

LAPORAN PRAKTIKUM TEKNOLOGI CLOUD COMPUTING

**PENAMBAHAN FITUR AUTENTIKASI LOGIN DAN REGISTER PADA
PROYEK NOTES MENGGUNAKAN JWT DAN BCRIPT TEKNOLOGI
CLOUD COMPUTING IF - F**



Disusun oleh

Nama : Aziizah Intan Ramadhan N
NIM : 123220201

**PROGRAM STUDI INFORMATIKA
JURUSAN INFORMATIKA
FAKULTAS TEKNIK INDUSTRI
UNIVERSITAS PEMBANGUNAN NASIONAL "VETERAN"
YOGYAKARTA
2025**

HALAMAN PENGESAHAN

LAPORAN PRAKTIKUM

PENAMBAHAN FITUR AUTENTIKASI LOGIN DAN REGISTER PADA PROYEK NOTES MENGGUNAKAN JWT DAN BCRIPT TEKNOLOGI CLOUD COMPUTING IF - F

Disusun Oleh :

Aziizah Intan Ramadhan N 123220201

Telah diperiksa dan disetujui oleh Asisten Praktikum.....

Pada tanggal :

Menyetujui.

Asisten Praktikum

Asisten Praktikum

Berlyandica Alam F
NIM 123210060

Faustina Chelloana T
NIM 123210139

KATA PENGANTAR

Sebagai ungkapan rasa syukur kepada Allah SWT atas rahmat dan hidayah-Nya, saya dapat menyelesaikan laporan Praktikum Teknologi Cloud Computing ini tepat pada waktunya. Laporan ini disusun sebagai bagian dari pemenuhan tugas praktikum serta sebagai sarana untuk memperdalam pemahaman mengenai konsep Autentikasi pada REST API.

Saya mengucapkan terima kasih yang sebesar-besarnya kepada Berlyandica Alam F dan Faustina Chelloana T selaku Asisten Praktikum Cloud Computing, yang telah memberikan tugas ini sehingga dapat menambah wawasan dan pengetahuan sesuai dengan bidang studi yang sedang ditekuni.

Saya berharap laporan ini dapat memberikan manfaat serta kontribusi positif bagi perkembangan ilmu pengetahuan di lingkungan praktikum Cloud Computing. Penulis menyadari bahwa pencapaian ini tidak terlepas dari bimbingan, arahan, dan dukungan dari Mas Berlyan dan Mba Chello selama proses pengerjaan.

Sebagai penutup, saya memohon maaf apabila terdapat kekurangan dalam laporan ini dan sangat mengharapkan kritik serta saran yang membangun guna perbaikan di masa mendatang.

Yogyakarta, 27 Mei 2025

Penulis

DAFTAR ISI

HALAMAN PENGESAHAN	2
KATA PENGANTAR	3
DAFTAR ISI	4
DAFTAR GAMBAR	5
BAB I	
PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	1
1.3 Tujuan	2
1.4 Manfaat	2
BAB II	
TINJAUAN LITERATUR	3
2.1 Autentikasi	3
2.2 JSON Web Token (JWT)	3
2.3 Proses Autentikasi Menggunakan JWT	4
BAB III	
METODOLOGI	6
3.1 Analisis Permasalahan	6
3.2 Perancangan Solusi	6
3.2.1 Mempersiapkan Aplikasi	7
3.2.2 Melakukan Deploy	11
3.2.3 Proses Penggantian URL Hasil Deploy	12
3.2.4 Pengujian Aplikasi	13
BAB IV	
HASIL DAN PEMBAHASAN	14
4.1 Hasil	14
4.1.1 Persiapan Aplikasi	14
4.1.4 Hasil Frontend	14
4.2 Pembahasan	15
BAB V	
PENUTUP	17
5.1 Kesimpulan	17
5.2 Saran	17
DAFTAR PUSTAKA	19

DAFTAR GAMBAR

Gambar 3.2.1.1 AuthController.js	8
Gambar 3.2.1.2 auth.js	8
Gambar 3.2.1.3 UserModel.js	8
Gambar 3.2.1.4 AuthRoute.js	9
Gambar 3.2.1.5 Login.js	9
Gambar 3.2.1.6 Register.js	10
Gambar 3.2.1.7 utils.js	10
Gambar 3.2.1.8 axiosInstance.js	11
Gambar 3.2.2.1 Deploy	12
Gambar 3.2.2.2 Berhasil di Deploy	12
Gambar 4.1.2.1 Hasil (1)	13
Gambar 4.1.2.2 Hasil (2)	13

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam pengembangan aplikasi web, aspek keamanan memegang peranan yang sangat penting, terutama pada sistem yang mengelola data pengguna. Salah satu komponen utama dalam keamanan aplikasi adalah autentikasi, yaitu proses verifikasi identitas pengguna sebelum diberikan akses ke sumber daya dalam sistem. Tanpa mekanisme autentikasi yang memadai, aplikasi berisiko mengalami akses tidak sah dan penyalahgunaan data.

JSON Web Token (JWT) merupakan salah satu metode autentikasi modern yang ringan dan efisien dalam mengelola sesi pengguna. JWT memungkinkan proses login dan pengelolaan sesi tanpa perlu menyimpan data pengguna secara berlebihan di sisi server, sehingga sangat sesuai untuk sistem yang bersifat stateless seperti aplikasi berbasis REST API.

