

Implementasi Zero Trust Access Control pada Aplikasi Sederhana

kelompok 4

Luthfi Kurniawan (2201020013)

M. Afief Anugrah (2201020015)

Aditya Firmansyah (2201020018)

Halta Putra Ash Sidiq (2201020092)

Minggu 3 : Implementasi role & protected route

1. Implementasi Role

A. Middleware Autentikasi: authenticateToken

Fungsi ini adalah lapisan pertahanan pertama yang mengatur apakah seorang pengguna login atau tidak, terlepas dari peran mereka.

```
● ● ●
1  export const authenticateToken = (
2    req: Request,
3    res: Response,
4    next: NextFunction
5  ) => {
6    const authHeader = req.headers["authorization"];
7
8    const token = authHeader && authHeader.split(" ")[1];
9
10   if (!token) {
11     return sendError(res, "Akses ditolak. Token tidak ditemukan.", 401);
12   }
13
14   const SECRET_KEY = process.env.JWT_SECRET || "rahasia_negara_api";
15
16   jwt.verify(token, SECRET_KEY, (err: any, user: any) => {
17     if (err) {
18       if (err.name === "TokenExpiredError") {
19         return sendError(
20           res,
21           "Sesi Anda telah berakhir. Silakan login ulang.",
22           401
23         );
24       }
25
26       if (err.name === "JsonWebTokenError") {
27         return sendError(res, "Token tidak valid. Silakan login ulang.", 401);
28       }
29
30       return sendError(res, "Autentikasi gagal.", 401);
31     }
32
33     (req as any).user = user;
34     next();
35   });
36 };
```

Komponen	Penjelasan Singkat	Fungsi Utama
authenticateToken	Middleware Autentikasi. Berjalan untuk semua <i>route</i> terproteksi.	Memverifikasi token JWT yang dikirim oleh pengguna.
Aksi Utama	Mengekstrak token dari <i>header</i> Authorization dan memverifikasinya menggunakan kunci rahasia (JWT_SECRET).	"Jika token valid, <i>payload</i> data pengguna (id, roleName, dll.) dilampirkan ke objek req.user."
Gagal	"Jika token hilang, palsu, atau kedaluwarsa."	Mengembalikan 401 Unauthorized .
Berhasil	Token valid.	Memanggil next() untuk melanjutkan ke <i>middleware</i> berikutnya (misalnya authorizeRole).

B. Middleware Otorisasi: authorizeRole

Fungsi ini adalah mekanisme **Role-Based Access Control (RBAC)** yang membatasi akses berdasarkan peran yang diizinkan.

```
● ● ●
1 export const authorizeRole = (allowedRoles: string[]) => {
2   return (req: Request, res: Response, next: NextFunction) => {
3     const userRole = req.user?.roleName;
4
5     if (!userRole) {
6       return sendError(
7         res,
8         "Akses Ditolak: Informasi peran tidak ditemukan.",
9         403
10    );
11  }
12
13  if (allowedRoles.includes("all")) {
14    return next();
15  }
16
17  if (allowedRoles.includes(userRole)) {
18    next();
19  } else {
20    return sendError(
21      res,
22      "Akses Ditolak: Peran Anda tidak memiliki izin untuk aksi ini.",
23      403
24    );
25  }
26};
27};
```

Komponen	Penjelasan Singkat	Fungsi Utama
authorizeRole(allowedRoles)	"Middleware Otorisasi. Menerima array peran yang diizinkan ('admin', 'editor')."	Membandingkan peran pengguna (req.user.roleName) dengan peran yang

		diizinkan untuk <i>route</i> tersebut.
Aksi Utama	Mengambil peran (<i>userRole</i>) dari <i>req.user</i> (yang sudah disediakan oleh <i>authenticateToken</i>).	"Jika <i>userRole</i> ditemukan di <i>allowedRoles</i> , akses diizinkan."
Kasus ['all']	"Mengizinkan semua peran yang terautentikasi (Admin, Editor, Viewer) untuk melanjutkan."	Digunakan pada <i>route</i> umum (misalnya GET /documents).
Gagal	"Peran pengguna (viewer) tidak termasuk dalam daftar yang diizinkan (['admin']). "	"Mengembalikan 403 Forbidden ("Akses Ditolak")."
Berhasil	Peran diizinkan.	Memanggil <i>next()</i> untuk melanjutkan ke <i>controller</i> (misalnya <i>createDocument</i>).

