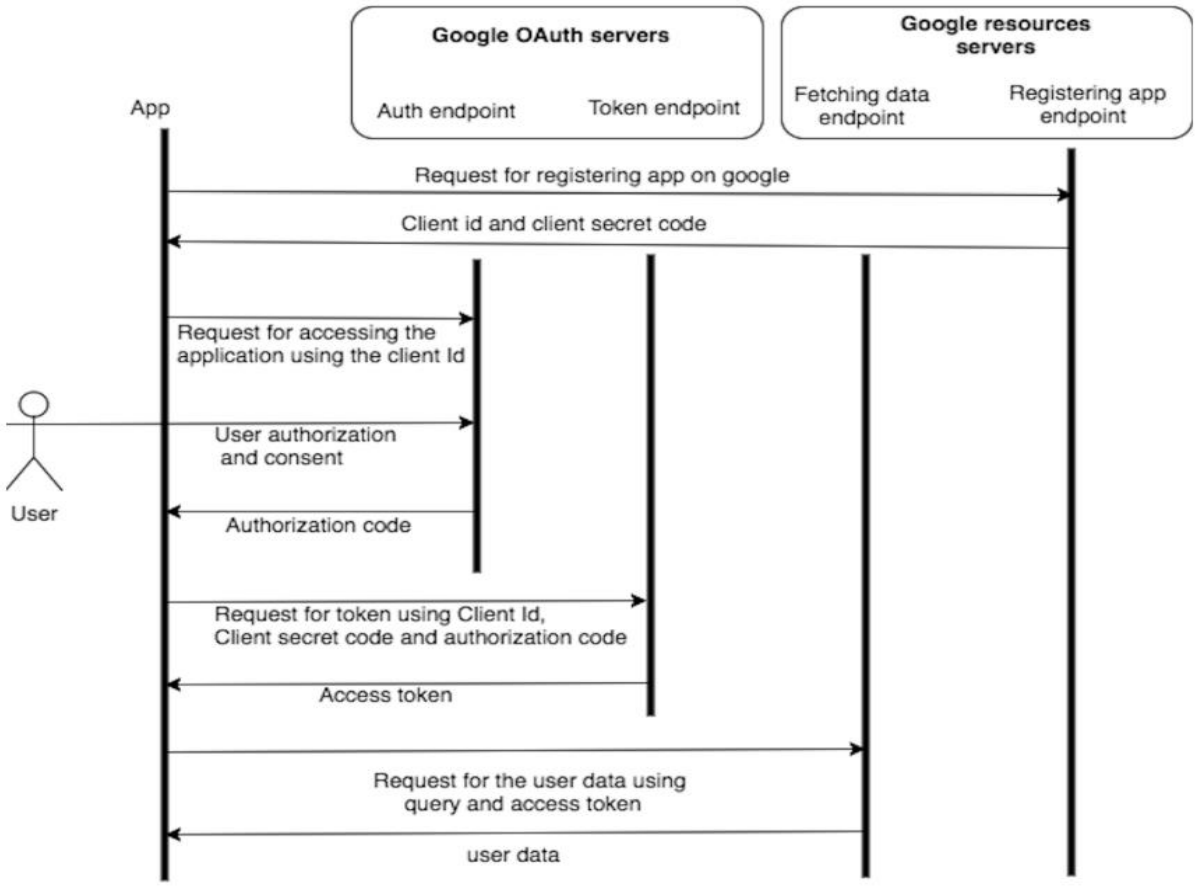
Step 2 – in first step after direct user to resource owner, credentials will request by resource owner and authorization server start to interfere with user application.

Step 3 – after user give related credential to authorization server, authorization cede will issue by authorization server to third party application.

Step 4 – user give credentials, authorization code with redirect URI (Uniform Resource identifier) to authorization server.

Step 6 – authorization server issue access token to user, access resource owner through user application.

Step 7 – user successfully logging into resource owner server.