Laporan ini membahas implementasi fitur autentikasi yang meliputi proses pendaftaran (register), masuk (login), dan keluar (logout) pada proyek aplikasi catatan (notes) yang telah dikembangkan sebelumnya. Fitur autentikasi ini diintegrasikan menggunakan JWT untuk menjamin keamanan dan kenyamanan dalam proses autentikasi pengguna.

1.2 Rumusan Masalah

Adapun rumusan masalah yang menjadi fokus dalam laporan ini adalah sebagai berikut:

1. Bagaimana cara menambahkan fitur pendaftaran (register), masuk (login), dan keluar (logout) pada aplikasi catatan (notes)?
2. Bagaimana mengimplementasikan autentikasi menggunakan JSON Web Token (JWT) pada sisi backend aplikasi?

3. Bagaimana proses pengamanan endpoint REST API melalui validasi token JWT dapat dilakukan secara efektif?
4. Bagaimana hasil pengujian terhadap fitur autentikasi serta interaksi antara frontend dan backend dalam aplikasi ini?

1.3 Tujuan

Tujuan dari penyusunan laporan ini adalah sebagai berikut:

1. Menjelaskan proses penambahan fitur autentikasi pada aplikasi catatan (notes).
2. Mendeskripsikan implementasi JSON Web Token (JWT) sebagai metode autentikasi.
3. Menganalisis integrasi autentikasi antara sisi backend dan frontend aplikasi.
4. Mengevaluasi aspek keamanan serta fungsionalitas dari fitur autentikasi yang telah diterapkan.

1.4 Manfaat

Manfaat yang dapat diperoleh dari penyusunan laporan ini antara lain:

1. Memberikan pemahaman yang mendalam mengenai mekanisme autentikasi berbasis token.
2. Meningkatkan aspek keamanan pada aplikasi web melalui penerapan JSON Web Token (JWT).
3. Menyediakan panduan teknis yang jelas dalam implementasi fitur login, pendaftaran (register), dan keluar (logout).
4. Mendorong pengembangan aplikasi yang lebih aman, scalable, dan profesional.

BAB II

TINJAUAN LITERATUR

Bab ini membahas teori serta konsep fundamental yang menjadi dasar dalam pengembangan fitur autentikasi, khususnya yang menggunakan JSON Web Token (JWT). Autentikasi merupakan elemen krusial dalam sistem keamanan aplikasi modern. Oleh karena itu, pemahaman mendalam mengenai prinsip kerja JWT serta penerapannya dalam arsitektur REST API sangat penting untuk memastikan bahwa sistem aplikasi yang dikembangkan tidak hanya aman, tetapi juga efisien dan mudah digunakan oleh pengguna.

2.1 Autentikasi

Autentikasi merupakan proses untuk memverifikasi identitas pengguna yang berusaha mengakses suatu sistem. Dalam konteks aplikasi web, autentikasi biasanya dilakukan melalui formulir login, di mana pengguna memasukkan kredensial seperti alamat email dan kata sandi. Sistem kemudian memvalidasi keabsahan data tersebut untuk memastikan bahwa pengguna memang berhak mengakses sistem.

Setelah proses autentikasi berhasil, pengguna diberikan hak akses sesuai dengan peran dan izin yang dimilikinya dalam sistem. Menurut Sharma & Goyal (2020), mekanisme autentikasi yang efektif harus mengutamakan aspek keamanan, kenyamanan pengguna, serta kemampuan skalabilitas. Hal ini penting untuk mencegah berbagai ancaman keamanan seperti serangan brute-force, pembajakan sesi (session hijacking), dan akses tidak sah (unauthorized access).

2.2 JSON Web Token (JWT)

JSON Web Token (JWT) merupakan standar terbuka yang diatur dalam RFC 7519, yang digunakan untuk pertukaran informasi secara aman dalam format objek JSON. JWT terdiri dari tiga komponen utama, yaitu Header, Payload, dan Signature. Setelah pengguna berhasil melakukan proses login,

server akan mengirimkan token yang telah dikodekan dan ditandatangani tersebut. Token ini kemudian disimpan di sisi klien, biasanya menggunakan `localStorage` atau `sessionStorage`, dan dikirimkan pada setiap permintaan ke server sebagai bukti autentikasi pengguna.

JWT banyak digunakan dalam berbagai arsitektur aplikasi modern, seperti Single Page Application (SPA) dan aplikasi mobile, karena sifatnya yang ringan, efisien, dan mudah diimplementasikan.

2.3 Proses Autentikasi Menggunakan JWT

Proses autentikasi menggunakan JWT dapat dijelaskan melalui langkah-langkah berikut:

1. Pengiriman Kredensial: Pengguna mengirimkan data kredensial, seperti email dan kata sandi, ke endpoint login pada server.
2. Verifikasi oleh Server: Server melakukan validasi terhadap kredensial yang diterima. Jika validasi berhasil, server akan membuat sebuah token JWT yang telah dikodekan dan ditandatangani, kemudian mengirimkannya kembali ke klien.
3. Penyimpanan Token oleh Klien: Token JWT yang diterima disimpan oleh klien, biasanya dalam `localStorage` atau `sessionStorage`.
4. Penggunaan Token pada Permintaan Selanjutnya: Pada setiap permintaan berikutnya yang memerlukan autentikasi, klien mengirimkan token tersebut melalui header HTTP Authorization dengan format Bearer <token>.
5. Verifikasi Token oleh Server: Server memeriksa keabsahan token pada setiap permintaan yang membutuhkan autentikasi untuk memastikan bahwa pengguna memiliki hak akses yang sesuai.