2. Contoh Penggunaan (Di Route)

Kode ini mendefinisikan *endpoint* untuk manajemen dokumen dan mengatur dua lapisan keamanan di *middleware*: Autentikasi (authenticateToken) dan Otorisasi (authorizeRole).



```
 1 import { Router } from "express";
 2 import documentsController from "../controllers/documents.controller";
 3 import {
 4   authenticateToken,
 5   authorizeRole,
 6 } from "../middleware/auth.middleware";
 7
 8 const router = Router();
 9
10 router.get(
11   "/",
12   authenticateToken,
13   documentsController.getAllDocuments.bind(documentsController)
14 );
15
16 router.get(
17   "/:id",
18   authenticateToken,
19   documentsController.getDocumentById.bind(documentsController)
20 );
21
22 router.post(
23   "/create",
24   authorizeRole(["admin", "editor"]),
25   documentsController.createDocument.bind(documentsController)
26 );
27
28 router.put(
29   "/:id",
30   authorizeRole(["admin", "editor"]),
31   documentsController.updateDocument.bind(documentsController)
32 );
33
34 export default router;
35
```

Endpoint	Metode	Middleware yang Diterapkan	Peran yang Diizinkan	Penjelasan RBAC
GET /	GET	authenticateToken	Semua Role	Mengambil daftar dokumen. Hanya memerlukan <i>login</i> saja. (Akses dibatasi ke data yang <i>Approved</i> di dalam controller).
GET /:id	GET	authenticateToken	Semua Role	Mengambil detail dokumen tunggal. Hanya memerlukan <i>login</i> saja.
POST /create	POST	authorizeRole(['admin', 'editor'])	Admin, Editor	Membuat dokumen baru. <i>Middleware</i> ini memastikan hanya peran kontributor yang dapat mengakses.

PUT /:id	PUT	authorizeRole(['admin', 'editor'])	Admin, Editor	Memperbarui dokumen. <i>Middleware</i> memastikan hanya peran yang memiliki hak <i>edit</i> yang dapat memanggil <i>route</i> ini.
----------	-----	------------------------------------	---------------	---

3. Verifikasi Implementasi

A. Pengujian Role-Based Access Control (RBAC) di Postman

Pengujian ini bertujuan untuk memverifikasi fungsionalitas dari kedua lapisan keamanan (*middleware authenticateToken* dan *authorizeRole*) yang telah diimplementasikan.

1. Proses cek semua dokumen

Sesuai konfigurasi *router*, *route* GET / hanya dilindungi oleh *middleware authenticateToken*. Logika ini hanya memeriksa apakah Token JWT valid dan belum kedaluwarsa. Setelah token valid, permintaan dilanjutkan ke *controller*.

```
● ● ●
1 router.get(
2   '/',
3   authenticateToken,
4   documentsController.getAllDocuments.bind(documentsController)
5 );
6
```

a. Sebagai viewer

Hasil Pengujian : Status 200 OK. Berhasil melihat data dokumen.

