OAuth

Gold - Chapter 7 - Topic 3

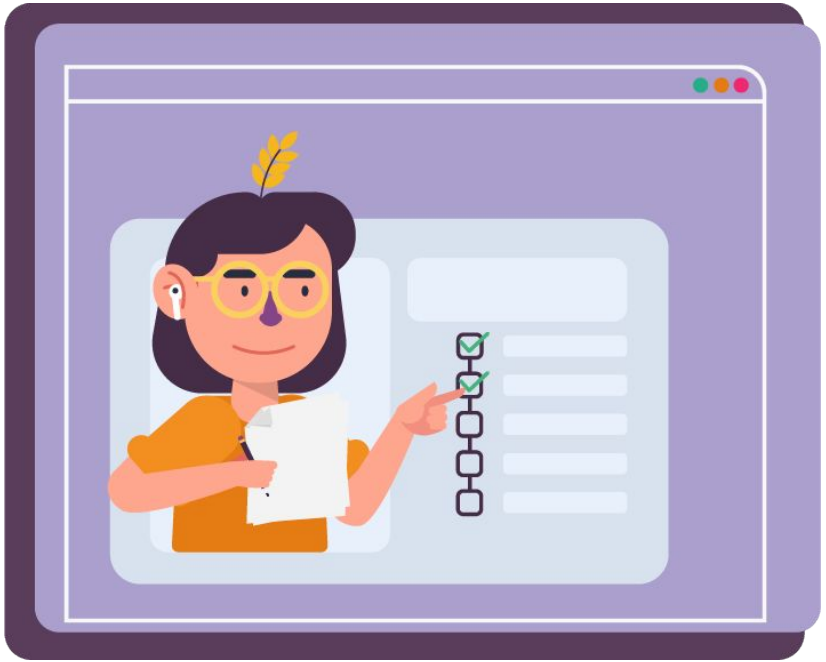
Selamat datang di Chapter 7 Topic 3 online course Fullstackweb Developer dari Binar Academy!

Siang-siang makan sop iga, selamat datang di topik 3 🌞

Seperti yang sudah dijelaskan sebelumnya, chapter 6 ini bakalan ngajak kamu menggunakan sistem authentication dari backend dan melakukan testing. **Pada topik sebelumnya kita sudah mempelajari implementasi autentikasi. Sekarang kita akan mempelajari tentang Oauth.**

Detailnya, kita bakal bahas hal-hal berikut ini:

- Konsep OAuth
- Google/Facebook OAuth



BINAR

Kamu pernah denger OAuth? Itu loh, jembatan antar aplikasi untuk mengakses data. Untuk lebih jelasnya mari simak lebih lanjut. Letsgooo~

Biar konsep OAuth lebih gampang dipahami coba bayangin analogi di bawah ini ya~

Bayangin kamu adalah tamu hotel yang ingin menginap dan menggunakan fasilitas kamar hotel. Sebelum mendapatkan kamar kamu harus memberitahukan identitasmu ke resepsionis terlebih dahulu. Kemudian kamu baru mendapatkan kunci kamar hotel agar dapat mengakses fasilitas kamar hotel tersebut.

Nah, kaitannya sama konsep OAuth, kunci kamar hotel tersebut adalah OAuth, resepsionis adalah pemilik resource, dan kamar sebagai data-nya.

Kalau secara definisi, OAuth itu apa ya?

OAuth adalah protokol otorisasi standar terbuka yang memberikan aplikasi kemampuan untuk **“akses yang ditentukan secara aman”**.

Secara sederhana, OAuth menjembatani antar aplikasi dalam mengakses data yang data yang dibutuhkan tanpa menyebarkan data yang penting seperti password.