Selain itu, JWT mendukung fitur `exp` (expired) yang membatasi masa berlaku token, sehingga token hanya valid dalam jangka waktu tertentu. Untuk menjaga keamanan dan kenyamanan pengguna, JWT juga dapat dikombinasikan

dengan refresh token yang memungkinkan perpanjangan sesi secara aman tanpa perlu pengguna melakukan login ulang secara terus-menerus.

2.4 Bcrypt

Bcrypt merupakan algoritma hashing yang dirancang khusus untuk mengamankan password dengan cara yang efektif. Salah satu fitur utama Bcrypt adalah kemampuannya menambahkan "salt" atau data acak pada setiap password sebelum proses hashing dilakukan. Dengan adanya salt ini, hasil hash akan selalu berbeda meskipun password yang dimasukkan sama, sehingga metode serangan seperti rainbow table menjadi tidak efektif.

Dalam penerapannya, saat pengguna melakukan pendaftaran akun, password yang dimasukkan akan di-hash menggunakan Bcrypt terlebih dahulu sebelum disimpan ke dalam database. Selanjutnya, ketika pengguna melakukan login, password yang dimasukkan akan di-hash kembali dan dibandingkan dengan hash yang tersimpan. Jika keduanya cocok, maka proses autentikasi dianggap berhasil.

Selain itu, Bcrypt dirancang untuk memperlambat proses hashing secara sengaja, sehingga dapat menghambat upaya brute-force attack yang mencoba menebak password dengan cepat. Karena alasan ini, Bcrypt sangat cocok digunakan dalam sistem autentikasi pada aplikasi web yang membutuhkan tingkat keamanan tinggi.

BAB III

METODOLOGI

3.1 Analisis Permasalahan

Aplikasi catatan yang dikembangkan sebelumnya belum dilengkapi dengan sistem autentikasi, sehingga semua pengguna dapat mengakses, menambah, dan mengubah data tanpa adanya pembatasan. Kondisi ini menimbulkan risiko keamanan karena tidak terdapat mekanisme verifikasi identitas pengguna maupun pengelompokan data berdasarkan kepemilikan.

Untuk mengatasi permasalahan tersebut, perlu diterapkan fitur autentikasi yang mencakup proses pendaftaran (register), masuk (login), dan keluar (logout) menggunakan JSON Web Token (JWT). Dengan autentikasi ini, hanya pengguna yang telah terverifikasi yang dapat mengakses dan mengelola data mereka secara eksklusif. Integrasi JWT pada sisi backend dan frontend akan meningkatkan keamanan aplikasi serta mempersiapkannya untuk digunakan oleh banyak pengguna secara aman.

3.2 Perancangan Solusi

Perancangan solusi difokuskan pada integrasi fitur autentikasi menggunakan JSON Web Token (JWT) ke dalam aplikasi catatan yang sudah ada, mencakup kedua sisi, yaitu frontend dan backend.

- Backend: Akan disediakan endpoint khusus untuk proses pendaftaran (register), masuk (login), dan keluar (logout). Selain itu, backend juga akan menggunakan middleware untuk memverifikasi token JWT pada setiap permintaan yang memerlukan autentikasi, sehingga hanya pengguna yang terverifikasi yang dapat mengakses sumber daya tertentu.
- Frontend: Akan disesuaikan agar dapat menyimpan token JWT yang diterima setelah login, serta menggunakannya dalam header permintaan API berikutnya untuk mengakses fitur yang memerlukan autentikasi.

Dengan pendekatan ini, aplikasi catatan akan memiliki sistem autentikasi yang aman dan terstruktur, memastikan bahwa hanya pengguna yang berwenang yang dapat mengakses dan mengelola data mereka.

3.2.1 Mempersiapkan Aplikasi

Tahap awal pengembangan dimulai dengan merombak struktur aplikasi pada sisi backend dan frontend.

- Backend: Ditambahkan model User untuk menyimpan data pengguna baru, seperti nama, email, dan password yang telah dienkripsi menggunakan bcrypt. Selain itu, dibuat endpoint untuk proses pendaftaran (register), masuk (login), dan keluar (logout) dengan penerapan autentikasi berbasis JSON Web Token (JWT).
- Frontend: Dimodifikasi dengan penambahan halaman login dan register. Setiap permintaan yang mengakses data catatan dari backend kini menggunakan token JWT yang disimpan di localStorage. Token ini dikirim melalui header Authorization dalam format Bearer <token>, sehingga hanya pengguna yang terverifikasi dapat mengakses catatan miliknya sendiri.

Pada tahap ini, aplikasi masih dijalankan secara lokal dengan menggunakan database lokal sebagai persiapan sebelum deployment ke lingkungan produksi.

