Geeshan Kumarasena

MS20918392

File upload component with google oauth api

Contents

[Introduction 2](#_Toc69678052)

[High level flow 2](#_Toc69678053)

[Steps 2](#_Toc69678054)

[Execution sequence 3](#_Toc69678055)

[App in Action 4](#_Toc69678056)

[Appendix 8](#_Toc69678057)

[File Upload Component. 8](#_Toc69678058)

[File Upload Service 11](#_Toc69678059)

## Introduction

This document explains the basic usage of OAuth protocol using a simple Angular app. The App will use the implicit grant type to get an access token on behalf of the user. The user needs to give the consent for the app to use profile information and google drive access. The app developer needs to register with Google beforehand to get the client id which is used to identify and authenticate the app request.

## High level flow

Following diagram depicts the high-level flow of the app functionality.

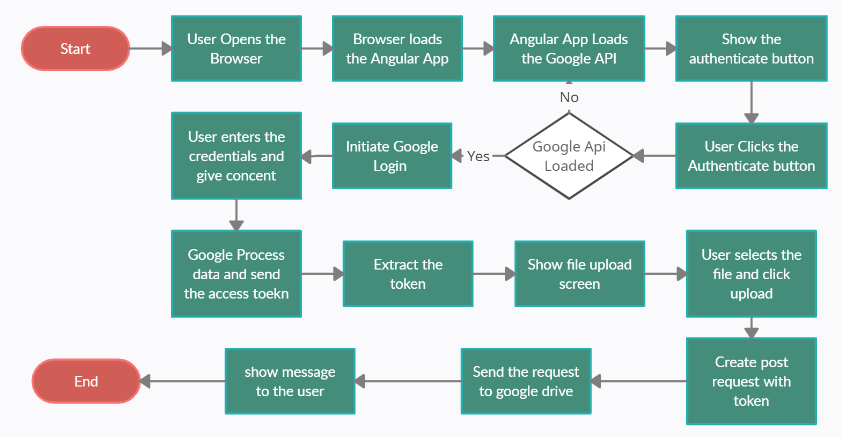


Figure 1-High level flow

### Steps

1. User opens the web browser with Angular app on port 8080. Port 8080 should be used since the app is registered to use localhost:8080 in Google developer console.
2. Angular app loads which setup the base parameters in Google Api. Following are the parameters
   1. client\_id – This is the id given to the app by google. Following is the client id for the angular app.

921335835339-9duhv4rhjp0tna5i4v09utsiak9h0pl7.apps.googleusercontent.com

* 1. Scope – this is the list scopes which the user gives consent to. Following are the scopes used.
     1. Profile
     2. Email
     3. <https://www.googleapis.com/auth/drivews>

1. Once the setup is completed, Angular app shows the “Authenticate” button.
2. Once clicked, authenticate button will initiate the google authentication by opening a popup window with google login.
3. User can enter the credentials in the google login window and should given consent for the app to use the mentioned scopes.
4. Google process the entered credentials and send the response. The response contains the access token.
5. App extracts the access token and show the file upload screen.
6. User selects the desired file and click “Upload to Google Drive”
7. App reads the file content, build the request by including the access token and post the content to the google drive api.
8. Google drive process the request, save the file and respond back.

## Execution sequence

Following diagram shows the sequence of the events.

