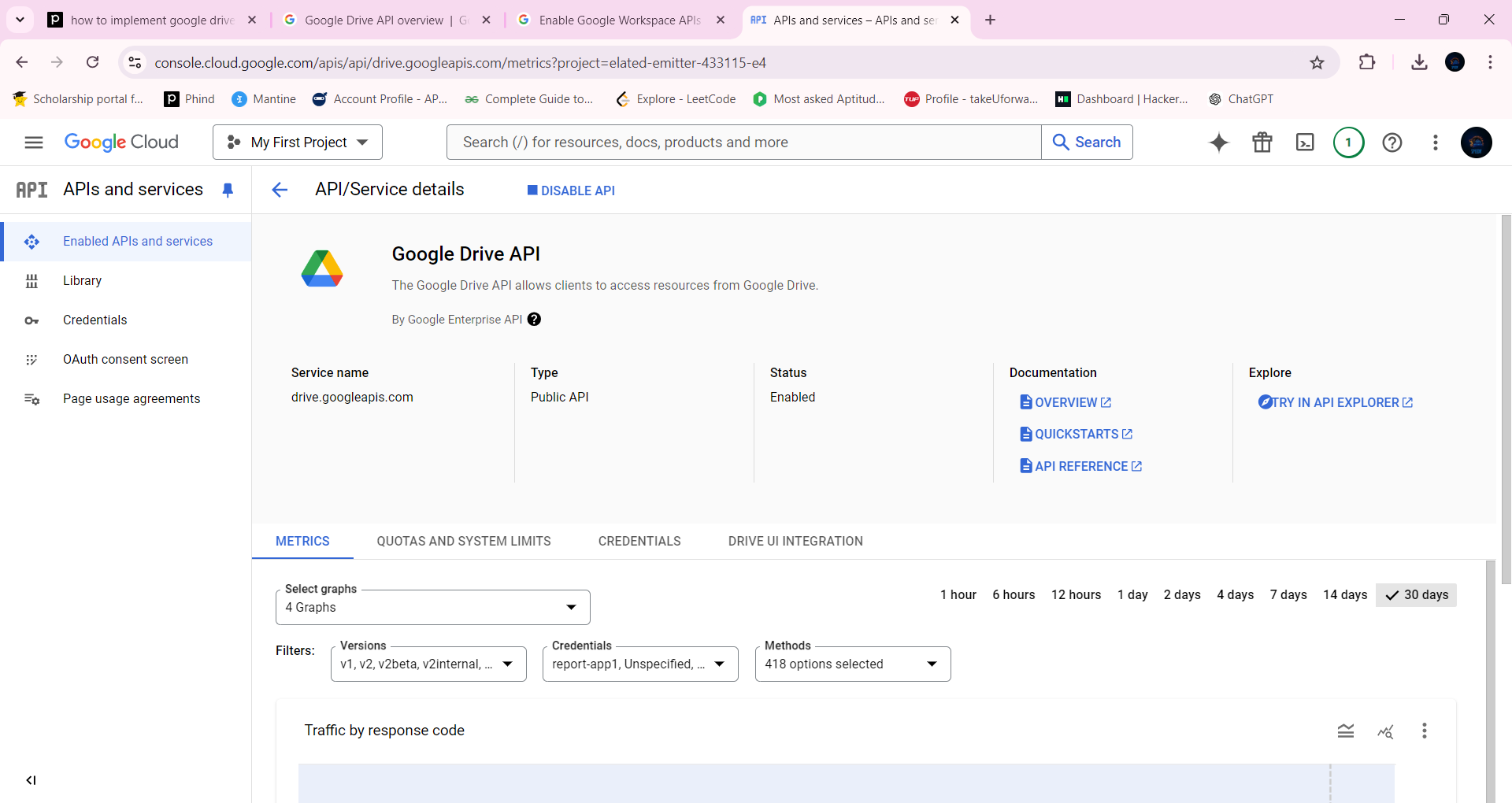
To implement Google Drive API for storing images from a web application and organizing them into multiple folders, you'll need to follow a series of steps. This guide assumes you have basic knowledge of web development and are familiar with Google Drive API authentication and setup. If not, please refer to the [Google Drive API documentation](https://developers.google.com/drive/api/v3/about-sdk) for initial setup instructions.

