

LAPORAN PRAKTIKUM TEKNOLOGI CLOUD COMPUTING

INTEGRASI AUTENTIKASI DENGAN JASON WEB TOKEN



Disusun oleh

Nama : Irsyad Khairullah

NIM : 123220176

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

HALAMAN PENGESAHAN

LAPORAN PRAKTIKUM

INTEGRASI AUTENTIKASI DENGAN JASON WEB TOKEN

Disusun Oleh :

Irsyad Khairullah 123220176

Telah diperiksa dan disetujui oleh Asisten Praktikum Teknologi

Cloud Computing

Pada tanggal :

Menyetujui.

Asisten Praktikum

Asisten Praktikum

Berlyandhica Alam Febriwantoro
NIM 123210060

Rafli Iskandar Kavarera
NIM 123210131

KATA PENGANTAR

Segala puji bagi Allah SWT yang telah memberikan kemudahan sehingga saya dapat menyelesaikan laporan ini dengan tepat waktu. Tanpa pertolongannya tentunya saya tidak akan sanggup untuk menyelesaikan laporan ini dengan baik.

Tidak lupa saya ucapkan terimakasih kepada Berlyandhica Alam Febriwanto dan Rafli Iskandar Kavarera selaku asisten lab pengampu matakuliah Praktikum Teknologi Cloud Computing. Dalam laporan ini saya menjelaskan tentang Integrasi Autentikasi Dengan Jason Web Token.

Saya menyadari bahwa laporan ini masih memiliki kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan untuk perbaikan ke depannya. Semoga laporan ini dapat bermanfaat bagi pembaca dan menjadi referensi dalam kegiatan praktikum selanjutnya.

Yogyakarta, 3 Maret 2025

Penulis

DAFTAR ISI

HALAMAN PENGESAHAN	ii
KATA PENGANTAR	iii
DAFTAR ISI	iv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	2
1.3 Tujuan	2
1.4 Manfaat	2
BAB II TINJAUAN LITERATUR	3
2.1 Autentikasi	3
BAB III METODOLOGI	4
3.1 Analisis Permasalahan	4
3.2 Perancangan Solusi	4
BAB IV HASIL DAN PEMBAHASAN	47
4.1 Hasil	47
4.2 Pembahasan	53
BAB V PENUTUP	54
5.1 Kesimpulan	54
5.2 Saran	54
DAFTAR PUSTAKA	55

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam era digital yang semakin berkembang pesat, keamanan aplikasi menjadi salah satu aspek yang sangat penting untuk diperhatikan. Salah satu komponen utama dalam menjaga keamanan aplikasi adalah proses autentikasi, yaitu proses untuk memastikan bahwa pengguna yang mengakses sistem adalah benar-benar pihak yang mereka klaim. Dengan meningkatnya penggunaan aplikasi berbasis *web* dan *mobile*, dibutuhkan metode autentikasi yang aman, efisien, dan mudah diimplementasikan.

Salah satu metode autentikasi yang saat ini banyak digunakan adalah JSON Web Token (JWT). JWT adalah standar terbuka yang memungkinkan proses autentikasi dan otorisasi berbasis token dengan format JSON yang terenkripsi. Teknologi ini memungkinkan aplikasi untuk mengelola akses pengguna dengan lebih aman, karena data pengguna disimpan dalam bentuk token yang dienkripsi. Selain itu, JWT tidak membutuhkan penyimpanan data sesi di *server*, sehingga meningkatkan skalabilitas aplikasi.

Integrasi autentikasi dengan JWT menjadi solusi ideal bagi aplikasi modern yang membutuhkan kecepatan dan keamanan. JWT memungkinkan pengguna untuk melakukan autentikasi satu kali dan kemudian menggunakan *token* tersebut untuk mengakses berbagai layanan tanpa harus melakukan *login* ulang. Hal ini sangat menguntungkan, terutama dalam arsitektur aplikasi berbasis *microservices* atau sistem terdistribusi.

Namun, meskipun JWT memiliki banyak kelebihan, implementasi yang kurang tepat dapat menyebabkan celah keamanan, seperti kebocoran *token* atau penggunaan algoritma enkripsi yang tidak aman. Oleh karena itu, pemahaman yang

baik tentang konsep, penerapan, dan pengelolaan JWT sangat penting bagi pengembang aplikasi.

1.2 Rumusan Masalah

Berikut adalah rumusan masalah yang ada :

1.2.1 Bagaimana cara integrasi autentikasi pada *backend* dengan JWT

1.3 Tujuan

Sesuai dengan latar belakang dan rumusan masalah diatas, beberapa tujuan yang ingin dicapai adalah sebagai berikut,

1.3.1 Mengetahui cara integrasi autentikasi pada *backend* dengan JWT.

1.4 Manfaat

Pengembangan aplikasi ini, diharapkan memberikan manfaat sebagai berikut:

1.4.1 Menambah wawasan dan pemahaman mengenai proses integrasi autentikasi dengan JWT.

1.4.2 Bahan Latihan dalam mengintegrasikan autentikasi pada suatu *project*.

BAB II

TINJAUAN LITERATUR

2.1 Autentikasi

Autentikasi adalah proses verifikasi identitas untuk memastikan bahwa seseorang yang mencoba mengakses sistem atau data adalah pengguna yang sah. Proses ini penting untuk menjaga keamanan sistem dan data dari akses yang tidak sah. Proses autentikasi ini dapat memastikan akses yang sah, mencegah akses yang tidak sah, dan meningkatkan keamanan sistem.