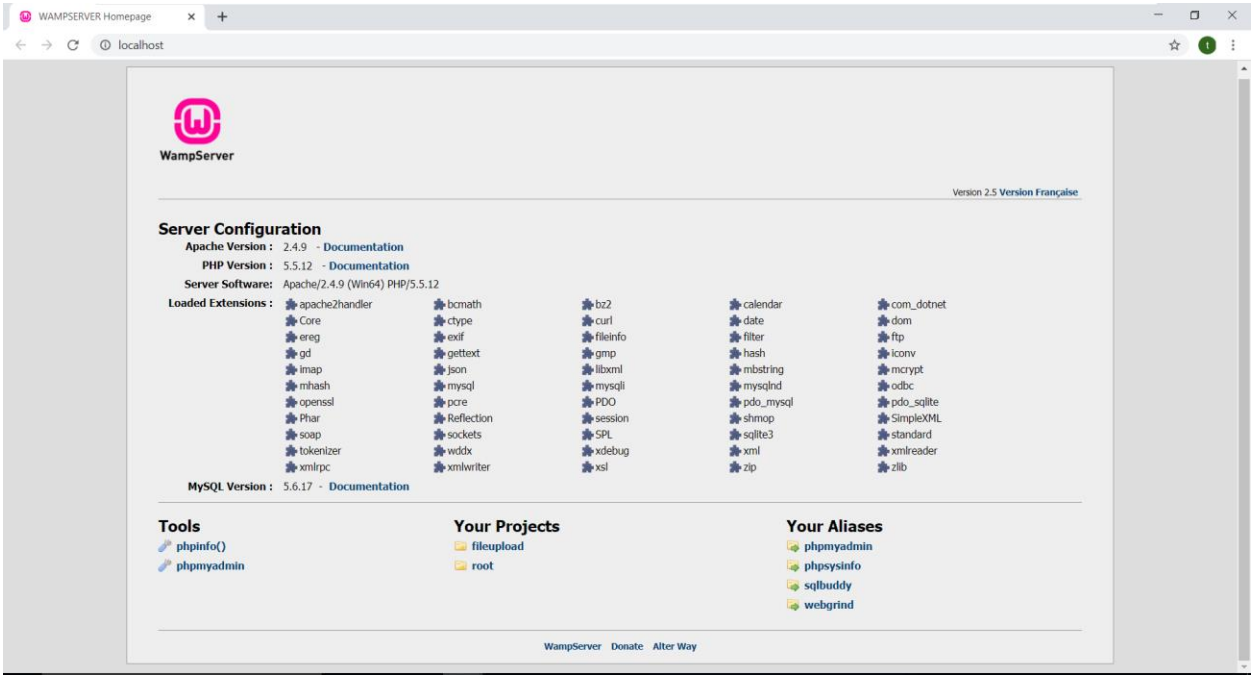


Application and Implementation

As an application, developed web-based file upload application. Mainly application run on web browser, user can upload files to google drive. Simply user click on upload button and it direct to google drive authentication and provide functionality to upload file for logging user's google drive.

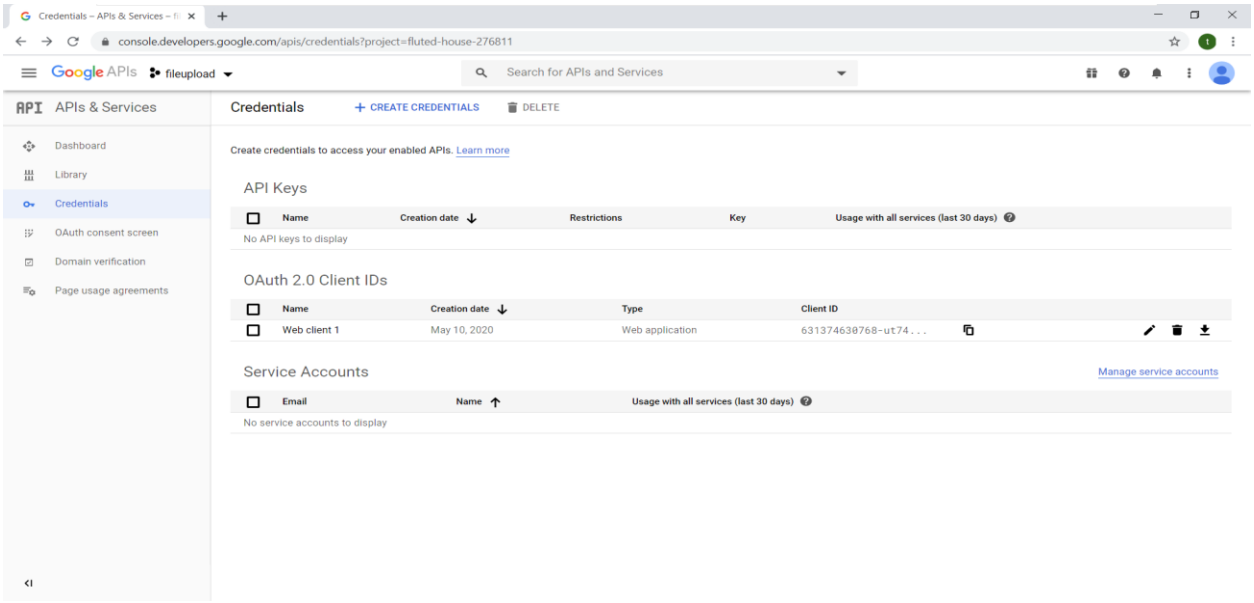
Implementation

This is web application that runs on web server, for demonstration purpose have run on WAMP server and as localhost.

