📆 Hari 2 (Selasa) – Middleware Authorization

🎯 Tujuan Pembelajaran

- Memahami peran dan cara kerja middleware dalam Express.js.
- Mampu membuat middleware kustom untuk otorisasi (memeriksa JWT).
- Mengintegrasikan middleware otorisasi dengan endpoint yang perlu dilindungi.

🧠 Materi Inti (2 Jam)

1. Konsep Middleware di Express.js

- **Apa itu Middleware?** Middleware adalah fungsi yang memiliki akses ke objek request (`req`), objek response (`res`), dan fungsi middleware berikutnya dalam siklus request-response aplikasi Express. Fungsi middleware dapat melakukan tugas-tugas seperti:
  - Mengeksekusi kode apa pun.
  - Membuat perubahan pada objek request dan response.
  - Mengakhiri siklus request-response.
  - Memanggil fungsi middleware berikutnya dalam stack.
- **Cara Kerja:** Middleware dieksekusi secara berurutan. Ketika sebuah middleware selesai, ia memanggil `next()` untuk meneruskan kontrol ke middleware berikutnya atau handler route.
- **Contoh Middleware Umum:** `express.json()` (untuk parsing body JSON), `express.static()` (untuk menyajikan file statis), middleware logging, error handling.

2. Implementasi Middleware untuk Otorisasi

- **Tujuan:** Membuat middleware yang bertugas memverifikasi apakah request memiliki JWT yang valid. Jika valid, request diteruskan ke handler route. Jika tidak valid, request dihentikan dengan response error (misalnya, 401 Unauthorized).
- **Langkah-langkah:**
  1. Ambil JWT dari header `Authorization` (format `Bearer TOKEN`).
  2. Periksa apakah token ada.
  3. Verifikasi token menggunakan `jsonwebtoken.verify()` dengan secret key yang sama saat token dibuat.
  4. Jika verifikasi berhasil, token valid. Ekstrak informasi pengguna dari payload JWT (misalnya, `userId`).
  5. Tambahkan informasi pengguna ke objek `req` (misalnya, `req.user = decodedToken;`) agar bisa diakses di handler route berikutnya.
  6. Panggil `next()` untuk melanjutkan ke handler route.
  7. Jika token tidak ada atau verifikasi gagal, kirimkan response error (misalnya, status 401 atau 403) dan jangan panggil `next()`.

3. Studi Kasus: Middleware untuk Proteksi Endpoint

- Menerapkan middleware otorisasi pada endpoint yang hanya boleh diakses oleh pengguna yang sudah login, misalnya: `GET /api/users/:id`, `POST /api/products` (jika hanya admin yang bisa

menambah produk), `GET /api/carts/:id`.
- Middleware ditempatkan di antara path route dan handler route.

```
const authMiddleware = require('./middlewares/auth'); // Asumsikan
middleware ada di sini

// Endpoint yang dilindungi
app.get('/api/users/:id', authMiddleware, (req, res) => {
  // Logic untuk mengambil data user berdasarkan ID
  // req.user sekarang berisi informasi dari JWT
});

// Endpoint publik (tidak dilindungi)
app.post('/api/login', (req, res) => {
  // Logic login
});
```

📝 Praktik Mandiri (8 Jam)

1. **Buat File Middleware:** Buat file baru (misalnya, `src/middlewares/auth.js`) untuk middleware otorisasi.
2. **Implementasi Middleware:** Tulis kode middleware otorisasi sesuai langkah-langkah di materi inti. Gunakan `jsonwebtoken.verify()` dan pastikan menangani kasus token tidak ada atau tidak valid.
3. **Integrasi Middleware:** Terapkan middleware otorisasi pada beberapa endpoint yang sudah ada dari minggu 4 (misalnya, `GET /api/products/:id`, `PUT /api/users/:id`, `DELETE /api/carts/:id`).
4. **Uji Middleware dengan Postman/Insomnia:**
    - Coba akses endpoint yang dilindungi tanpa menyertakan JWT. Pastikan mendapatkan response error (401/403).
    - Lakukan login untuk mendapatkan JWT.
    - Coba akses endpoint yang dilindungi dengan menyertakan JWT yang valid di header `Authorization: Bearer TOKEN`. Pastikan request berhasil.
    - Coba akses endpoint yang dilindungi dengan JWT yang tidak valid atau sudah expired. Pastikan mendapatkan response error.

💡 Tips untuk Pemula

- Pastikan secret key yang digunakan untuk verifikasi JWT sama persis dengan yang digunakan saat membuat JWT.
- Tangani error dari `jsonwebtoken.verify()` (misalnya, `TokenExpiredError`, `JsonWebTokenError`).
- Middleware otorisasi biasanya ditempatkan di awal rantai middleware untuk endpoint yang dilindungi.
- Anda bisa menambahkan informasi tambahan ke `req.user` di middleware (misalnya, role pengguna) untuk otorisasi berbasis peran (role-based access control - RBAC) di handler route.

📚 Referensi

- Express.js Middleware Docs
- Jsonwebtoken Docs (npm)
- Best Practices for Express.js Security