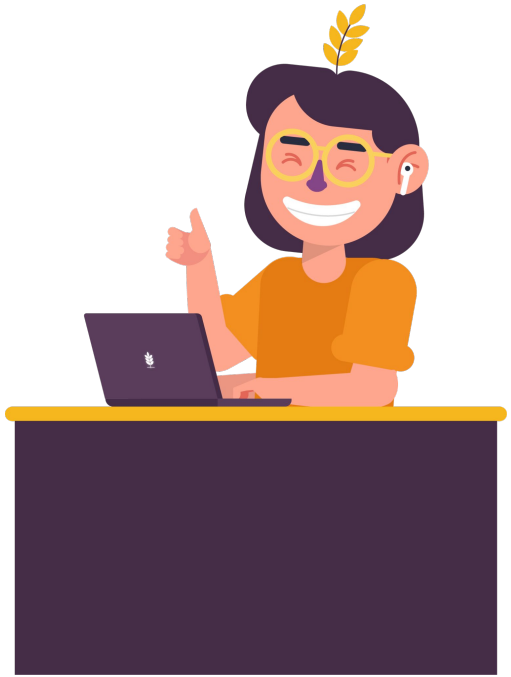
Contohnya seperti ini, kalian pasti pernah melakukannya. Kita ingin memakai aplikasi X yang bisa mendeteksi 9 foto Instagram kita dengan like terbanyak, nah saat login di aplikasi X itu kita mengijinkan aplikasi X dan Instagram untuk memakai data yang diperlukan (foto, jumlah like), tetapi tidak dengan password, dll.

Kalau seandainya Aplikasi X itu diretas, maka password kita tetap aman bersama Instagram.

OAuth tidak membagikan data password kita, tetapi menggunakan token autorisasi sebagai bukti identitas antara konsumen dan penyedia servis.

Sampai dengan hari ini terdapat dua versi OAuth, yaitu OAuth 1.0 dan OAuth 2.0. OAuth 2.0 ada versi terbaru dari yang 1.0, tetapi mereka tidak saling kompatibel, dan tidak bisa di rollback jika sudah menggunakan salah satunya.

OAuth 2.0 lebih simpel dan cepat untuk diimplementasi, hampir semuanya sekarang menggunakan OAuth versi 2.0.



Terus, gimana sih alur kerja OAuth?

1. Aplikasi meminta autorisasi dari user.

2. User mengautorisasi dan mengirimkan bukti.

3. Aplikasi menunjukkan bukti autorisasi ke server untuk mendapatkan token.

4. Token dibatasi hanya untuk mengakses apa yang diizinkan pengguna untuk aplikasi tertentu.

Nih, komponen-komponen utama yang ngebentuk sebuah OAuth :

1. Scopes and Consent
2. Actors
3. Client
4. Token
5. Authorization Server
6. Flows

Biar lebih jelas, kita bahas satu-satu kuy~

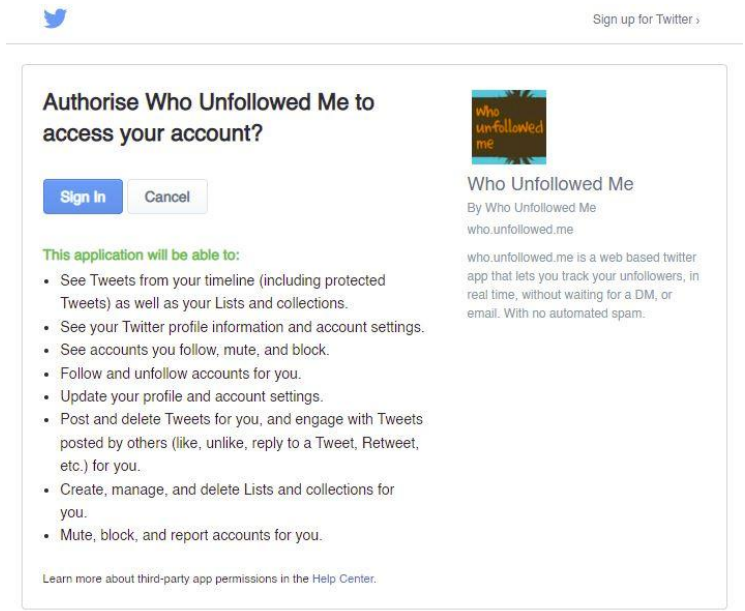
1. Scopes and Consent

Scopes adalah **apa yang tertera di layar saat aplikasi meminta izin**. Berisi tentang kumpulan izin yang diminta oleh klien saat meminta token.