### Step 1: Set Up Google Drive API

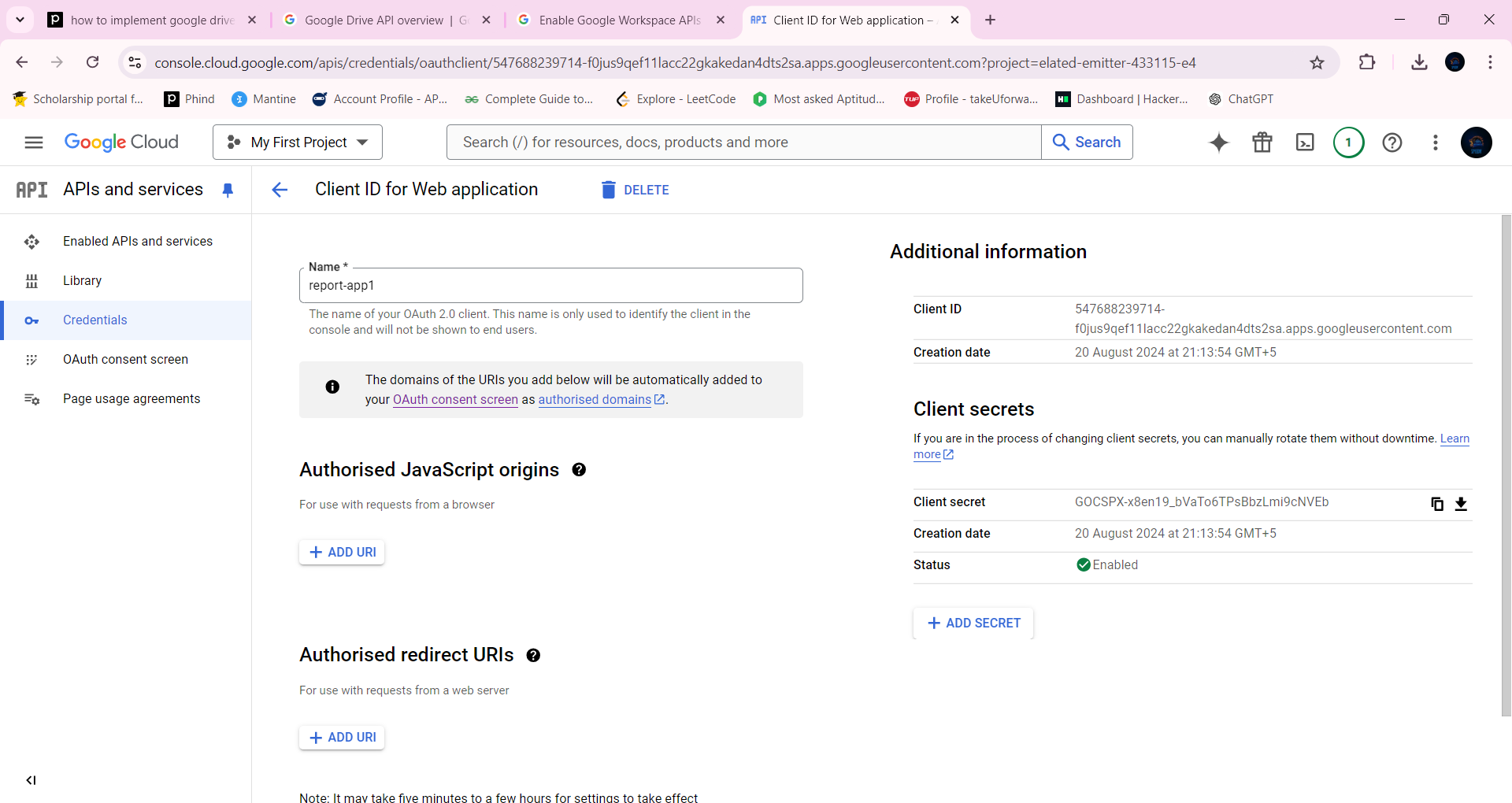
First, ensure you have enabled the Google Drive API for your project in the Google Cloud Console and have obtained the necessary credentials (OAuth 2.0 Client IDs).



Eg: 547688239714-f0jus9qef11lacc22gkakedan4dts2sa.apps.googleusercontent.com

### Step 2: Authenticate and Obtain Access Token

Use OAuth 2.0 to authenticate users and obtain an access token. This token will be used to authorize requests to the Google Drive API.



### Step 3: Create Folders

Before uploading images, you'll want to organize them into folders. Here's how you can create a folder programmatically:

```javascript

function createFolder(folderName) {

var fileMetadata = {

'name': folderName,

'mimeType': 'application/vnd.google-apps.folder'

};

drive.files.create({

resource: fileMetadata,

fields: 'id'

}, function(err, file) {

if (err) {

// Handle error

console.error(err);

} else {

console.log('Folder Id: ', file.id);

}

});

}

```

### Step 4: Upload Images to Specific Folders

To upload images to a specific folder, you need to specify the `parents` field in the file metadata with the ID of the target folder.

```javascript

function uploadImage(folderId, fileName, mimeType, content) {

var fileMetadata = {

'name': fileName,

'parents': [folderId]

};

var media = {

mimeType: mimeType,

body: content

};

drive.files.create({

resource: fileMetadata,

media: media,

fields: 'id'

}, function(err, file) {

if (err) {

// Handle error

console.error(err);

} else {

console.log('File Id: ', file.id);

}

});

}

```

In this example, `folderId` is the ID of the folder you want to upload the image to, `fileName` is the name you want to give to the uploaded image, `mimeType` should be set according to the image type (e.g., `'image/jpeg'`, `'image/png'`), and `content` is the binary content of the image.

### Step 5: Handling Multiple Folders

If you need to store images in multiple folders, simply call `uploadImage` with different `folderId`s as needed. Ensure each folder exists before attempting to upload files to them. You can create folders dynamically based on your application's logic.

### Summary

This guide provides a basic framework for integrating Google Drive API into a web application to store images in organized folders. Remember to handle errors appropriately and secure your application by validating user inputs and managing access tokens securely. For more advanced features, such as sharing files or setting permissions, refer to the [Google Drive API documentation](https://developers.google.com/drive/api/v3/reference/files).

Citations:

[1] https://developers.google.com/drive/api/guides/folder

[2] https://stackoverflow.com/questions/78180266/multiple-folders-are-created-while-using-google-drive-api

[3] https://developers.google.com/drive/api/guides/about-files

[4] https://stackoverflow.com/questions/28250336/how-to-send-multiple-images-those-are-present-in-folder-to-the-google-drive-in-a

[5] https://www.bannerbear.com/blog/the-ultimate-guide-to-using-google-drive-for-image-and-video-hosting/

[6] https://support.google.com/sites/thread/253003338/images-from-google-drive-in-embedded-html-no-longer-working?hl=en

[7] https://codelabs.developers.google.com/codelabs/gsuite-apis-intro

[8] https://www.youtube.com/watch?v=1y0-IfRW114

[9] https://www.youtube.com/watch?v=gErpJee99Fs