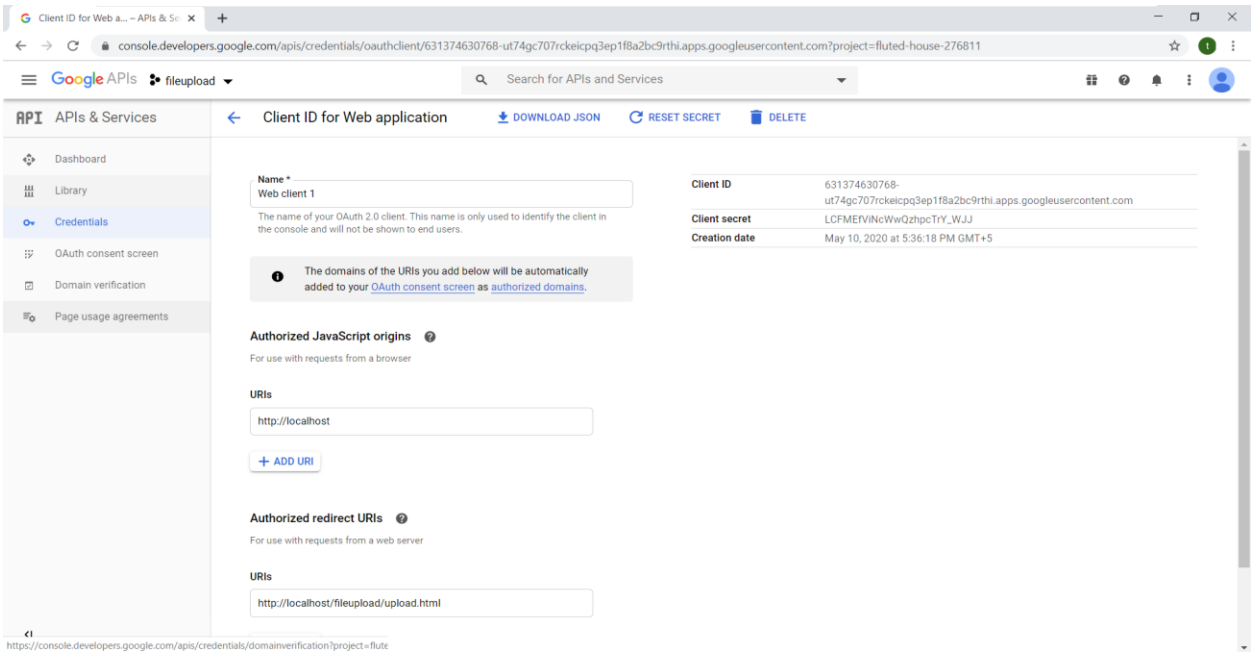


As a language have been used HTML, CSS and JavaScript to develop this web application. HTML language use for design form of the web page, button and form of the web application. CSS language use for design, background colors, interactive purposes. JavaScript used for run functions, task of web page that has to perform connect with google drive, logging into drive, exchange details with resource server and other functions.

After developing web application there is requirement of implementing OAuth client ID on selected or prefer identity server. In this case use Google API for implement client ID and OAuth credentials.



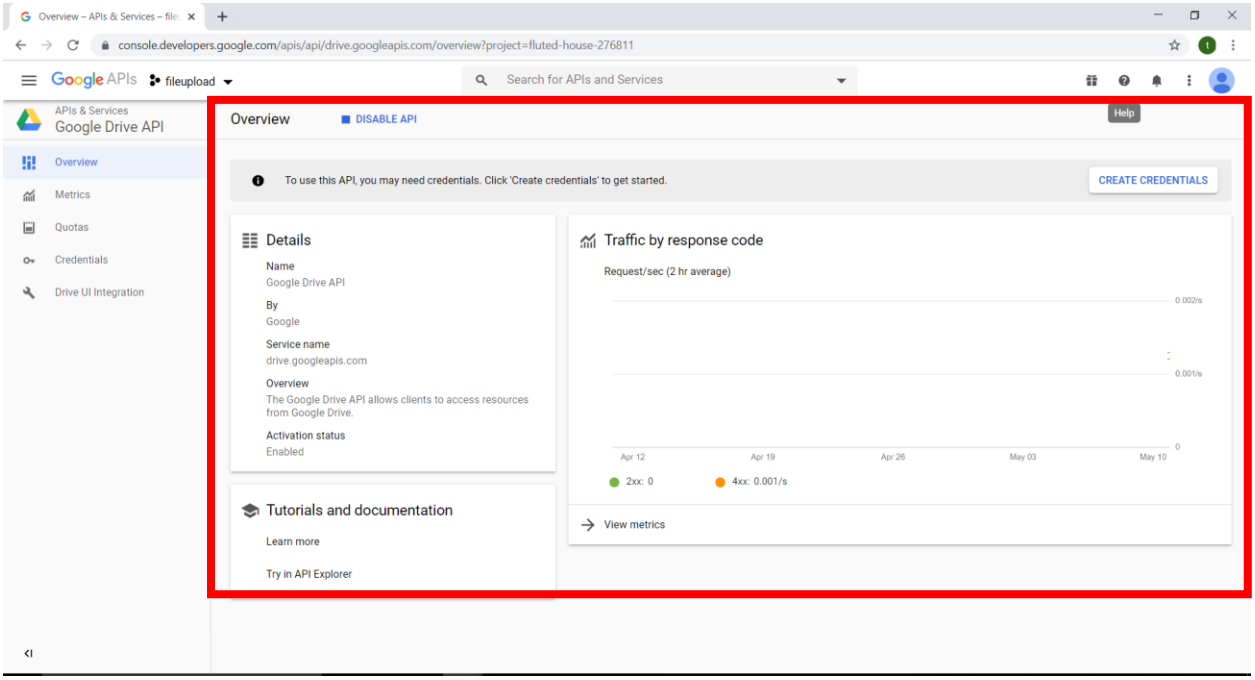
OAuth client ID created name as web client 1 in google developer console (Google API).