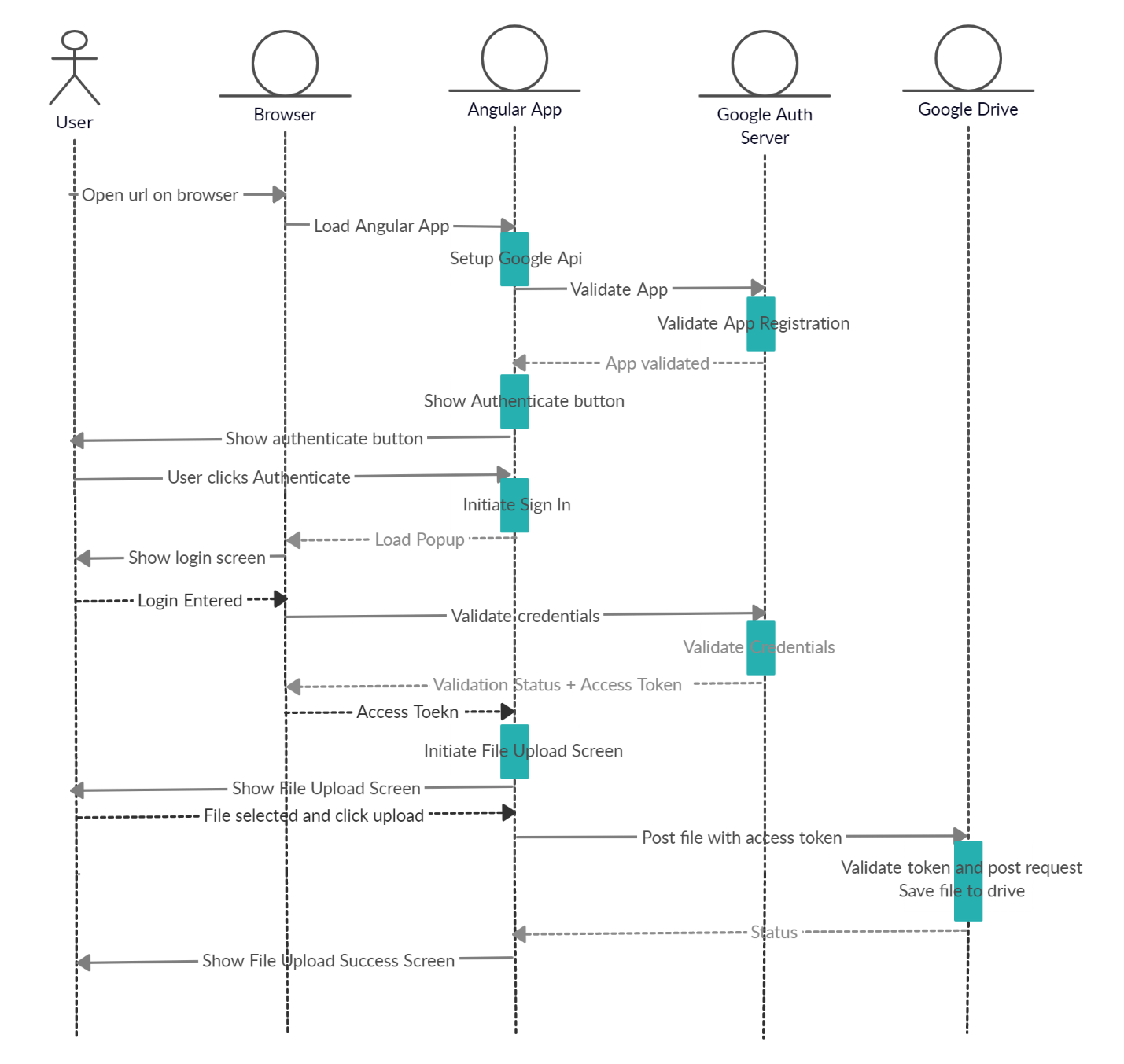