Foto di samping adalah contoh Scope dan Consent.

Aplikasi tersebut akan bisa melihat tweet kita, melihat informasi profile dan setting, dan sebagainya. Biasanya tertera dibawah tulisan hijau.

Sedangkan ada scope yang tidak bisa diakses atau diizinkan juga harus ditulis juga, seperti: Aplikasi ini tidak bisa melihat password anda, dll. Biasanya ditulis dibawah tulisan warna Merah.

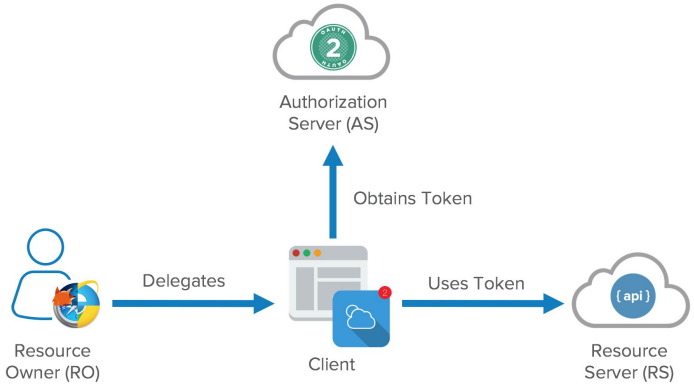


2. Actors

Terdapat empat aktor dalam alur OAuth, antara lain:

- Resource Owner
  Adalah pemilik dari Resource Server.
- Resource Server
  API dimana tempat menyimpan data yang akan diakses
- Client
  Aplikasi yang ingin mengakses data tersebut.
- Authorization Server
  Mesin dari OAuth.

Resource Owner adalah peranan yang dapat berganti-ganti credentials. Bisa sebagai End User atau sebuah perusahaan.



3. Client

Terdapat dua Client, yaitu publik dan credential (rahasia).

- **Public Client**

  Misalnya browser, mobile app atau device lainnya.

- **Client Credential**

  Klien yang dipercaya dapat menyimpan secret. Mereka tidak berjalan di desktop atau aplikasi, dan tidak bisa di reverse-engineer untuk mendapatkan secret key. Mereka bekerja di area yang aman dimana pengguna tidak dapat mengaksesnya.

4. Tokens

Berdasarkan umur pemakaiannya, token dapat dapat dibagi menjadi:

- **Access Token**

  Token ini cepat kadaluarsa dalam hitungan menit atau jam. Karena token ini berumur pendek, mereka tidak bisa dicabut atau ditarik kembali. Token ini dipakai client untuk mengakses Resource Server (API). Tidak diperlukan menjadi client rahasia untuk mendapatkan token, cukup dengan menjadi public client.