In the “web client 1” can be find client ID, client secret and creation date of OAuth client ID. This client ID and client secret will use for authentication of user with google drive via third party web application.

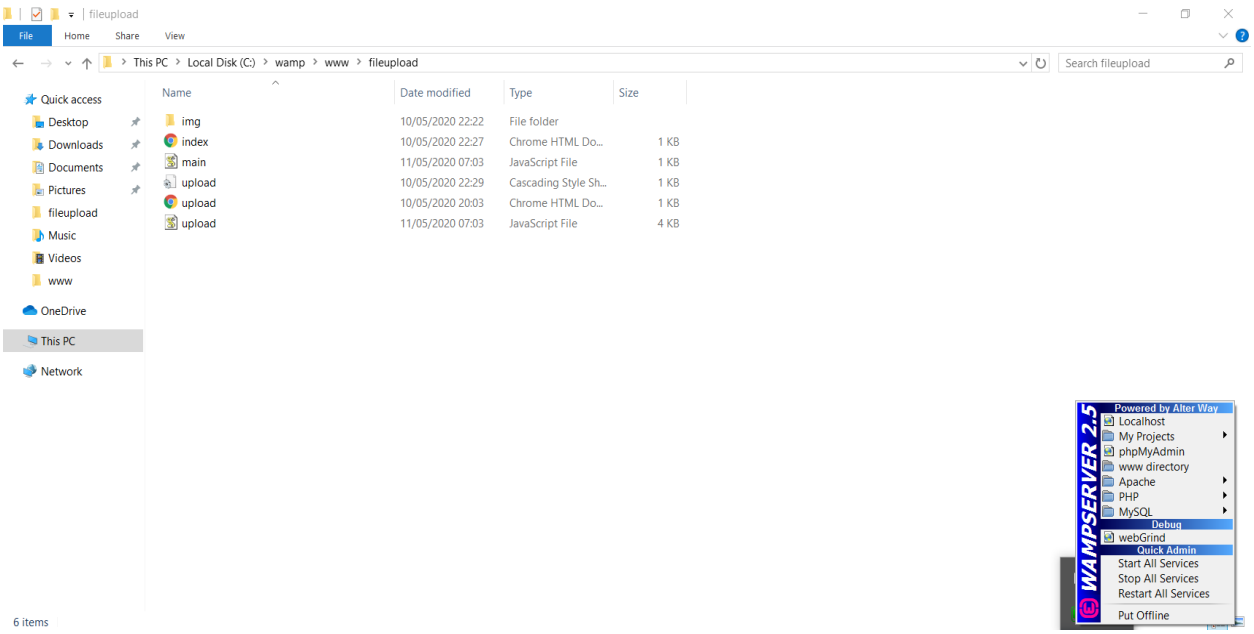
Client ID	631374630768-ut74gc707rckeicpq3ep1f8a2bc9rthi.apps.googleusercontent.com
Client secret	LCFMEfViNcWwQzhpcTrY_WJJ
Creation date	May 10, 2020 at 5:36:18 PM GMT+5

After create OAuth client ID, need to configure specific API that related to application going to develop. This application design for upload file to google drive. Hence, in the Google API console API named “Google Drive API” should be enable. It gives permission to application access without any issue. Once enable Google API, it will be displayed like below.