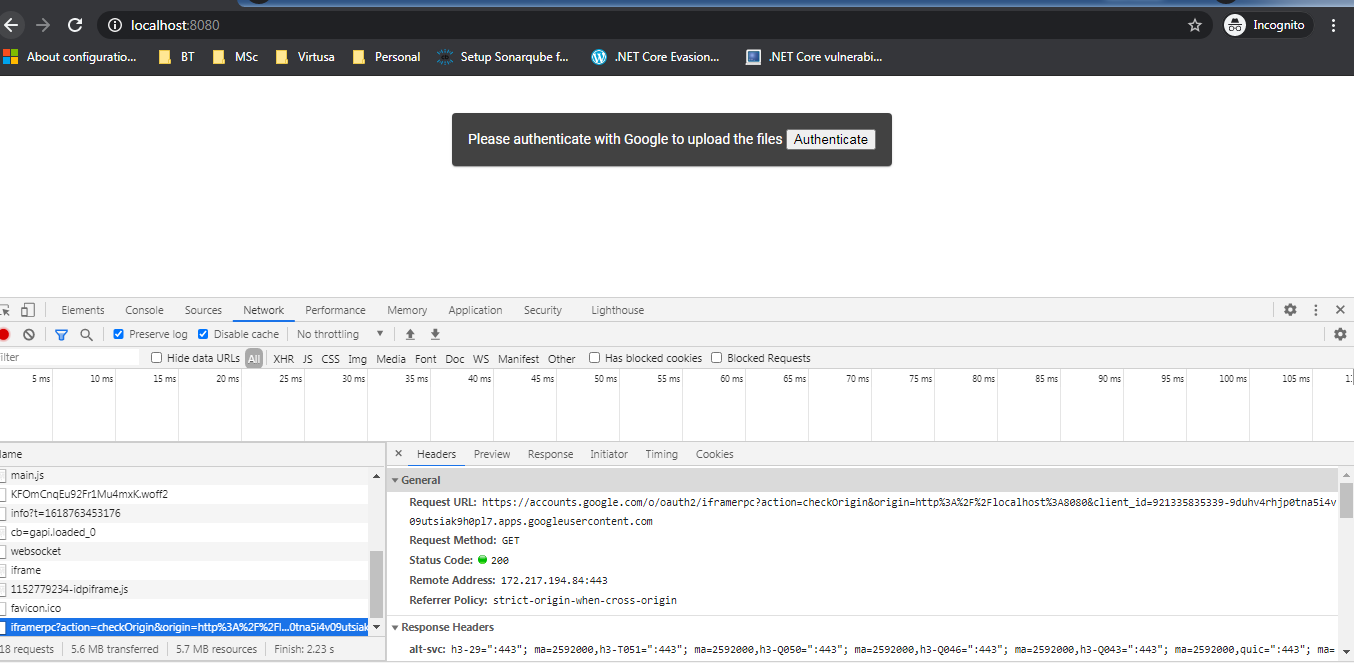


