# 08c [Individual/Pairs/Group] Auth Integration

## Configure Google OAuth 2.0 with Node.js

The following steps describe implementing Google OAuth 2.0 in a Node.js application.



### **Step 1** Configure Google Developers Console Project

1. Go to the Google Developers Console at https://console.developers.google.com/
2. Create a new project or select an existing project
3. Complete step 1b
4. Complete step 1c

#### **Step 1b** Create OAuth Consent Screen

1. Go to the `OAuth Consent Screen` tab
2. Select `internal` to configure the application to be available to users within an organisation and `external` to configure the application to be available to users outside an organisation.
3. Click `Create`
4. Add the following details: App Name; User Support Email; Developer Contact Info Email;
5. Click `Save and Continue`
6. Next, you will be able to add scopes, you can just click `Save and Continue` at this point.
7. Next, add the email you want to test.
8. Click `Save and Continue`

#### **Step 1c** Create Credentials

1. Go to the `Credentials` tab
2. Click `Create credentials` and select `OAuth client ID`
3. Select `Web application`

4. Add `http://localhost:3000` to the Authorized JavaScript origins
5. Add `http://localhost:3000/auth/google/callback` to the Authorized redirect URIs
6. Click `Create`
7. Copy the Client ID and Client Secret for later.

## Step 2 Create Node Application

The following describes how to configure the Node Application to support Google OAuth 2.0.

### Step 2a Setup Node Application

1. Create a new or use an existing application.
2. Install googleapis npm package: `npm install googleapis`; used to contact Google's API.
3. Install express npm package: `npm install express`; used to create a simple HTTP server.
4. Install dotenv npm package: `npm install dotenv`; used to handle environment variables.
5. Create a `.gitignore` and add the following just to be sure you don't push the credentials of the environment file at some point:
   node_modules
   .env
   package-lock.json
6. Create a `.env` file with the following keys and the credentials you created in step 1c:
   GOOGLE_CLIENT_ID=your_id
   GOOGLE_CLIENT_SECRET=your_secret
   GOOGLE_REDIRECT_URI=http://localhost:3000/auth/google/callback
7. Open package.json and add the following under scripts:
   "start": "node index.js"

### Step 2b Create Index.js

1. Create an `index.js` file
2. Add `require('dotenv').config();` to read the environment variables in `.env`.
3. Add `const { google } = require('googleapis');` to load googleapis npm package.
4. Add `const express = require('express');` to load express npm package.
5. Add `const app = express();` to create a new express app.
6. Add `const port = 3000;` to define the server's port.
7. Add `app.use(express.static('public'));` to define the directory for public assets.
8. Add `app.use(express.json());` to tell express to parse JSON.
9. Add the following to create an Oauth client object that can interact with Google API:
   ```
   const oauth2Client = new
   google.auth.OAuth2(process.env.GOOGLE_CLIENT_ID,
   process.env.GOOGLE_CLIENT_SECRET, process.env.GOOGLE_REDIRECT_URI);
   ```

10. Add the following to create an endpoint redirecting to Google's consent page with an object providing objects such as the scope you want to get access to. The example uses the scope ``https://www.googleapis.com/auth/userinfo.profile`` to get the name and picture of the authenticated user:

```
app.get('/auth/google/consent', (req, res) => {
      const url = oauth2Client.generateAuthUrl({
            access_type: 'offline',
            scope: ['https://www.googleapis.com/auth/userinfo.profile']
      });
      res.redirect(url)
})
```

11. Add the following to create an endpoint Google uses to redirect with a code after the user has accepted the consent page. The code is used to obtain an access and refresh token, which in the following example is sent back to the user's client:

```
app.get('/auth/google/callback', async (req, res) => {
      try {
        const { code } = req.query;
        const { tokens } = await oauth2Client.getToken(code);
        res.redirect(`/index.html?tokens=${JSON.stringify(tokens)}`);
      } catch (error) {
        console.error('Error', error);
        res.send('Error');
      }
});
```

12. Add the following to create a where the user who just obtained the tokens can send their tokens to get information about who they are:

```
app.get('/auth/whoami', async (req, res) => {
      try {
        const tokens = JSON.parse(req.query.tokens);
        oauth2Client.setCredentials(tokens);
        const userInfo = await google.oauth2('v2').userinfo.get({
            auth: oauth2Client
        });
        res.send(userInfo.data);
      } catch (error) {
        console.error('Error', error);
        res.send('Error');
      }
});
```

13. Add the following to start the server on the defined port.

```
app.listen(port, () => {console.log(`Server running on port
${port}`);});
```

## Step 2c Create Index.html

1. Create a new directory called `public`.
2. Create a new file in that directory called `index.html`.
3. Add the usual HTML skeleton:
   <!DOCTYPE html>

```
<html lang="en">
<head>
<meta charset="UTF-8" />
</head>
<body>
</body>
</html>
```

4. Add some style to the header:

```
<style>
      body {
         font-family: Arial, Helvetica, sans-serif;
         margin: 0;
      }
      h1 {
         margin: 0;
      }
      .btn {
         display: inline-block;
         padding: 10px 20px;
         background-color: #007bff;
         color: #fff;
         text-decoration: none;
         border-radius: 5px;
      }
      .header {
         padding: 20px;
         background-color: #f4f4f4;
         display: flex;
         align-items: center;
         justify-content: space-between;
      }
      .main {
         padding: 20px;
      }
      .profile-img {
         width: 40px;
         height: 40px;
         border-radius: 50%;
         border: 1px solid #333;
         box-shadow: 0 0 5px rgba(0, 0, 0, 0.1);
      }
   </style>
```

5. Give the body an id: <body id="body">; for easy reference.
6. Add a header with a link to the endpoint redirecting to the consent page:

```
<header class="header">
        <a href="/auth/google/consent" class="btn">Login with Google</a>
</header>
```

7. Add a title and paragraph telling the user about their current state:

```
<div class="main">
  <div id="user">
    <h1>Welcome Guest</h1>
    <p> Please login to continue.</p>
  </div>
</div>
```

8. Add a script element and the following three lines to search the URL for a parameter called `token`:
```
<script>
const queryString = window.location.search;
const urlParams = new URLSearchParams(queryString);
const tokens = urlParams.get('tokens');
</script>
```
9. Add the following lines after `const tokens = …` to create a fetch request to the endpoint returning information about the authenticated user, and insert the name and picture of the user on success:
```
if (tokens) {
  (async () => {
    const response = await fetch(`/auth/whoami?tokens=${tokens}`, {
      method: 'GET',
    });
    const data = await response.json();
      if (data.error) {
        console.error(data.error);
        return;
      }
      document.getElementById('body').innerHTML = `
                <header class="header">
                    <img class="profile-img" src="${data.picture}" alt="${data.name}" />
                    <a href="/" class="btn">Logout</a>
                </header>
                <div class="main">
                    <div id="user">
                        <h1>Welcome ${data.name}</h1>
                    </div>
                </div>
      `;
  })();
}
```

## **Step 3** Start Server and Test

1. Execute `npm start` in a terminal.
2. Go to http://localhost:3000/index.html
3. Click `Login with Google`.
4. Select Account
5. Accept Consent
6. See your name and profile on your site.