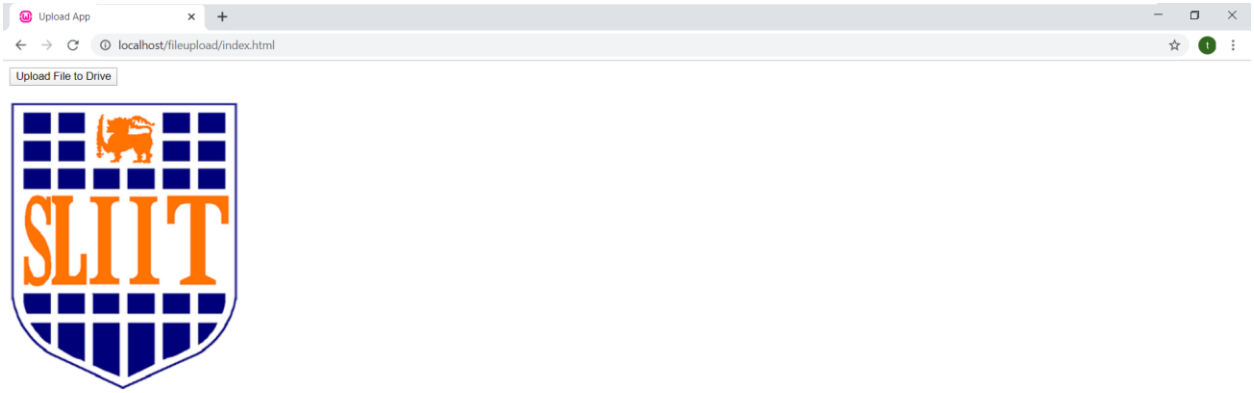


Flows

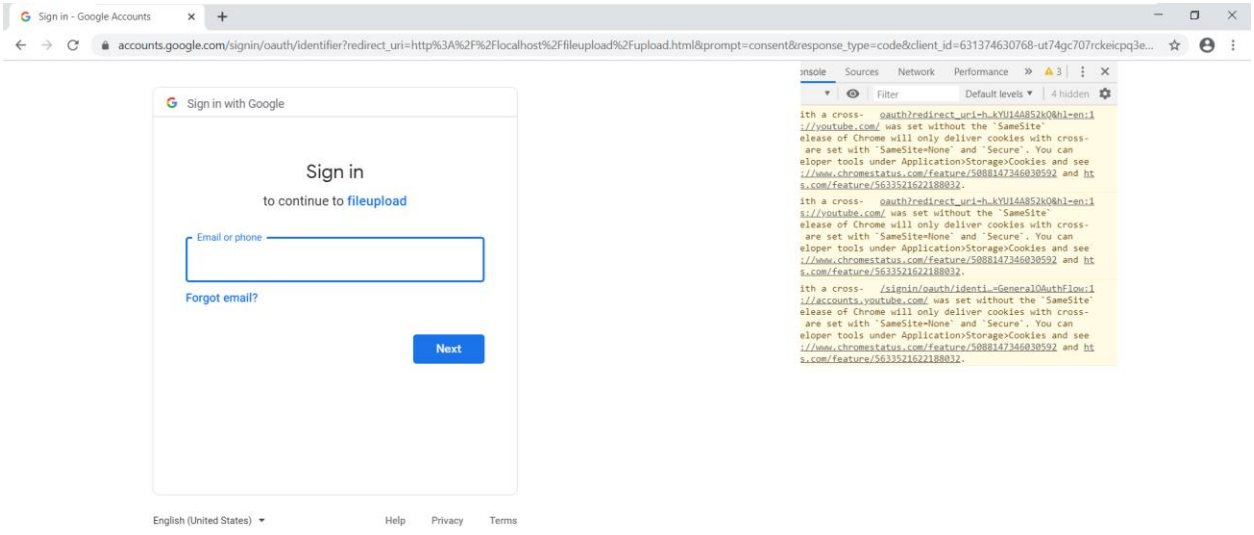
Step 1 – run WAMP server ensue web application folder inside the directory. This application folder named “fileupload”



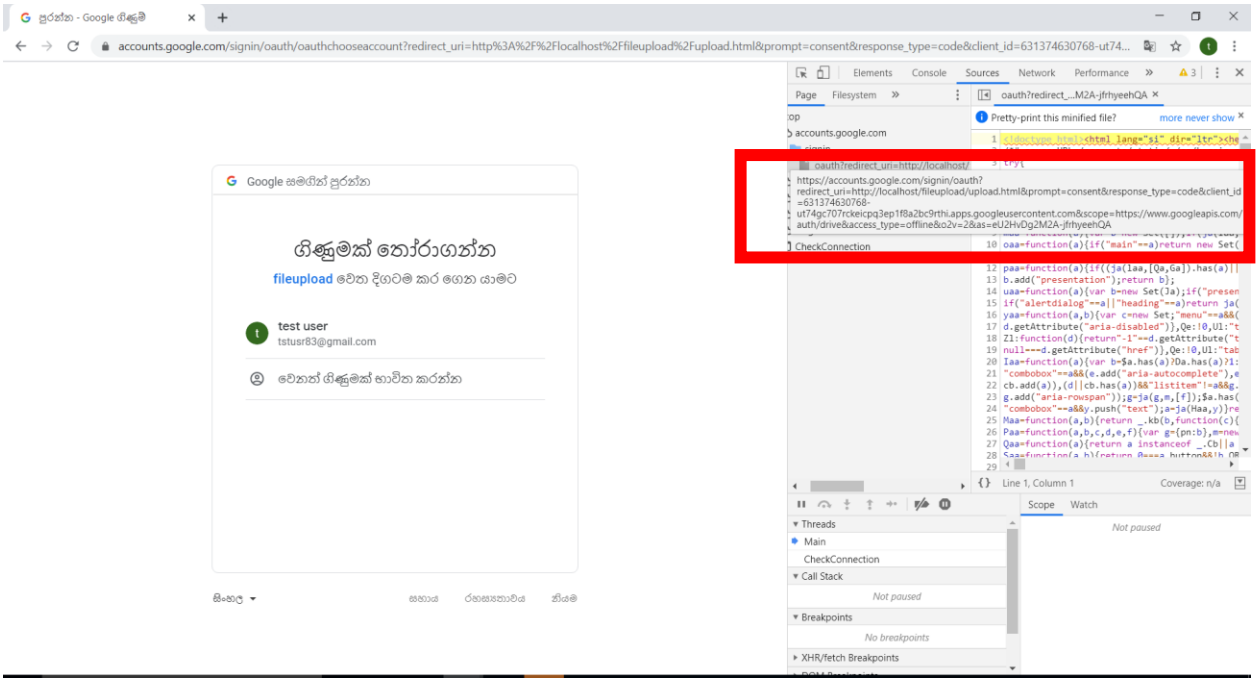
Step 2 – open web browser and type web page name. this scenario web page hosted in local host and URL is http://localhost/fileupload/index.html



Step 3 – click on the "upload to drive button and it will direct to Google logging page and to enter username and password of the google account. After enter credentials it will ask for permission.