Figure -Sequence Diagram

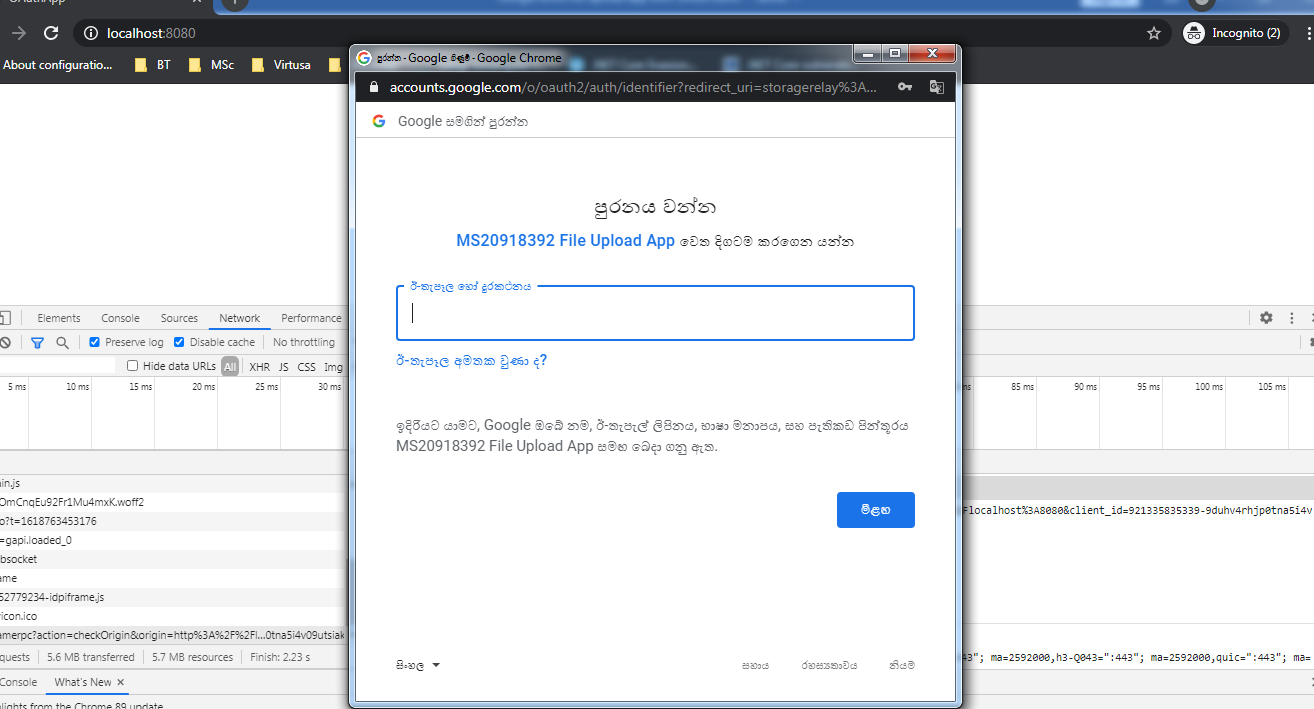
## App in Action

Following set of screen grabs show the above sequence in action.

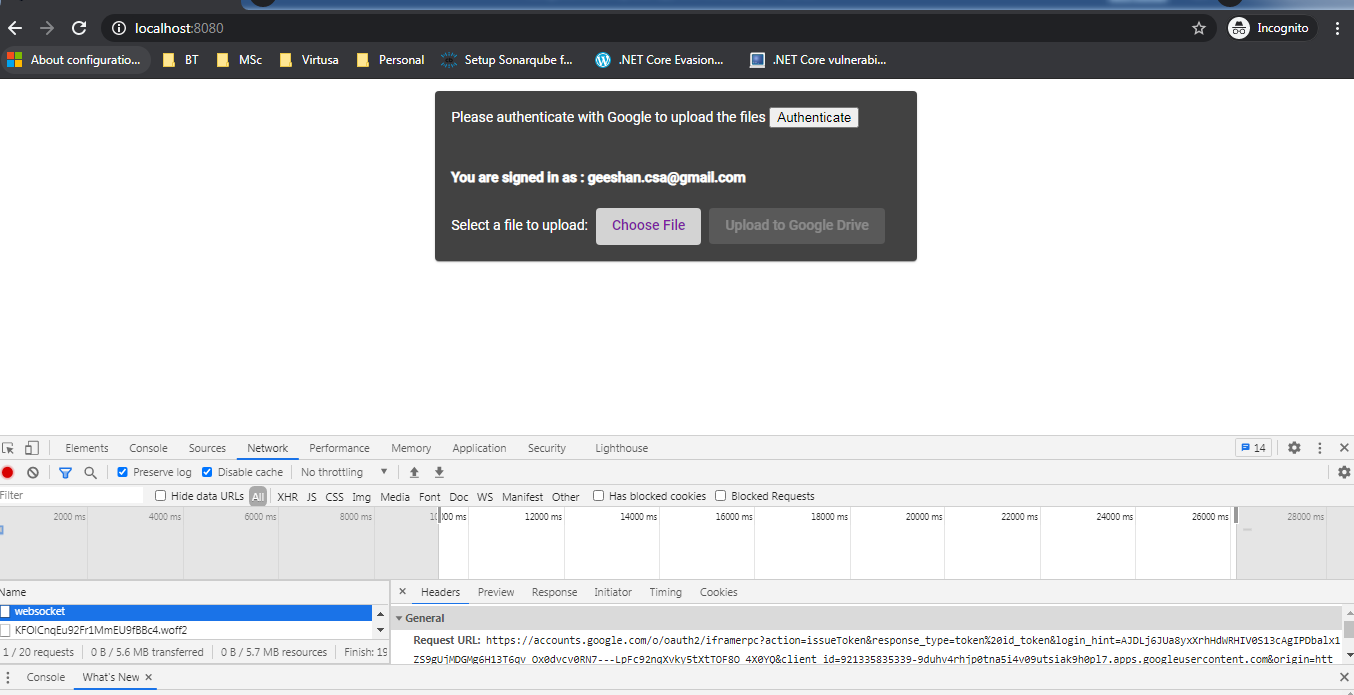
1. User opens app on port 8080.
2. NgOnInit calls the Google Api setup and calls the OAuth2 endpoint for client app validation.
3. Google validates the app and respond with success.



