Google Calendar Integration

Google Calendar API
-https://developers.google.com/calendar
-https://developers.google.com/calendar/api

node.js seems to best option between the supported languages

Install/Setup (Requires npm)
npm install googleapis@39 --save

The following lines will create the calendar constant:

```
const {google} = require('googleapis');
```

```
const calendar = google.calendar('v3');
```

Sample Setup Code (May not be 100% applicable to ARORA, but is a basic example)

```
const fs = require('fs');
const readline = require('readline');
const {google} = require('googleapis');

// If modifying these scopes, delete token.json.
const SCOPES = ['https://www.googleapis.com/auth/calendar.readonly'];
// The file token.json stores the user's access and refresh tokens, and is
// created automatically when the authorization flow completes for the first
// time.
const TOKEN_PATH = 'token.json';

// Load client secrets from a local file.
fs.readFile('credentials.json', (err, content) => {
  if (err) return console.log('Error loading client secret file:', err);
  // Authorize a client with credentials, then call the Google Calendar API.
  authorize(JSON.parse(content), listEvents);
});

/**
 * Create an OAuth2 client with the given credentials, and then execute the
 * given callback function.
 * @param {Object} credentials The authorization client credentials.
 * @param {function} callback The callback to call with the authorized client.
 */
```

```
function authorize(credentials, callback) {
  const {client_secret, client_id, redirect_uris} = credentials.installed;
  const oAuth2Client = new google.auth.OAuth2(
      client_id, client_secret, redirect_uris[0]);

  // Check if we have previously stored a token.
  fs.readFile(TOKEN_PATH, (err, token) => {
    if (err) return getAccessToken(oAuth2Client, callback);
    oAuth2Client.setCredentials(JSON.parse(token));
    callback(oAuth2Client);
  });
}

/**
 * Get and store new token after prompting for user authorization, and then
 * execute the given callback with the authorized OAuth2 client.
 * @param {google.auth.OAuth2} oAuth2Client The OAuth2 client to get token for.
 * @param {getEventsCallback} callback The callback for the authorized client.
 */
function getAccessToken(oAuth2Client, callback) {
  const authUrl = oAuth2Client.generateAuthUrl({
    access_type: 'offline',
    scope: SCOPES,
  });
  console.log('Authorize this app by visiting this url:', authUrl);
  const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout,
  });
  rl.question('Enter the code from that page here: ', (code) => {
    rl.close();
    oAuth2Client.getToken(code, (err, token) => {
      if (err) return console.error('Error retrieving access token', err);
      oAuth2Client.setCredentials(token);
      // Store the token to disk for later program executions
      fs.writeFile(TOKEN_PATH, JSON.stringify(token), (err) => {
        if (err) return console.error(err);
        console.log('Token stored to', TOKEN_PATH);
      });
      callback(oAuth2Client);
    });
  });
}
```

```
/**
 * Lists the next 10 events on the user's primary calendar.
 * @param {google.auth.OAuth2} auth An authorized OAuth2 client.
 */
function listEvents(auth) {
  const calendar = google.calendar({version: 'v3', auth});
  calendar.events.list({
    calendarId: 'primary',
    timeMin: (new Date()).toISOString(),
    maxResults: 10,
    singleEvents: true,
    orderBy: 'startTime',
  }, (err, res) => {
    if (err) return console.log('The API returned an error: ' + err);
    const events = res.data.items;
    if (events.length) {
      console.log('Upcoming 10 events:');
      events.map((event, i) => {
        const start = event.start.dateTime || event.start.date;
        console.log(`${start} - ${event.summary}`);
      });
    } else {
      console.log('No upcoming events found.');
    }
  });
}
```