Backend:

```

backend > controllers > JS AuthController.js > @ register
1 import User from "../models/UserModel.js";
2 import bcrypt from "bcryptjs";
3 import jwt from "jsonwebtoken";
4
5 // Register
6 export const register = async (req, res) => {
7   const { username, email, password } = req.body;
8
9   try {
10    if (!username || !email || !password) {
11      return res.status(400).json({ msg: "Semua field wajib diisi." });
12    }
13
14    const userExist = await User.findOne({ where: { email } });
15    if (userExist) return res.status(400).json({ msg: "Email sudah terdaftar" });
16
17    const usernameExist = await User.findOne({ where: { username } });
18    if (usernameExist) return res.status(400).json({ msg: "Username sudah digunakan" });
19
20    await User.create({ username, email, password });
21    res.status(201).json({ msg: "User berhasil didaftarkan" });
22
23  } catch (error) {
24    console.error("X Register Error:", error);
25    res.status(500).json({ msg: "Terjadi kesalahan pada server", error: error.message });
26  }
27 };
28
29 // Login
30 export const login = async (req, res) => {
31   const { email, password } = req.body;
32
33   try {
34     if (!email || !password) {
35       return res.status(400).json({ msg: "Email dan password wajib diisi." });
36     }
37
38     const user = await User.findOne({ where: { email } });
39
40     if (!user) {
41       console.warn("⚠ Login gagal: user dengan email '${email}' tidak ditemukan.");
42       return res.status(404).json({ msg: "User tidak ditemukan" });
43     }
44
45     const match = await bcrypt.compare(password, user.password);
46
47     if (!match) {
48       console.warn("⚠ Password salah untuk email '${email}'");
49       return res.status(400).json({ msg: "Password salah" });
50     }
51   }
52 };

```

Gambar 3.2.1.1 AuthController.js

```

backend > middleware > JS authjs > ...
1 // middleware/auth.js
2 import jwt from "jsonwebtoken";
3
4 export const verifyToken = (req, res, next) => {
5   const authHeader = req.headers["authorization"];
6   if (!authHeader) return res.status(403).json({ message: "No token provided" });
7
8   const token = authHeader.split(" ")[1];
9   if (!token) return res.status(403).json({ message: "No token provided" });
10
11   jwt.verify(token, process.env.JWT_SECRET, (err, user) => {
12     if (err) return res.status(403).json({ message: "Invalid token" });
13     req.user = user;
14     next();
15   });
16 };
17

```

Gambar 3.2.1.2 auth.js

```

backend > models > JS UserModel.js > ...
1 import { DataTypes } from "sequelize";
2 import bcrypt from "bcryptjs";
3 import db from "../config/Database.js";
4
5 const User = db.define("users", {
6   id: {
7     type: DataTypes.INTEGER,
8     primaryKey: true,
9     autoIncrement: true
10   },
11   username: {
12     type: DataTypes.STRING,
13     allowNull: false,
14     unique: true
15   },
16   email: {
17     type: DataTypes.STRING,
18     allowNull: false,
19     unique: true,
20     validate: {
21       isEmail: true
22     }
23   },
24   password: {
25     type: DataTypes.STRING
26   }
27 }, {
28   hooks: {
29     beforeCreate: async (user) => {
30       if (user.password) {
31         const salt = await bcrypt.genSalt(10);
32         user.password = await bcrypt.hash(user.password, salt);
33       }
34     },
35     beforeUpdate: async (user) => {
36       if (user.changed("password")) {
37         const salt = await bcrypt.genSalt(10);
38         user.password = await bcrypt.hash(user.password, salt);
39       }
40     }
41   }
42 });

```

Gambar 3.2.1.3 UserModel.js

```

backend > routes > JS AuthRoute.js > ...
1
2 import express from "express";
3 import { register, login } from "../controllers/AuthController.js";
4
5 const router = express.Router();
6
7 router.post('/register', register);
8 router.post('/login', login);
9
10
11 export default router;
12

```

Gambar 3.2.1.4 AuthRoute.js

Frontend:

```

frontend > src > components > JS Login.js > Login > @ handleSubmit
1 // src/components/Login.js
2 import React, { useState } from "react";
3 import { useNavigate, Link } from "react-router-dom";
4
5 function Login() {
6   const [email, setEmail] = useState("");
7   const [password, setPassword] = useState("");
8   const [message, setMessage] = useState("");
9   const navigate = useNavigate();
10
11   const handleSubmit = async (e) => {
12     e.preventDefault();
13     setMessage("");
14
15     try {
16       const response = await fetch("http://localhost:5000/api/auth/login", {
17         method: "POST",
18         headers: { "Content-Type": "application/json" },
19         body: JSON.stringify({ email, password }),
20       });
21
22       const data = await response.json();
23
24       if (response.ok) {
25         localStorage.setItem("token", data.token); // Simpan token
26         navigate("/dashboard"); // Navigasi ke path dashboard yang benar
27       } else {
28         setMessage(data.message || "Login gagal");
29       }
30     } catch (error) {
31       setMessage("Terjadi kesalahan jaringan");
32     }
33   };
34
35   return (
36     <div style={{ maxWidth: "400px", margin: "50px auto", padding: "20px" }}>
37       <h2>Login</h2>
38       <message && <div style={{ color: "red", marginBottom: "15px" }}>{message}</div>
39       <form onSubmit={handleSubmit}>
40         <input
41           type="email"
42           placeholder="Email"
43           value={email}
44           onChange={e => setEmail(e.target.value)}
45           required
46           style={{ width: "100%", padding: "10px", marginBottom: "15px" }}
47         />
48         <input
49           type="password"
50           placeholder="Password"
51           value={password}
52           onChange={e => setPassword(e.target.value)}
53           required
54           style={{ width: "100%", padding: "10px", marginBottom: "15px" }}
55         />
56         <button