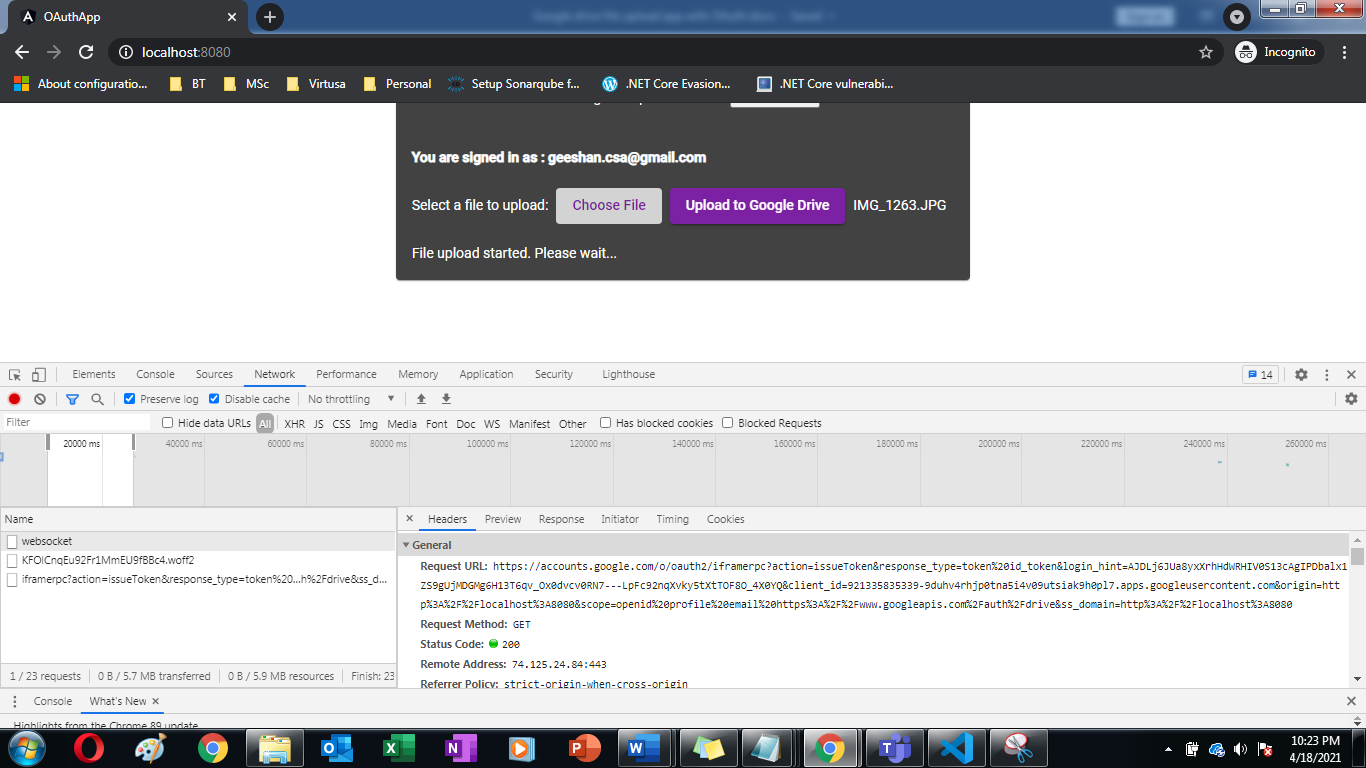
1. User clicks on authenticate button.
2. Angular app execute sign in functionality and browser opens the google sign in window.
3. User enters the credentials and proceed.



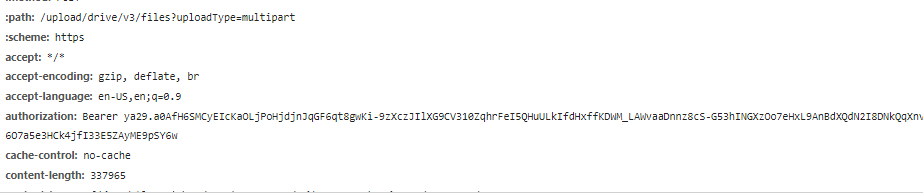
1. Google validates the credentials and return the access token.
2. App extract user information and show file upload screen.



