Google Calendar Sync Feature

**Requirement**:

Recruiter should be able to sync their work calendar initially with Galaxy so they don't have to add each individual event to their calendar.

Investigate and provide documentation around proposed solution.

**Proposed Solution:**The below are the steps we need to do to retrieve the refresh token from Google API & use it to sync with Recruiter’s Google Calendar.

**Generate Authorization Code:**

We can use the react module **“react-google-login”** in order to generate the authorization code from front end.

Below are the steps to get the authorization code.

1. Install the module

npm install react-google-login

2. Import the module in the component & use the shortcode to implement the module which will show you an option to login & will return the “code” as response

import { GoogleLogin } from 'react-google-login';  
  
render(

<GoogleLogin clientId="990776390180-nc395n9bsfffo8cslpq6kojgl8acd1ap.apps.googleusercontent.com"

buttonText="Login"

onSuccess={responseGoogle}

onFailure={responseGoogle}

responseType = 'code'

scope = 'https://www.googleapis.com/auth/calendar'

/>

);

const responseGoogle = (response) => {  
 console.log(response); // Response will be the code which we need to pass to the server to get refresh token  
}

Sample Generated Code :‘4/AACV70C9-WUezZXwLa3a24PG8d-bmq-onPlRK6m1CEdlSD0Zsv0E5BG\_A8qC3QGaCPgoWsWhTZzFLke2JxUkzcA’

**Send Authorization Code to API:**

1. Create an API to handle Authorization code
2. Create a Lambda function to receive the Authorization code from API & generate Refresh Token
3. Save the generated Refresh Token to S3 with an Unique Identifier for the recruiter in an encrypted fashion.

Sample code to generate Refresh Token using Node.js & Google API:  
const oauth2Client = new OAuth2Client(CLIENT\_ID, CLIENT\_SECRET, REDIRECT\_URL); oauth2Client.getToken('4/AACV70C9-WUezZXwLa3a24PG8d-bmq-onPlRK6m1CEdlSD0Zsv0E5BG\_A8qC3QGaCPgoWsWhTZzFLke2JxUkzcA', (err, tokens) => {

if (err) {

console.log("mahesh",err);

}

console.log(tokens); // Generated refresh token

});

Sample Generated refresh token: ‘1/Do22UX2-lg10pgKc5fmkfgRE9wH-JxZ8kkBbAxE05As’

// Set credentials for the user  
oauth2Client.setCredentials({

refresh\_token: '1/Do22UX2-lg10pgKc5fmkfgRE9wH-JxZ8kkBbAxE05As'

// Optional, provide an expiry\_date (milliseconds since the Unix Epoch)

// expiry\_date: (new Date()).getTime() + (1000 \* 60 \* 60 \* 24 \* 7)

});

// generate Access Token using Refresh token

oauth2Client.refreshAccessToken(function(err, tokens) {

// your access\_token is now refreshed and stored in oauth2Client

// store these new tokens in a safe place (e.g. database)

console.log(tokens);  
});

**Conclusion:**

Having the refresh token saved in the server, using that we can generate access token whenever we want to sync the recruiter’s calendar with new events, etc.

**Estimation:**

Development : 16 hrs

QA : 4 hrs

Total : 20 hrs

**References:**

<https://github.com/anthonyjgrove/react-google-login>

<https://github.com/google/google-api-nodejs-client/#manually-refreshing-access-token>

<https://github.com/google/google-api-nodejs-client/blob/master/samples/oauth2.js>