```

Gambar 3.2.1.5 Login.js

```

frontend > src > components > JS Register.js > ...
44  };
45
46  function Register() {
47    // Gunakan state object untuk semua input
48    const [form, setForm] = useState({
49      username: "",
50      email: "",
51      password: "",
52    });
53
54    const [message, setMessage] = useState("");
55    const navigate = useNavigate();
56
57    // Fungsi untuk handle perubahan input
58    const handleChange = (e) => {
59      setForm({
60        ...form,
61        [e.target.name]: e.target.value,
62      });
63    };
64
65    const handleSubmit = async (e) => {
66      e.preventDefault();
67      setMessage("");
68
69      try {
70        const response = await fetch("http://localhost:5000/api/auth/register", {
71          method: "POST",
72          headers: { "Content-Type": "application/json" },
73          // Kirim semua field username, email, password
74          body: JSON.stringify({
75            username: form.username,
76            email: form.email,
77            password: form.password,
78          }),
79        });
80
81        const data = await response.json();
82
83        if (response.ok) {
84          navigate("/login");
85        } else {
86          setMessage(data.message || "Registrasi gagal");
87        }
88      } catch (error) {
89        setMessage("Terjadi kesalahan jaringan");
90      }
91    };
92
93    return (
94      <div style={containerStyle}>
95        <h2>Register</h2>
96        {message && <div style={messageStyle}>{message}</div>}
97        <form onSubmit={handleSubmit}>
98          <input
99            style={inputStyle}

```

Gambar 3.2.1.6 Register.js

```

frontend > src > JS utils.js > ...
7  const TOKEN_KEY = "accessToken";
8
9  // Simpan token ke localStorage
10 export const setToken = (token) => {
11   localStorage.setItem("token", token);
12 };
13
14 // Ambil token dari localStorage
15 export const getToken = () => {
16   return localStorage.getItem("token");
17 };
18
19 // Hapus token dari localStorage (misal saat logout)
20 export const removeToken = () => {
21   localStorage.removeItem("token");
22 };
23
24 // Fungsi fetch dengan otentikasi token
25 export const authFetch = async (url, options = {}) => {
26   const token = getToken();
27   const headers = {
28     "Content-Type": "application/json",
29     ...(token && { Authorization: `Bearer ${token}` }),
30     ...options.headers,
31   };
32
33   const response = await fetch(url, { ...options, headers });
34
35   if (response.status === 401 || response.status === 403) {
36     // Token invalid atau expired, hapus token dan redirect ke login
37     removeToken();
38     window.location.href = "/login";
39   }
40
41   return response;
42 };

```

Gambar 3.2.1.7 utils.js

```

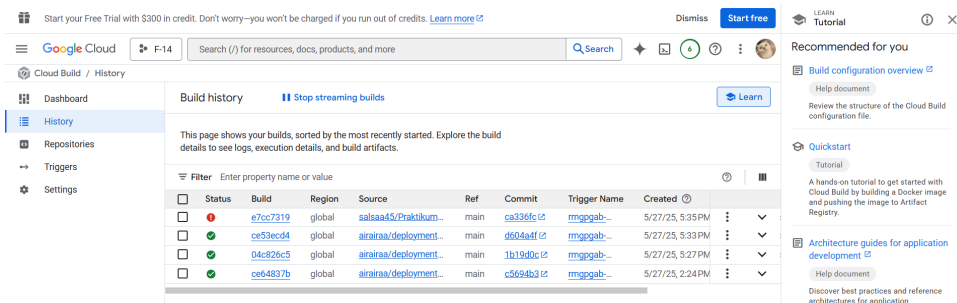
frontend > src > api > JS axiosInstance.js > axiosInstance
1 import axios from "axios";
2
3 const axiosInstance = axios.create({
4   baseURL: "http://localhost:5000",
5   headers: {
6     "Content-Type": "application/json",
7   },
8   withCredentials: true,
9 });
10
11 // Interceptor untuk menambahkan token ke header Authorization
12 axiosInstance.interceptors.request.use((config) => {
13   const token = localStorage.getItem("token"); // atau dari context/state
14   if (token) {
15     config.headers.Authorization = `Bearer ${token}`;
16   }
17   return config;
18 });
19
20 export default axiosInstance;

```

Gambar 3.2.1.8 axiosInstance.js

3.2.2 Melakukan Deploy

Tahap selanjutnya dalam pengembangan adalah melakukan deployment aplikasi ke Google Cloud Platform. Karena pengembangan dilakukan pada branch “tugas7” yang belum memiliki trigger otomatis di Cloud Build, maka langkah pertama adalah melakukan merge branch “tugas7” ke branch “main”. Hal ini bertujuan agar pipeline CI/CD yang sudah dikonfigurasi sebelumnya dapat berjalan secara otomatis.