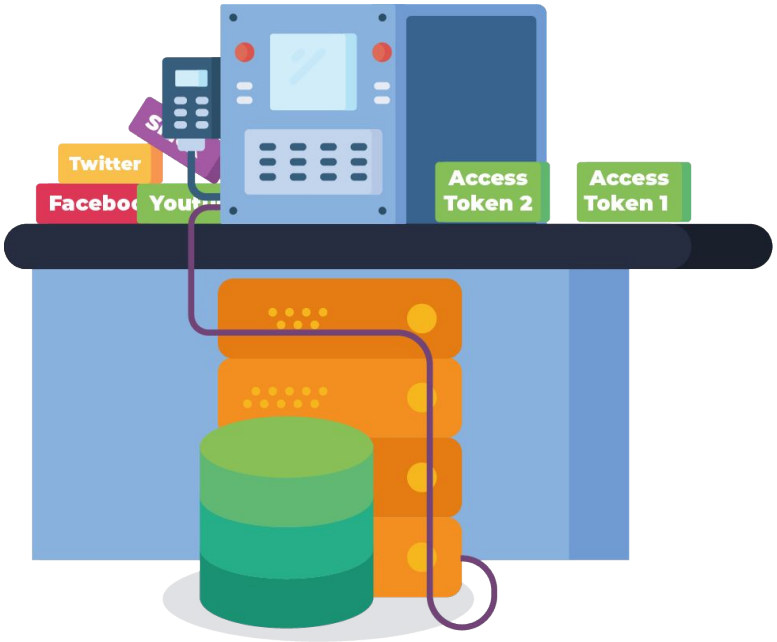
- **Refresh Token**

  Token ini berumur lebih panjang, bisa hitungan hari sampai bulan bahkan tahun. Token ini bisa dicabut, untuk mendapatkan Token ini harus dibutuhkan klien rahasia dengan autentikasi.

5. Authorization Server

Authorization server merupakan mesin dari OAuth. Server ini bertanggung jawab untuk mengautentikasi pengguna dan mengeluarkan access token yang berisi data pengguna dan kebijakan akses yang tepat.

Authorization server bisa digunakan untuk mengamankan API yang kamu miliki dan memberikan otorisasi user untuk mengakses layanan web kamu .



6. Flows

1. **Implicit Flow**

   Flow pertama adalah Implicit Flow, dinamakan Implisit karena semua komunikasi terjadi didalam browser.

2. **Authorization Code Flow**

   Atau disebut juga Legged 3, flow ini menggunakan dua channel, yaitu depan dan belakang. Channel depan digunakan untuk mendapatkan kode autorisasi. Sedangkan channel belakang digunakan oleh aplikasi klien untuk menukar kode autorisasi dengan token akses.

3. **Client Credential Flow**

   Flow ini digunakan untuk skenario flow server ke server. Disini aplikasi klien adalah aplikasi rahasia dan bekerja untuk dia sendiri, bukan untuk kepentingan user.

4. **Resource Owner Password Flow**

   Flow ini adalah flow model lama. Sangat mirip dengan autentikasi langsung dengan username dan password, dan ini tidak direkomendasikan.

5. **Assertion Flow**

   Ini adalah flow yang paling baru, mirip dengan Flow nomor 3. Flow ini memungkinkan Authorization Server untuk mempercayai pemberian otorisasi dari pihak ketiga, seperti SAML IdP.

6. **Device Flow**

   Flow ini tidak menggunakan web browser, hanya menggunakan Controller seperti TV. Kode user dikembalikan dari permintaan otorisasi yang harus ditukar dengan mengklik URL di device dengan browser untuk di autorisasi.

Tadi kita udah bahas tentang konsep Oauth. Sekarang kita akan pelajari salah satu Oauth yang banyak digunakan, yaitu **Google/Facebook Oauth**.

Konsep Google/Facebook OAuth

Sama seperti OAuth 2.0, Google/Facebook OAuth adalah **protokol otorisasi yang memakai Google Client API Libraries** atau Google OAuth sebagai endpoint untuk implementasi otorisasi OAuth 2.0 untuk mengakses Google API's.

Implementasi Google OAuth

Buka Terminal atau Command-Prompt atau GitBash di komputer kalian masing-masing lalu buka direktori projek kalian, kita akan menambah package baru.

Ketikkan syntax ini untuk menambah package react-google-login

```
npm install react-google-login
```

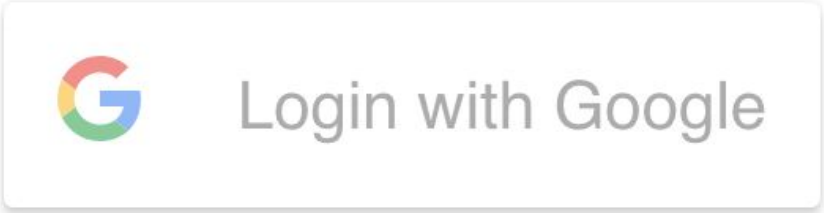
Kemudian import Component GoogleLogin di projek kalian. Ada beberapa Component, yaitu:

- clientID
- buttonText
- onSuccess
- onFailure
- cookiePolicy

Untuk onSuccess dan onFailure akan diberi callback function agar bisa dilihat response-nya di console dengan syntax di samping ini.

```
const responseGoogle = response => {
  console.log(response);
};
```

Komponen GoogleLogin akan terlihat seperti ini. Masih dalam bentuk *disable* karena belum diberi Client ID.