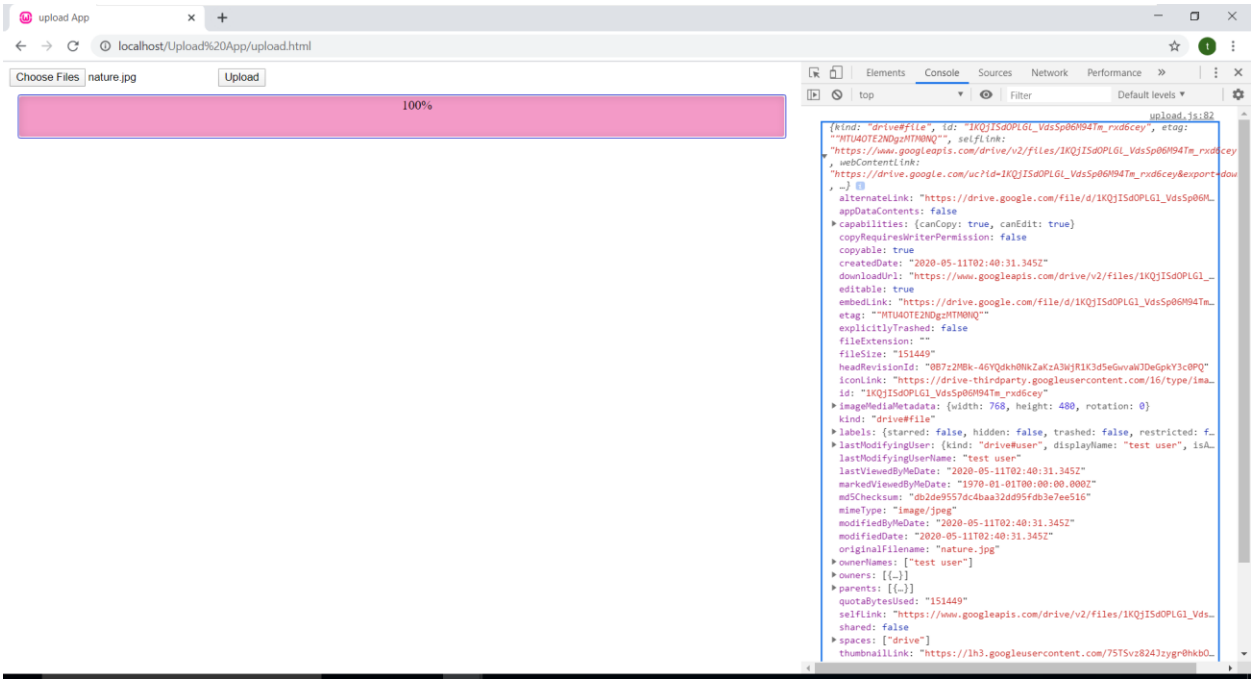


After user click on upload file button and then it will direct to google logging page, then web application request client id with generated in OAuth. Below image highlighted area with client id request from google sever.