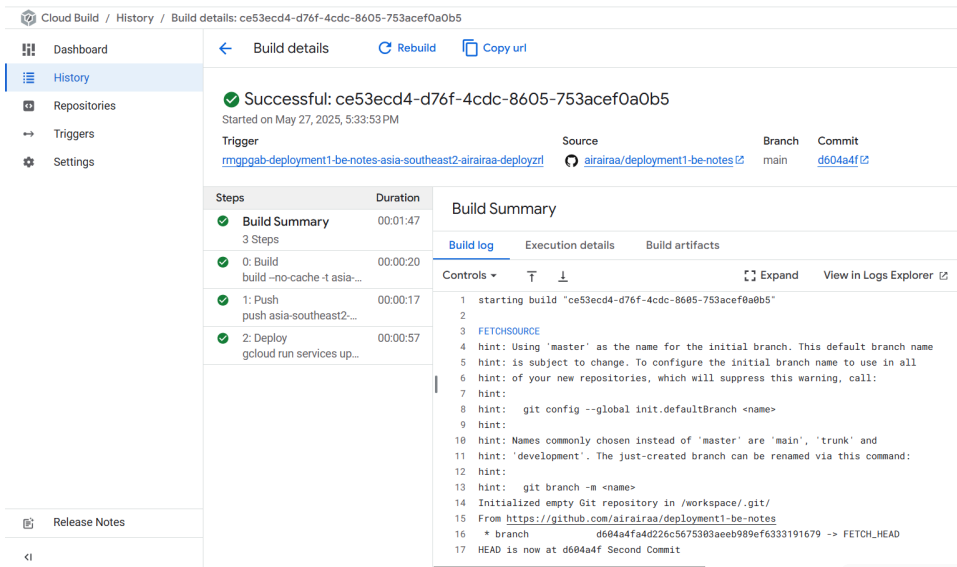


The screenshot shows the Google Cloud Build console. The 'Build history' tab is active, displaying a table of recent builds. The table has columns for Status, Build ID, Region, Source, Ref, Commit, Trigger Name, and Created. There are four builds listed, all with a status of 'Succeeded' (green checkmark). The first build has a red status icon, likely due to a missing image in the screenshot.

Status	Build	Region	Source	Ref	Commit	Trigger Name	Created
	e7cc7319	global	salaa45/Praktikum...	main	ca336fc	rmggab...	5/27/25, 5:35 PM
	ce53ec04	global	airairaa/deployment...	main	d604a4f	rmggab...	5/27/25, 5:33 PM
	04c826c5	global	airairaa/deployment...	main	1b19d0c	rmggab...	5/27/25, 5:27 PM
	ce64837b	global	airairaa/deployment...	main	c5694b3	rmggab...	5/27/25, 2:24 PM

Gambar 3.2.2 Deploy

Setelah proses merge selesai, pipeline Cloud Build akan secara otomatis melakukan build dan deploy aplikasi backend ke Cloud Run serta frontend ke App Engine. Pada tahap ini juga dilakukan konfigurasi tambahan untuk menyertakan variabel lingkungan (environment variables) yang diperlukan oleh aplikasi, seperti URL backend dan secret key JWT, agar aplikasi dapat berjalan dengan konfigurasi yang tepat di lingkungan produksi.



Gambar 3.2.2.2 Berhasil di Deploy

3.2.3 Proses Penggantian URL Hasil Deploy

Setelah proses deployment selesai, langkah berikutnya adalah menyesuaikan URL pada sisi frontend agar dapat mengakses API backend yang telah di-deploy di Cloud Run. Penyesuaian ini penting agar frontend dapat berkomunikasi dengan backend yang sudah berjalan di lingkungan produksi.

Di sisi backend, juga dilakukan konfigurasi Cross-Origin Resource Sharing (CORS) untuk mengizinkan permintaan dari domain frontend yang telah di-deploy di App Engine. Konfigurasi ini memastikan bahwa browser mengizinkan komunikasi antara frontend dan backend yang berada di domain berbeda.

Setelah penggantian URL dan konfigurasi CORS berhasil dilakukan, seluruh perubahan dikomit dan dipush kembali ke branch main. Hal ini bertujuan untuk memastikan pipeline CI/CD berjalan dengan baik dan memperbarui deployment dengan URL yang baru. Dengan langkah ini, koneksi antara aplikasi frontend dan backend di lingkungan produksi dapat berjalan dengan lancar dan aman.

3.2.4 Pengujian Aplikasi

Tahap terakhir dalam proses implementasi adalah melakukan pengujian menyeluruh terhadap seluruh fitur autentikasi yang telah dikembangkan. Pengujian dilakukan melalui antarmuka frontend dengan langkah-langkah sebagai berikut:

1. Proses Registrasi: Pengguna melakukan pendaftaran akun baru melalui halaman register.
2. Proses Login: Pengguna masuk menggunakan kredensial yang telah didaftarkan. Jika login berhasil, token JWT akan disimpan secara aman di localStorage.
3. Akses Halaman Utama: Setelah login, aplikasi secara otomatis mengarahkan pengguna ke halaman utama yang menampilkan catatan pribadi mereka.
4. Pengamanan Akses Data: Setiap permintaan frontend untuk mengakses data catatan menyisipkan token JWT ke dalam header permintaan. Token ini kemudian diverifikasi oleh middleware di backend.
5. Validasi Keamanan: Hanya permintaan dengan token yang valid yang diproses, sehingga memastikan bahwa data pengguna terlindungi dan akses antar pengguna tidak tercampur.

Pengujian ini membuktikan bahwa sistem autentikasi berbasis JWT berfungsi dengan baik, termasuk dalam hal pengamanan endpoint dan perlindungan data antar pengguna, sehingga aplikasi dapat berjalan dengan aman dan andal.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil

Pada bagian ini, ditampilkan hasil dari implementasi fitur autentikasi yang telah dirancang pada BAB III. Fitur tersebut mencakup penambahan mekanisme register, login, dan logout menggunakan JSON Web Token (JWT) pada proyek aplikasi catatan (notes). Implementasi ini bertujuan untuk meningkatkan keamanan dan pengelolaan akses pengguna dalam aplikasi.

4.1.1 Persiapan Aplikasi

Melakukan perombakan menyeluruh pada struktur aplikasi, baik di sisi backend maupun frontend.