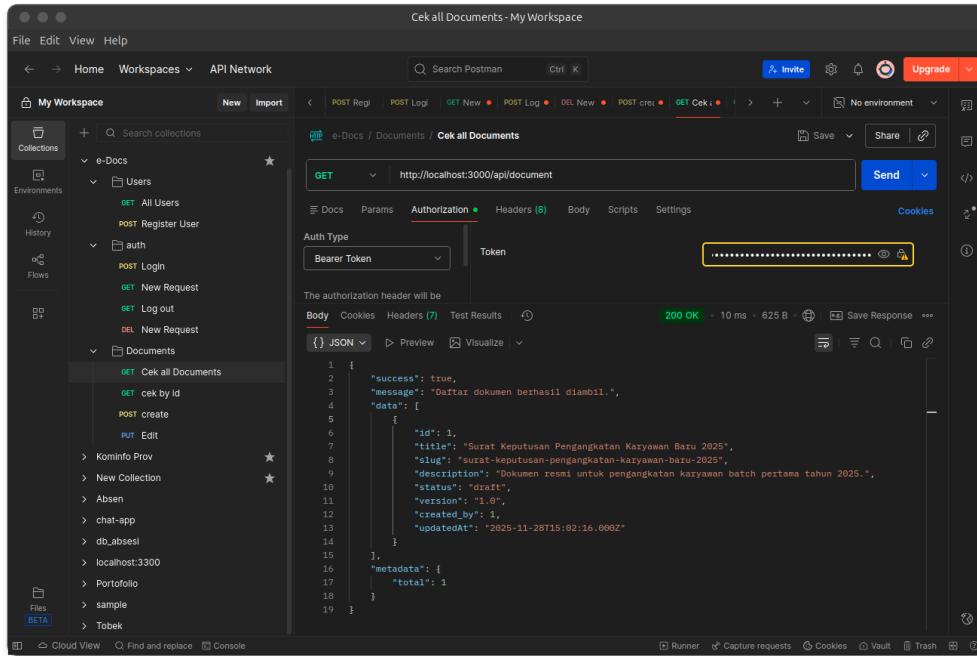
The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections like 'e-Docs' and 'auth'. In the main area, a request is being made to 'http://localhost:3000/api/document'. The method is 'GET', and the URL is 'http://localhost:3000/api/document'. The 'Authorization' tab is selected, showing 'Bearer Token' and a redacted token value. The 'Headers' tab shows '(8)' items. The response status is '200 OK' with a response time of '12 ms' and a body size of '625 B'. The response body is displayed in JSON format:

```
1 {
2   "success": true,
3   "message": "Daftar dokumen berhasil diambil.",
4   "data": [
5     {
6       "id": 1,
7       "title": "Surat Keputusan Pengangkatan Karyawan Baru 2025",
8       "slug": "surat-keputusan-pengangkatan-karyawan-baru-2025",
9       "description": "Dokumen resmi untuk pengangkatan karyawan batch pertama tahun 2025.",
10      "status": "draft",
11      "version": "1.0",
12      "created_by": 1,
13      "updatedAt": "2025-11-28T15:02:16.000Z"
14    },
15  ],
16  "metadata": {
17    "total": 1
18  }
19 }
```

Akses berhasil karena hanya diperlukan **Autentikasi** sesi aktif. Viewer memiliki hak akses dasar untuk melihat data.

b. Sebagai Editor

Hasil Pengujian : Status 200 OK Berhasil melihat data dokumen.



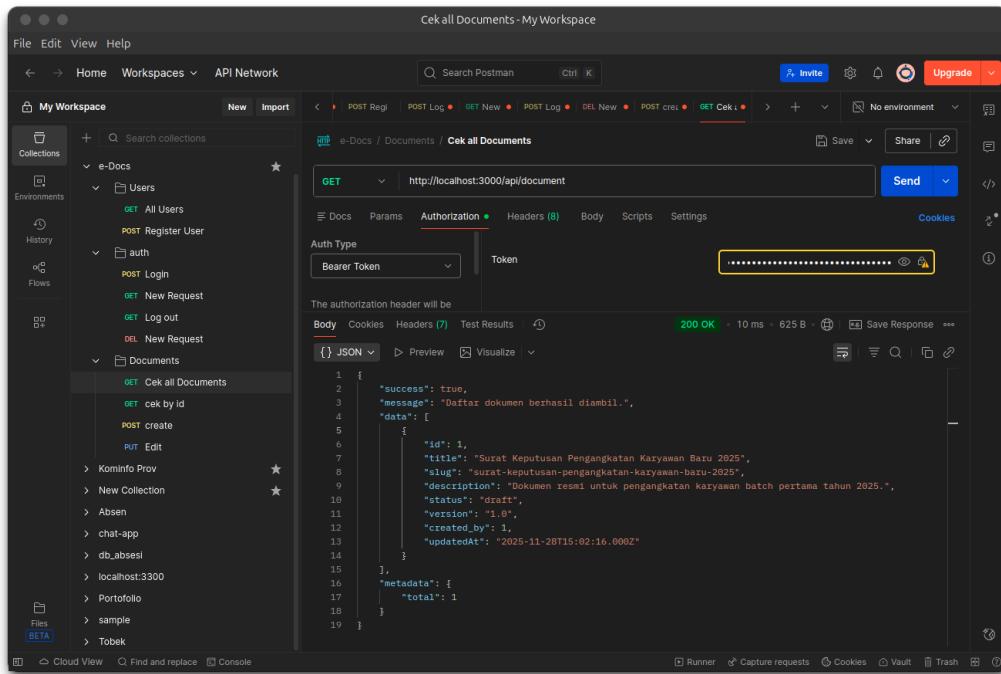
The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections like 'e-Docs', 'e-Absen', 'e-Chat', 'e-Portfolio', and 'e-Samples'. The main workspace shows a request for 'Cek all Documents' with a 'GET' method. The URL is set to 'http://localhost:3000/api/document'. In the 'Authorization' tab, 'Bearer Token' is selected, and a token value is entered. The 'Headers' tab shows 'Content-Type: application/json'. The 'Body' tab is set to 'JSON'. The 'Test Results' section shows a successful response with status '200 OK', time '10 ms', and size '625 B'. The JSON response body is displayed:

```
1 {  
2   "success": true,  
3   "message": "Daftar dokumen berhasil diambil.",  
4   "data": [  
5     {  
6       "id": 1,  
7       "title": "Surat Keputusan Pengangkatan Karyawan Baru 2025",  
8       "slug": "surat-keputusan-pengangkatan-karyawan-baru-2025",  
9       "description": "Dokumen resmi untuk pengangkatan karyawan batch pertama tahun 2025.",  
10      "status": "draft",  
11      "version": "1.0",  
12      "created_by": 1,  
13      "updated_at": "2025-11-28T15:02:16.000Z"  
14    },  
15  ],  
16  "metadata": {  
17    "total": 1  
18  }  
}
```