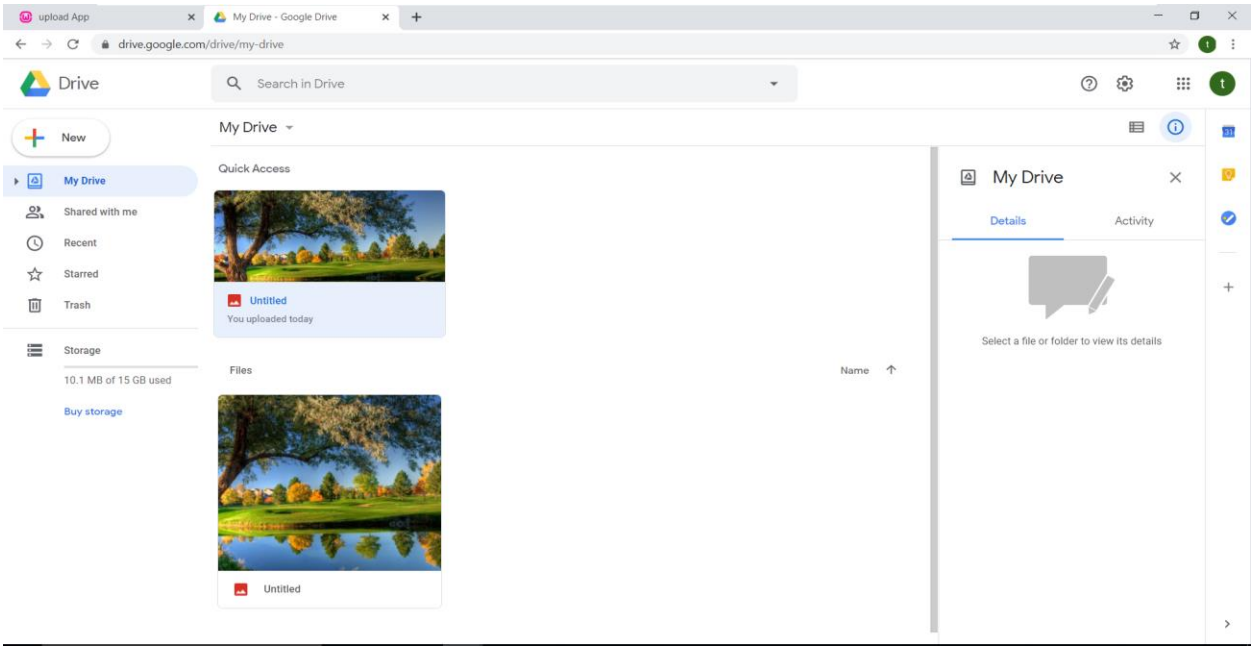


Step 4 – after approve logging permission of google authentication it will redirect to upload page of app.

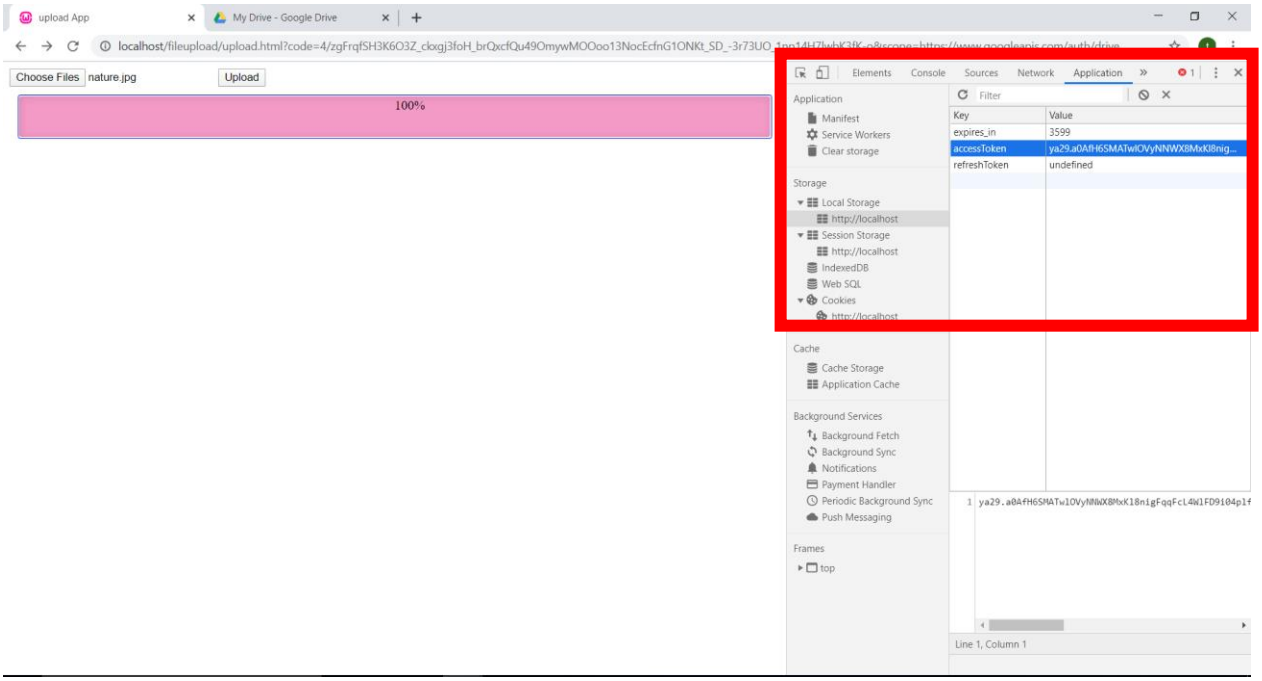
Step 5 – select files to upload and click upload button.



Step 6 – after go to drive.google.com, can be found uploaded image via web application in drive



Step 7 – once completed the authentication with and logged into google drive login, can upload file to drive. Then go to browser>right click>inspect>application tab. In the application tab area under local storage, URL of the server can be seen. When click that URL token access will display. In below image highlighted area displayed token access given by Google API OAuth framework when enter credential to google drive as feedback instead of giving credential details to third party application directly.



Conclusion.

When using this type of authorization mechanism will provide more security instead of using credential exchange directly with the resource owner. In a current period, security and confidentiality become more important than other criteria. But this framework reduces that risk by interfering with third party web applications, mobile application or any other applications. Even thought very secure way there might be some critical points. As an example, in this web application running on hosted web server and that server allow external parties to use application. Then one user is using this application but, user's browser not up to date and may be with serios vulnerabilities. If a hacker is monitoring this task, but he cannot attack on credential exchange part, because it stronger. But user's browser vulnerable, he can gain access very easily and one access browser attacker can get access to Google drive via access token. So mainly this framework avoid critical aspect of information leakage and it will be useful in different type of platform and difference type of devices.