1. User selects the file and click “Upload to Google Drive”
2. Angular app extracts the token from auth response and attach it to the file upload request.



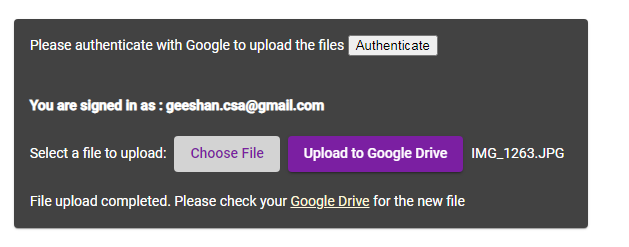
1. Auth token in file upload request



1. File uploaded to google drive.



1. File upload success message in Angular app



## Appendix

### File Upload Component.

import { state } from '@angular/animations';

import { HttpEvent, HttpEventType } from '@angular/common/http';

import { Component, OnInit } from '@angular/core';

import { FileUploadService } from '../services/fileupload.service';

@Component({

selector: 'app-file-upload',

templateUrl: './file-upload.component.html',

styleUrls: ['./file-upload.component.scss']

})

export class FileUploadComponent implements OnInit {

gapiSetupCompleted: boolean = false;

googleAuthenticationInstance: gapi.auth2.GoogleAuth;

error: string;

user: gapi.auth2.GoogleUser;

signedInEmail: string;

fileUploadSuccess:boolean=false;

fileUploadStarted:boolean=false;

constructor(private uploadService: FileUploadService) { }

async ngOnInit() {

this.fileUploadStarted=false;

this.fileUploadSuccess = false;

if (await this.checkIfUserAuthenticated()) {

this.user = this.googleAuthenticationInstance.currentUser.get();

}

}

async initGoogleAuth():Promise<void> {

const loadedGapi = new Promise((resolve) => {

gapi.load('auth2', resolve);

});

return loadedGapi.then(async () => {

await gapi.auth2

.init({

client\_id: '921335835339-9duhv4rhjp0tna5i4v09utsiak9h0pl7.apps.googleusercontent.com',

scope: 'profile email https://www.googleapis.com/auth/drive'

})

.then(authInstance => {

this.gapiSetupCompleted = true;

this.googleAuthenticationInstance = authInstance;

});

});

}

async authenticate(): Promise<gapi.auth2.GoogleUser> {

if (!this.gapiSetupCompleted) {

await this.initGoogleAuth();

}

return new Promise(async () => {

await this.googleAuthenticationInstance.signIn().then(

user => {

this.user = user,

this.signedInEmail = this.googleAuthenticationInstance.currentUser.get().getBasicProfile().getEmail()

},

error => this.error = error);

});

}

async checkIfUserAuthenticated(): Promise<boolean> {

if (!this.gapiSetupCompleted) {

await this.initGoogleAuth();

}

return this.googleAuthenticationInstance.isSignedIn.get();

}

onUploadClicked(event) {

this.fileUploadStarted=true;

if (event.length > 0) {

this.uploadService.uploadFile(event[0], this.googleAuthenticationInstance.currentUser.get().getAuthResponse(true).access\_token)

.then( event =>

{

this.fileUploadStarted=false;

this.fileUploadSuccess = true;

});

}

}

}

### File Upload Service

import { Injectable } from '@angular/core';

import { HttpClient, HttpHeaders } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root'

})

export class FileUploadService {

googleApiUrl: string = "https://www.googleapis.com/upload/drive/v3/files?uploadType=media";

constructor(private http: HttpClient) { }

uploadFile(file: any, token: any) : Promise<any>{

const metadata = { name: file.name, mimeType: file.type };

let form = new FormData();

form.append('metadata', new Blob([JSON.stringify(metadata)], { type: 'application/json' }));

form.append('media', new Blob([file], { type: file.type}));

return fetch('https://www.googleapis.com/upload/drive/v3/files?uploadType=multipart', {

method: 'POST',

headers: new Headers({'Authorization': 'Bearer ' + token}),

body: form

}).then();

}

}