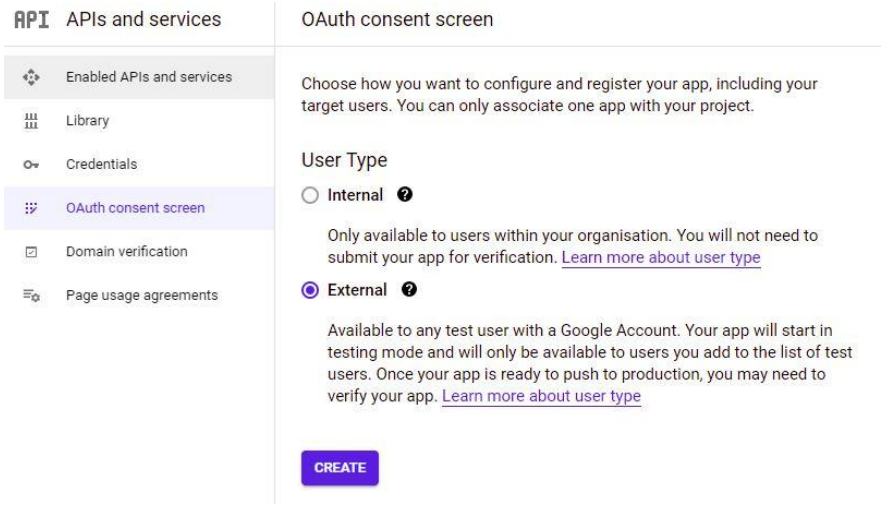
Langka pertama yaitu, buka link dibawah ini:

https://console.cloud.google.com/apis/dashboard

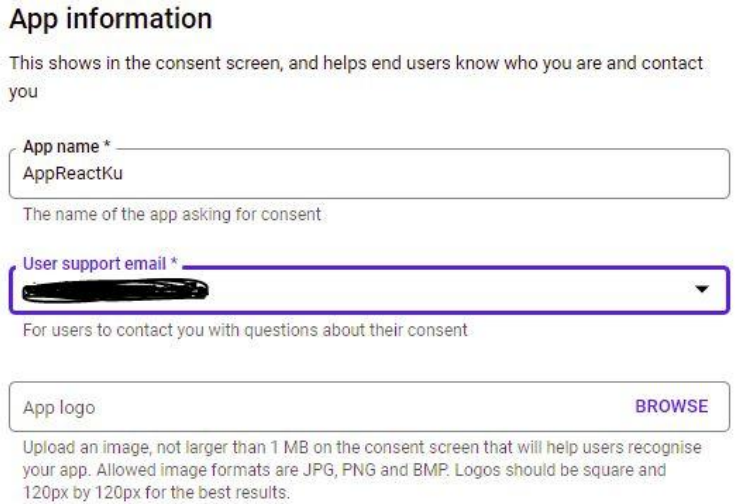
Lalu klik Create New Project, isikan nama project kalian masing-masing lalu klik Create.

Kemudian setelah project selesai dibuat, klik menu OAuth consent screen di sidebar sebelah kiri.