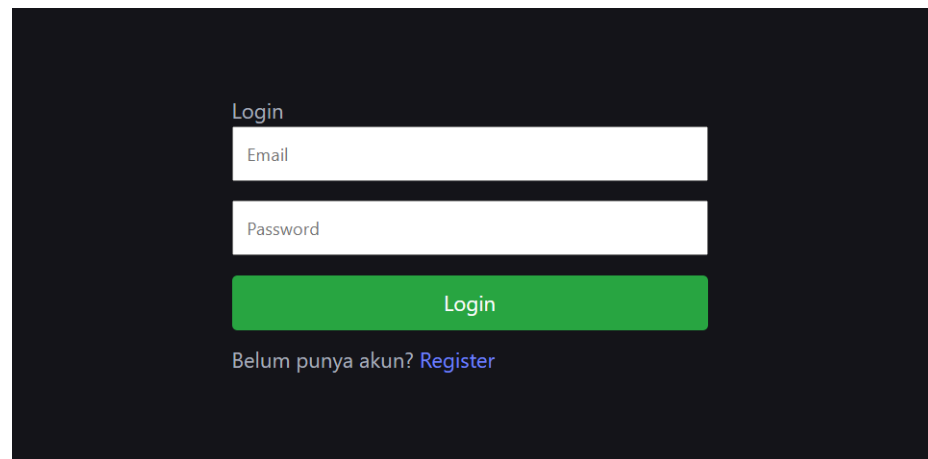
- Backend: Ditambahkan model User yang berfungsi sebagai representasi pengguna dalam sistem. Selain itu, disiapkan endpoint khusus untuk pengelolaan pengguna yang menggunakan mekanisme token berbasis JWT, sehingga keamanan dan validitas autentikasi dapat terjamin dengan lebih baik.
- Frontend: Ditambahkan halaman-halaman baru yang mendukung proses autentikasi, seperti halaman Login dan Register. Selain itu, akses ke fitur catatan (notes) kini memerlukan token JWT yang valid, sehingga hanya pengguna yang terautentikasi yang dapat menggunakan fitur tersebut.

Perombakan ini bertujuan untuk memperkuat sistem autentikasi dan memastikan bahwa hanya pengguna yang sah yang dapat mengakses data dan fitur dalam aplikasi.

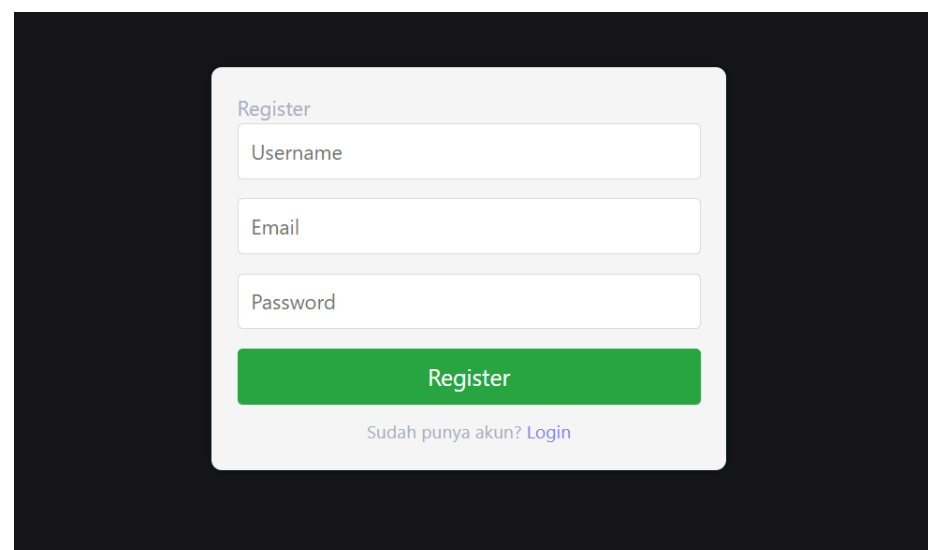
4.1.4 Hasil

Pengujian pada aplikasi frontend difokuskan pada proses autentikasi login. Setelah pengguna berhasil melakukan login, aplikasi akan menampilkan seluruh data catatan (notes) yang sesuai dengan pengguna tersebut.

Untuk pengambilan data pengguna, aplikasi memerlukan token autentikasi yang valid. Token ini merupakan bagian dari mekanisme JSON Web Token (JWT) yang digunakan untuk menjamin keamanan dan validitas akses data. Dengan demikian, proses pengambilan data sudah mencakup penggunaan token JWT secara efektif, memastikan bahwa hanya pengguna yang terautentikasi yang dapat mengakses data mereka.

A screenshot of a login form on a dark background. The form is titled "Login" in a light blue font. It contains two white input fields: "Email" and "Password". Below these fields is a green button with the text "Login" in white. At the bottom of the form, there is a link that says "Belum punya akun? Register" in a light blue font.

Gambar 4.1.2.1 Hasil (1)

A screenshot of a register form on a dark background. The form is titled "Register" in a light blue font. It contains three white input fields: "Username", "Email", and "Password". Below these fields is a green button with the text "Register" in white. At the bottom of the form, there is a link that says "Sudah punya akun? Login" in a light blue font.