Akses berhasil. Semua peran yang terotentikasi diizinkan melihat daftar dokumen.

c. Sebagai Admin

Hasil Pengujian : Status 200 OK Berhasil melihat data dokumen.



The screenshot shows the Postman interface with the following details:

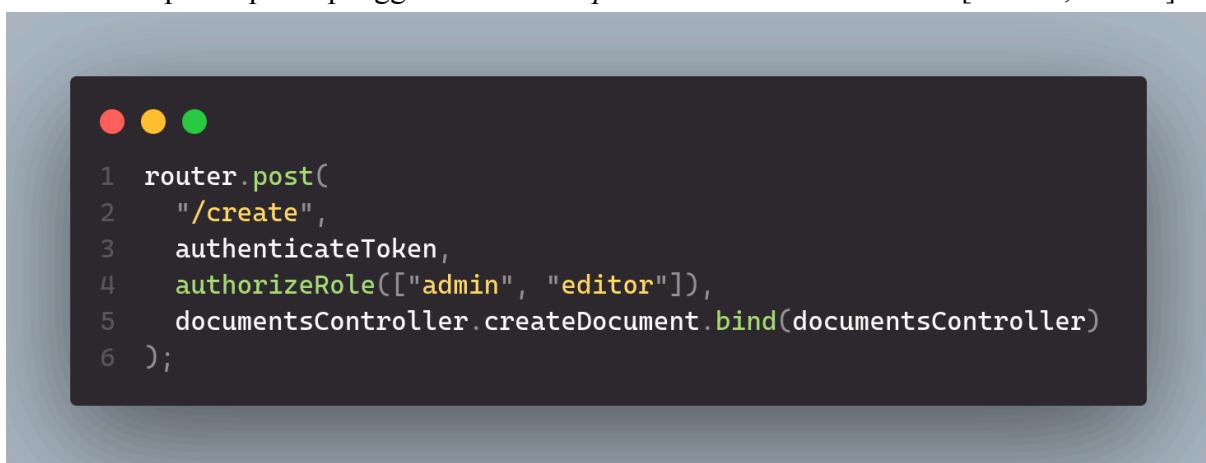
- Request URL:** http://localhost:3000/api/document
- Method:** GET
- Authorization:** Bearer Token (with a placeholder token)
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 {
2   "success": true,
3   "message": "Daftar dokumen berhasil diambil.",
4   "data": [
5     {
6       "id": 1,
7       "title": "Surat Keputusan Pengangkatan Karyawan Baru 2025",
8       "slug": "surat-keputusan-pengangkatan-karyawan-baru-2025",
9       "description": "Dokumen resmi untuk pengangkatan karyawan batch pertama tahun 2025.",
10      "status": "draft",
11      "version": "1.0",
12      "created_by": 1,
13      "updated_at": "2025-11-28T15:02:16.000Z"
14    },
15  ],
16  "metadata": {
17    "total": 1
18  }
19 }
```

Akses berhasil. Admin memiliki hak paling tinggi, sehingga lolos tahap autentikasi.

2. Pengujian untuk create documents

Route POST /create dilindungi oleh **dua middleware**: authenticateToken dan **authorizeRole(['admin', 'editor'])**. Setelah token diverifikasi, authorizeRole akan memeriksa apakah peran pengguna secara *eksplisit* termasuk dalam daftar ['admin', 'editor'].