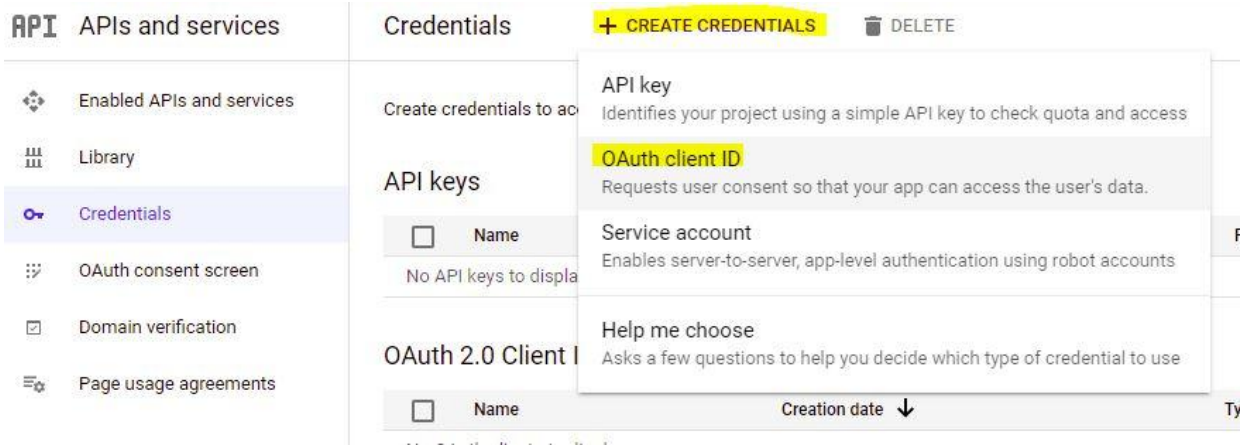
Lalu pilih opsi External, dan klik Create.



Setelah itu beri nama Aplikasi yang akan memakai Google Oauth ini. Isi juga input email-nya dan App Logo itu opsional. Kemudian klik Save and Continue, Scope dan User tidak perlu diisi hanya optional juga.



Setelah selesai, klik menu Credentials yang ada di sidebar kiri. Lalu klik Create Credentials yang ada diatas kemudian pilih OAuth client ID.

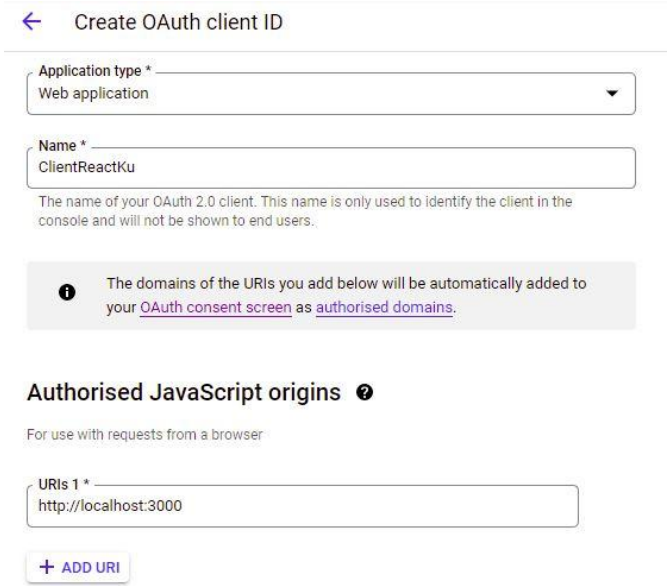


Setelah klik OAuth client ID, lalu muncul form seperti gambar disamping ini. Pilih Web Application dari menu Application Type karena kita akan memakainya di web browser.

Lalu beri nama kemudian kita isi input URIs dengan alamat project kita, sebagai contoh: http://localhost:3000

Sesuaikan dengan project atau kebutuhan masing-masing ya.