Gambar 4.1.2.2 Hasil (2)

4.2 Pembahasan

Pada bagian ini dibahas hasil dari implementasi fitur autentikasi yang telah dijelaskan sebelumnya. Pembahasan mencakup beberapa aspek penting, yaitu:

1. Pencapaian Tujuan: Fitur autentikasi berbasis JWT berhasil diimplementasikan sesuai dengan tujuan yang telah ditetapkan, yaitu meningkatkan keamanan dan pengelolaan akses pengguna dalam aplikasi catatan.
2. Evaluasi Teknis: Secara teknis, sistem autentikasi berjalan dengan baik, termasuk proses pembuatan akun, login, penyimpanan token JWT, serta validasi token pada setiap permintaan data. Mekanisme ini memastikan bahwa hanya pengguna yang terverifikasi yang dapat mengakses data mereka.
3. Keterkaitan dengan Teori: Implementasi ini sesuai dengan teori-teori autentikasi dan keamanan yang telah diuraikan sebelumnya, khususnya mengenai penggunaan token JWT sebagai metode autentikasi yang efisien dan aman.
4. Tantangan yang Dihadapi: Selama proses implementasi, terdapat beberapa tantangan teknis seperti pengaturan token pada frontend dan backend, serta sinkronisasi antara proses autentikasi dan pengelolaan data pengguna yang harus diatasi.
5. Manfaat Nyata: Penerapan autentikasi JWT memberikan manfaat nyata berupa peningkatan keamanan aplikasi, kemudahan dalam pengelolaan sesi pengguna, serta perlindungan data yang lebih baik antar pengguna.
6. Pembahasan ini memberikan gambaran menyeluruh mengenai keberhasilan dan aspek-aspek penting yang perlu diperhatikan dalam pengembangan fitur autentikasi berbasis JWT pada aplikasi catatan.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil implementasi dan pengujian yang telah dilakukan, dapat disimpulkan bahwa penambahan fitur autentikasi pada aplikasi catatan dengan menggunakan JSON Web Token (JWT) berhasil diterapkan dengan baik. Fitur utama seperti proses pendaftaran (register), masuk (login), dan keluar (logout) telah terintegrasi secara menyeluruh pada sisi backend maupun frontend.

Pengguna kini dapat melakukan autentikasi dengan aman untuk mengakses dan mengelola data catatan secara pribadi dan efisien. Penggunaan JWT sebagai metode autentikasi memberikan keuntungan berupa fleksibilitas dan tingkat keamanan yang tinggi, karena sifatnya yang stateless dan mudah dikelola.

Selain itu, integrasi sistem Continuous Integration/Continuous Deployment (CI/CD) memastikan bahwa proses deployment aplikasi berjalan secara otomatis dan konsisten, mendukung kelancaran pengembangan dan pemeliharaan aplikasi. Dengan demikian, aplikasi catatan ini menjadi lebih siap untuk digunakan secara luas dan sejalan dengan praktik pengembangan perangkat lunak modern yang berbasis REST API. Mba Chello dan Mas Berlyan maaf ya...

5.2 Saran

Untuk meningkatkan kualitas dan keamanan aplikasi autentikasi, terdapat beberapa rekomendasi yang dapat dipertimbangkan dalam pengembangan selanjutnya, antara lain:serta skalabilitas yang lebih baik untuk mendukung pertumbuhan pengguna dan kompleksitas aplikasi.

- Pengalaman Pengguna (UX) dan Dokumentasi
 - Peningkatan UI/UX pada Proses Autentikasi: Buat tampilan login, register, dan logout yang lebih intuitif dan responsif, serta berikan feedback yang jelas saat terjadi kesalahan autentikasi.

- Dokumentasi API yang Lengkap: Sediakan dokumentasi API yang lengkap dan mudah dipahami, termasuk contoh penggunaan token JWT, untuk memudahkan pengembang lain dalam integrasi dan pengembangan lebih lanjut.
- Analitik dan Pelaporan
 - Analitik Penggunaan Autentikasi: Tambahkan fitur analitik untuk memantau pola penggunaan autentikasi, seperti jumlah login harian, lokasi login, dan perangkat yang digunakan. Ini dapat membantu dalam mendeteksi aktivitas mencurigakan.
 - Laporan Keamanan Berkala: Buat laporan rutin terkait keamanan autentikasi, termasuk insiden keamanan, upaya login gagal, dan aktivitas mencurigakan lainnya.

Dengan mengimplementasikan saran-saran di atas, aplikasi autentikasi berbasis JWT tidak hanya akan lebih aman dan andal, tetapi juga memberikan pengalaman pengguna yang lebih baik serta mendukung pengelolaan dan pemantauan yang lebih efektif. Ini akan sangat membantu dalam menjaga integritas data dan kepercayaan pengguna terhadap aplikasi.

DAFTAR PUSTAKA

- Auth0. (2023). Introduction to JSON Web Tokens. auth0.com
- IETF. (2015). RFC 7519 - JSON Web Token (JWT). tools.ietf.org
- OpenID Foundation. (2014). OpenID Connect Core 1.0. openid.net
- IETF. (2012). OAuth 2.0 Authorization Framework. tools.ietf.org
- Danutirta, N. R., & Christy, Y. L. (2021). Implementasi JSON Web Token untuk Dynamic Access Rights Authentication pada Aplikasi Klinik Pratama UPN “Veteran” Yogyakarta Berbasis RESTful API. eprints.utdi.ac.id
- Rahmatulloh, A., Sulastri, H., & Nugroho, R. (2018). Keamanan RESTful Web Service Menggunakan JSON Web Token (JWT) HMAC SHA-512. eprints.umpo.ac.id