📆 Hari 1 (Senin) – JWT Auth: Login & Register

🎯 Tujuan Pembelajaran

- Memahami konsep JWT (JSON Web Token) dan cara kerjanya dalam autentikasi.
- Mampu mengimplementasikan alur login dan register menggunakan JWT.
- Menguasai penggunaan library bcrypt untuk hashing password demi keamanan.

🧠 Materi Inti (2 Jam)

1. Konsep JWT dan Cara Kerjanya

- **Apa itu JWT?** JWT adalah standar terbuka (RFC 7519) yang mendefinisikan cara aman untuk mentransmisikan informasi antar pihak sebagai objek JSON.
- **Struktur JWT:** Terdiri dari tiga bagian yang dipisahkan oleh titik (`.`): Header, Payload, dan Signature.
  - **Header:** Berisi tipe token (JWT) dan algoritma hashing (misalnya, HS256, RS256).
  - **Payload:** Berisi klaim (claims), yaitu entitas (biasanya pengguna) dan data tambahan. Ada tiga jenis klaim: Registered, Public, dan Private.
  - **Signature:** Dibuat dengan menggabungkan Header, Payload, dan secret key menggunakan algoritma yang ditentukan di Header. Ini digunakan untuk memverifikasi bahwa pengirim JWT adalah siapa yang diklaimnya dan memastikan bahwa pesan tidak diubah di tengah jalan.
- **Alur Autentikasi dengan JWT:**
  1. Pengguna mengirimkan kredensial (username/email dan password) ke server.
  2. Server memverifikasi kredensial. Jika valid, server membuat JWT.
  3. Server mengirimkan JWT kembali ke klien (biasanya dalam header `Authorization` sebagai `Bearer Token`).
  4. Klien menyimpan JWT (misalnya, di Local Storage atau Cookies).
  5. Untuk mengakses resource yang dilindungi, klien mengirimkan JWT di setiap request ke server.
  6. Server memverifikasi signature JWT. Jika valid, server mengizinkan akses ke resource.

2. Implementasi Login dan Register dengan JWT

- **Register:**
  - Menerima data pengguna baru (username/email, password).
  - Hashing password menggunakan bcrypt.
  - Menyimpan data pengguna (termasuk password yang sudah di-hash) ke database.
  - Mengirimkan response sukses.
- **Login:**
  - Menerima kredensial pengguna (username/email, password).
  - Mencari pengguna di database berdasarkan username/email.
  - Membandingkan password yang diinput dengan password yang sudah di-hash di database menggunakan bcrypt.
  - Jika cocok, buat JWT yang berisi informasi identitas pengguna (misalnya, user ID).
  - Kirimkan JWT sebagai response.

3. Hashing Password Menggunakan Bcrypt

- **Mengapa Hashing?** Jangan pernah menyimpan password dalam bentuk plain text di database. Hashing mengubah password menjadi string acak yang tidak bisa dikembalikan ke bentuk aslinya (one-way function). Ini melindungi password pengguna jika database bocor.
- **Mengapa Bcrypt?** Bcrypt adalah algoritma hashing password yang kuat dan direkomendasikan. Ia dirancang agar lambat secara komputasi, yang membuatnya lebih tahan terhadap serangan brute-force.
- **Cara Kerja Bcrypt:** Bcrypt menggunakan salt (string acak unik) yang digabungkan dengan password sebelum di-hash. Salt ini disimpan bersama hash password di database. Saat verifikasi, salt yang tersimpan digunakan untuk me-hash password yang diinput, lalu hasilnya dibandingkan dengan hash yang tersimpan.

📝 Praktik Mandiri (8 Jam)

1. **Setup Project:** Jika belum, inisialisasi project Node.js baru atau gunakan project backend dari minggu 4. Install dependency yang dibutuhkan: `express`, `jsonwebtoken`, `bcrypt`, dan library database (`pg` atau `prisma`).

   ```
   npm install express jsonwebtoken bcrypt pg # atau prisma
   ```

2. **Buat Model Pengguna:** Definisikan struktur data untuk pengguna (misalnya, ID, username/email, password_hash).
3. **Implementasi Register Endpoint (`POST /api/register`):**
   - Terima `username` dan `password` dari body request.
   - Gunakan `bcrypt.hash()` untuk me-hash password. Pilih jumlah salt rounds yang sesuai (misalnya, 10).
   - Simpan `username` dan `password_hash` ke database.
   - Kirimkan response sukses (misalnya, status 201).
4. **Implementasi Login Endpoint (`POST /api/login`):**
   - Terima `username` dan `password` dari body request.
   - Cari pengguna di database berdasarkan `username`.
   - Jika pengguna ditemukan, gunakan `bcrypt.compare()` untuk membandingkan password yang diinput dengan `password_hash` dari database.
   - Jika cocok, buat JWT menggunakan `jsonwebtoken.sign()`. Masukkan payload (misalnya, `{ userId: user.id }`) dan secret key. Atur expiration time jika perlu.
   - Kirimkan JWT sebagai response (misalnya, `{ token: jwtToken }`).
   - Jika kredensial tidak valid, kirimkan response error (misalnya, status 401).
5. **Uji Endpoint dengan Postman/Insomnia:**
   - Uji endpoint register dengan data pengguna baru.
   - Uji endpoint login dengan kredensial yang benar dan salah. Pastikan mendapatkan JWT saat login berhasil.

💡 Tips untuk Pemula

- Gunakan `.env` file dan library seperti `dotenv` untuk menyimpan secret key JWT dan konfigurasi database. Jangan hardcode nilai-nilai sensitif!
- Pastikan untuk menangani error dengan baik, misalnya jika pengguna sudah terdaftar saat register atau kredensial salah saat login.
- Baca dokumentasi resmi `jsonwebtoken` dan `bcrypt` untuk memahami opsi dan best practice lebih lanjut.
- Mulai dengan implementasi sederhana, lalu tambahkan fitur atau validasi tambahan.

📚 Referensi

- JWT Docs
- Bcrypt Docs (npm)
- Jsonwebtoken Docs (npm)
- Express.js Routing
- Dotenv Docs (npm)