```
1 router.post(
2   "/create",
3   authenticateToken,
4   authorizeRole(["admin", "editor"]),
5   documentsController.createDocument.bind(documentsController)
6 );
```

a. Sebagai Viewer

Hasil Pengujian : Status 403 Forbidden. Respons: "Akses Ditolak: Peran Anda tidak memiliki izin untuk aksi ini."

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists various collections and environments. In the center, a specific collection named 'e-Docs / Documents' is selected, and its 'create' endpoint is highlighted. The 'Body' tab shows a raw JSON payload being sent to the URL `http://localhost:3000/api/document/create`. The response on the right is a 403 Forbidden status with the message: "success": false, "message": "Akses Ditolak: Peran Anda tidak memiliki izin untuk aksi ini.".

Akses Ditolak. Ini adalah bukti kunci penerapan Zero Trust. Meskipun *Viewer* berhasil Terautentikasi (memiliki Token JWT valid), ia gagal di lapisan Otorisasi (RBAC). Sistem menolak akses karena *Viewer* tidak diberikan hak istimewa (privilege) untuk membuat dokumen.

b. Sebagai Editor

Hasil Pengujian : Status 201 Created.

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists various collections and environments. The main workspace shows a POST request to `http://localhost:3000/api/document/create`. The request body contains JSON data:

```
1 {  
2     "title": "Dokumen D - Review Konten Editor",  
3     "description": "Dokumen yang dibuat oleh peran Editor untuk pengujian kepemilikan dan hak edit DRAFT.",  
4     "markdown_content": "# Konten Dibuat Editor\n\nInilah adalah konten **Draft** yang dibuat oleh Editor B. Editor B harus  
dapat mengedit dokumen ini, tetapi tidak bisa disetujui sendiri.",  
5 }
```

The response tab shows the result of the request:

```
1 {  
2     "success": true,  
3     "message": "Dokumen berhasil dibuat dan disimpan sebagai Draft.",  
4     "data": {  
5         "id": 2,  
6         "title": "Dokumen D - Review Konten Editor",  
7         "slug": "dokumen-d-review-konten-editor",  
8         "status": "draft",  
9         "version": "1.0",  
10        "checksum": "551c69076c423331c614a4b92b60a068f69c3441f9178eb2988e9e1b758541e",  
11        "created_by": 2,  
12        "createdAt": "2025-11-28T17:19:21.256Z"  
13    }  
14 }
```

The status bar at the bottom indicates a **201 Created** response with a duration of 31 ms and a size of 586 B.

Akses Diizinkan. Editor adalah peran kontributor, membuktikan otorisasi berhasil.

c. Sebagai Admin

Hasil Pengujian : Status 201 Created.

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists various collections and environments. The main workspace shows a POST request to 'http://localhost:3000/api/document/create'. The request body contains JSON data representing a document creation. The response tab shows a 201 Created status with a JSON payload indicating success and providing document details like ID, title, and slug. The bottom navigation bar includes 'Cloud View', 'Find and replace', and 'Console'.

```
POST http://localhost:3000/api/document/create
{
  "title": "Dokumen A - Prosedur Rapat Mingguan",
  "description": "Dokumen yang mendefinisikan standar dan alur rapat mingguan departemen.",
  "markdown_content": "# Prosedur Rapat\n\n---\n\n**Agenda:** Wajib disiapkan H-1.\n\n**Kehadiran:** Semua staf manajerial diwajibkan hadir.\n\n**Catatan:** Dokumen ini bersifat internal dan wajib ditaati."
}

{
  "success": true,
  "message": "Dokumen berhasil dibuat dan disimpan sebagai Draft.",
  "data": {
    "id": 1,
    "title": "Dokumen A - Prosedur Rapat Mingguan",
    "slug": "dokumen-a-prosedur-rapat-mingguan",
    "status": "draft",
    "version": "1.0",
    "checksum": "a5439b3680c351ddfaec5927353605422ee6697b3884ce63733d81b9ef5d0f74",
    "created_by": 1,
    "createdAt": "2025-11-28T17:16:25.205Z"
  }
}
```

Akses Diizinkan. Admin memiliki izin kontributor. Ini adalah kasus sukses yang memverifikasi bahwa peran yang berhak dapat mengakses fungsi kritis.