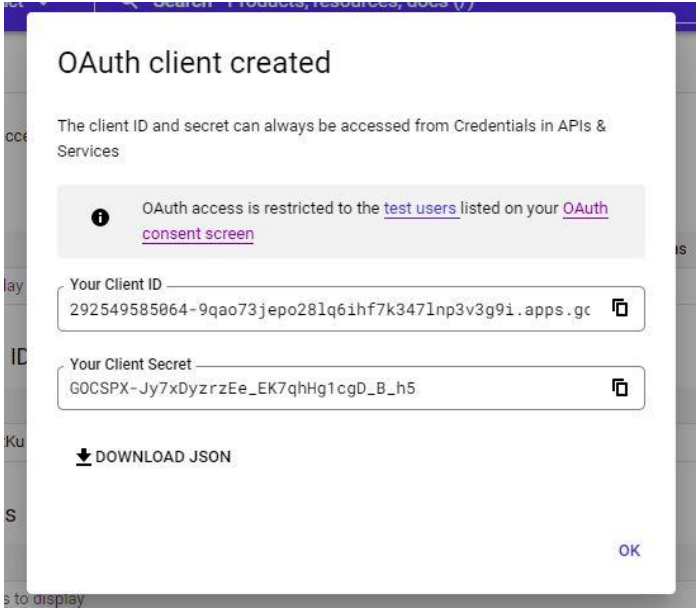
Kemudian klik Create dan modal akan muncul.



OAuth client telah terbentuk! Yeaay~

Modal ini berisi Client ID beserta Client Server, bisa dicopas maupun didownload dalam bentuk JSON.

Untuk kali ini Copy Client ID kita masing-masing.



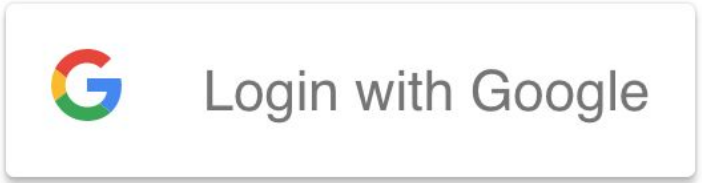
Implementasi Google OAuth

Kembali ke project di text editor masing-masing.

Client ID yang sudah kita copy tadi lalu dimasukkan di Component Google Login didalam tanda petik.

Lalu button Login with Google akan ter-enable. Jangan lupa setelah login dengan Google, cek console kita tentang response yang didapat didalam fungsi onSuccess yang akan berisi access token, dll.

```
<GoogleLogin
  clientId="292549585064-
9qao73jepo28lq6ihf7k347lnp3v3g9i.apps.googleus
ercontent.com"
  buttonText="Login with Google"
  onSuccess={responseGoogle}
  onFailure={responseGoogle}
  cookiePolicy="single_host_origin"
/>
```



Untuk mengimplementasikan Google OAuth di backend, sebenernya banyak approach, namun karena kita udah punya FE, maka dari itu kita pakai backend kita sebagai **Google API Client**. Maka dari itu backend kita akan membutuhkan **Google OAuth Access Token**, nah si token ini akan dikirim oleh FE ke BE ketika user selesai melakukan Login atau Register melalui Google.

Setelah BE menerima request dari FE yang berisi **Google OAuth Access Token**, BE akan melakukan request ke Google API untuk mencari siapa pemilik token tersebut, dan ketika sudah mendapatkan pemilik token tersebut, BE akan mencari user yang relevan terhadap informasi pemilik token tersebut.

Request handler-nya kurang lebih akan seperti contoh di-samping. Sisa implementasinya dapat kalian lihat pada repository berikut.

```
async function handleGoogleLoginOrRegister(req, res) {
  const { accessToken } = req.body;
  const options = { headers: { Authorization: `Bearer ${accessToken}` } }
};

  try {
    const response = await axios.get(
      "https://www.googleapis.com/oauth2/v2/userinfo",
      options
    );
    const { id, email, name } = response.data;

    let user = await User.findOne({ where: { googleId: id } });
    if (!user) user = await User.create({ email, name });

    const accessToken = createToken(user);

    res.status(201).json({ accessToken });
  } catch (err) {
    res.status(401).json({
      error: {
        name: err.name,
        message: err.message,
      }
    });
  }
}
```

Yeay kamu sudah berhasil memahami Oauth! Pada topik selanjutnya kita akan membahas tentang Unit testing & TDD (Test Driven Development). Semangat terus yaa 🎊

Terima Kasih!

Next Topic

loading...

BINAR