Appendix

Index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Upload App</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script src="main.js"></script>
</head>
<body>
  <div>
      <button id="login">
          Upload File to Drive
      </button>
  </div>

<h2>
<div class="imgcontainer">
       <img src="img/sliit.png" alt="sliit" class="sliit">
</div>

</h2>
</body>
</html>
```

Main.js

```
// 631374630768-ut74gc707rckeicpq3ep1f8a2bc9rthi.apps.googleusercontent.com   -clienid
// LCFMEfViNcWwQzhpcTrY_WJJ   -clientsecret


$(document).ready(function(){
    var clientId = "631374630768-ut74gc707rckeicpq3ep1f8a2bc9rthi.apps.googleusercontent.com";
    var redirect_uri = "http://localhost/fileupload/upload.html";
    var scope = "https://www.googleapis.com/auth/drive";


 var url = "http://localhost";
    $("#login").click(function(){
      signIn(clientId,redirect_uri,scope,url);


    });
    function signIn(clientId,redirect_uri,scope,url){
      url = "https://accounts.google.com/o/oauth2/v2/auth?redirect_uri="+redirect_uri
      +"&prompt=consent&response_type=code&client_id="+clientId+"&scope="+scope
      +"&access_type=offline";



      window.location = url;
    }




});
```

Upload.css

```
#progress-wrp {
    border: 1px solid #0009CC;
    padding: 1px;
    position: relative;
    height: 50px;
    border-radius: 3px;
    margin: 10px;
    text-align: left;
    background: #fff;
    box-shadow: inset 1px 3px 6px rgba(0, 0, 0, 0.12);
  }

  #progress-wrp .progress-bar {
    height: 100%;
    border-radius: 3px;
    background-color: #f39ac7;
    width: 0;
    box-shadow: inset 1px 1px 10px rgba(0, 0, 0, 0.11);
  }

  #progress-wrp .status {
    top: 3px;
    left: 50%;
    position: absolute;
    display: inline-block;
    color: #000000;
  }
```

Upload.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>upload App</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" type="text/css" media="screen" href="upload.css" />
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
    <script src="upload.js"></script>
</head>
<body>
  <div>
     <input id="files" type="file" name="files[]" multiple/>
     <button id="upload">Upload</button>
     <div id="progress-wrp">
        <div class="progress-bar"></div>
        <div class="status">0%</div>
    </div>
  </div>
  <div id="result"
  </div>
</body>
</html>
```

Upload.js

```
$(document).ready(function(){
    const urlParams = new URLSearchParams(window.location.search);
    const code = urlParams.get('code');
    const redirect_uri = "http://localhost/fileupload/upload.html"
    const client_secret = "LCFMEfViNcWwQzhpcTrY_WJJ";
    const scope = "https://www.googleapis.com/auth/drive";
    var access_token= "";
    var client_id = "631374630768-ut74gc707rckeicpq3ep1f8a2bc9rthi.apps.googleusercontent.com"

    $.ajax({
        type: 'POST',
        url: "https://www.googleapis.com/oauth2/v4/token",
        data: {code:code
            ,redirect_uri:redirect_uri,
            client_secret:client_secret,
        client_id:client_id,
        scope:scope,
        grant_type:"authorization_code"},
        dataType: "json",
        success: function(resultData) {
          localStorage.setItem("accessToken",resultData.access_token);
          localStorage.setItem("refreshToken",resultData.refreshToken);
          localStorage.setItem("expires_in",resultData.expires_in);
          window.history.pushState({}, document.title, "/Upload App/" + "upload.html");

        }
  });
```

```
function stripQueryStringAndHashFromPath(url) {
    return url.split("?")[0].split("#")[0];
}
var Upload = function (file) {
    this.file = file;
};

Upload.prototype.getType = function() {
    localStorage.setItem("type",this.file.type);
    return this.file.type;
};
Upload.prototype.getSize = function() {
    localStorage.setItem("size",this.file.size);
    return this.file.size;
};
Upload.prototype.getName = function() {
    return this.file.name;
};
Upload.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",
        beforeSend: function(request) {
```

```
            request.setRequestHeader("Authorization", "Bearer" + " " + localStorage.getItem("accessToken"));
        },
        url: "https://www.googleapis.com/upload/drive/v2/files",
        data:{
            uploadType:"media"
        },
        xhr: function () {
            var myXhr = $.ajaxSettings.xhr();
            if (myXhr.upload) {
                myXhr.upload.addEventListener('progress', that.progressHandling, false);
            }
            return myXhr;
        },
        success: function (data) {
            console.log(data);
        },
        error: function (error) {
            console.log(error);
        },
        async: true,
        data: formData,
        cache: false,
        contentType: false,
        processData: false,
        timeout: 60000
    });
};
Upload.prototype.progressHandling = function (event) {
```

```
        var percent = 0;
        var position = event.loaded || event.position;
        var total = event.total;
        var progress_bar_id = "#progress-wrp";
        if (event.lengthComputable) {
            percent = Math.ceil(position / total * 100);
        }
        $(progress_bar_id + " .progress-bar").css("width", +percent + "%");
        $(progress_bar_id + " .status").text(percent + "%");
    };
    $("#upload").on("click", function (e) {
        var file = $("#files")[0].files[0];
        var upload = new Upload(file);
          upload.doUpload();
    });
});
```