Beberapa teknologi yang digunakan dalam integrasi autentikasi dengan Jason Web Token adalah sebagai berikut :

Tabel 2.1 Teknologi pendukung

No	Teknologi	Keterangan
1	Google Cloud Platform	<i>Platform</i> untuk menyimpan data di <i>cloud</i> .
2	App Engine	Layanan GCP untuk menjalankan <i>web</i> .
3	MySQL	DBMS untuk menyimpan data.
4	Cloud Run	Layanan <i>serverless</i> untuk menjalankan aplikasi dalam.
5	Cloud Build	Layanan GCP untuk <i>build, test</i> , dan <i>deploy app</i> secara otomatis. Bisa dibilang <i>Continuous Integration/Continuous Delivery (CI/CD)</i> .
6	Jason Web Token	Standar terbuka (RFC 7519) untuk membuat token akses.
7	Bcrypt	Algoritma <i>hashing</i> kata sandi yang aman dan kuat.

BAB III

METODOLOGI

3.1 Analisis Permasalahan

Pada *project* yang telah dibuat sebelumnya, semua sumber data atau API dapat diakses oleh semua orang, yang dimana ini akan membuat sumber data tersebut sangat rentan. Sehingga dibutuhkan sebuah sistem untuk melakukan autentikasi atau verifikasi hanya kepada pengguna sah saja yang dapat mengakses API tersebut.

3.2 Perancangan Solusi

Berikut adalah perancangan Solusi untuk masalah – masalah diatas :

3.2.1 Setup Backend

3.2.1.1 Membuat .env

```
DB_NAME = notes_prakcc
DB_USERNAME = root
DB_PASSWORD =
DB_HOST = 34.122.227.61

ACCESS_TOKEN_SECRET = SECRETTOKEN
REFRESH_TOKEN_SECRET = SECRETTOKEN
```

Tabel 3.1 Membuat .env

3.2.1.2 Konfigurasi Database.js

```
import { Sequelize } from "sequelize";
import dotenv from "dotenv";

dotenv.config();

const DB_NAME = process.env.DB_NAME;
const DB_USERNAME = process.env.DB_USERNAME;
const DB_PASSWORD = process.env.DB_PASSWORD;
const DB_HOST = process.env.DB_HOST;
```



```

console.log(DB_NAME,DB_USERNAME,DB_PASSWORD,DB_HOST);

const db = new Sequelize(DB_NAME, DB_USERNAME, DB_PASSWORD, {
  host: DB_HOST,
  dialect: "mysql",
});

export default db;

```

Tabel 3.2 Konfigurasi Database.js

3.2.1.3 Konfigurasi NoteController.js

```

import Note from "../models/NoteModel.js";

// GET NOTE
export const getNote = async(req,res) =>{
  try {
    console.log("req.user: ", req.user.id);
    const id = req.user.id;
    const notes = await Note.findAll({ where: { userId: id } });
    res.status(200).json({
      status: "Success",
      message: "Notes Retrieved",
      data: notes,
    });
  } catch (error) {
    res.status(error.statusCode || 500).json({
      status: "Error",
      message: error.message,
    });
  }
}

// GET NOTE BY ID

```

```

export const getNoteById = async(req,res) =>{
  try {
    const note = await Note.findOne({ where: { id: req.params.id } });
    if (!note) {
      const error = new Error("User tidak ditemukan !");
      error.statusCode = 400;
      throw error;
    }
    res.status(200).json({
      status: "Success",
      message: "Note Retrieved",
      data: note,
    });
  } catch (error) {
    res.status(error.statusCode || 500).json({
      status: "Error",
      message: error.message,
    });
  }
}

// ADD NOTE
export const addNote = async(req,res) => {
  try {
    const { title, content } = req.body;
    const id = req.user.id;

    if (!title || !content) {
      const msg = `${
        !title ? "Title" : "Content"
      } field cannot be empty !`;
      const error = new Error(msg);
      error.statusCode = 401;
    }
  }
}

```

```

        throw error;
    }
    await Note.create({
        userId: id,
        title: title,
        content: content
    });
    res.status(201).json({
        status: "Success",
        message: "Note Created",
    });
} catch (error) {
    res.status(error.statusCode || 500).json({
        status: "Error",
        message: error.message,
    });
}
}

// UPDATE NOTE
export const updateNote = async(req,res) => {
    try {
        const { title, content } = req.body;
        const ifNoteExist = await Note.findOne({ where: { id: req.params.id } });
        if (!title || !content) {
            const msg = `${
                !title ? "Title" : "Content"
            } field cannot be empty !`;
            const error = new Error(msg);
            error.statusCode = 401;
            throw error;
        }
        if (!ifNoteExist) {

```

```

    const error = new Error("Note tidak ditemukan !");
    error.statusCode = 400;
    throw error;
  }
  let updatedData = {title,content};
  await Note.update(updatedData, {
    where: { id: req.params.id },
  });
  res.status(200).json({
    status: "Success",
    message: "Note Updated",
  });
} catch (error) {
  res.status(error.statusCode || 500).json({
    status: "Error",
    message: error.message,
  });
}
}

// DELETE NOTE
export const deleteNote = async(req,res) => {
  try {
    const ifNoteExist = await Note.findOne({ where: { id: req.params.id } });
    if (!ifNoteExist) {
      const error = new Error("Note tidak ditemukan !");
      error.statusCode = 400;
      throw error;
    }

    await Note.destroy({ where: { id: req.params.id } });
    res.status(200).json({
      status: "Success",

```

```

        message: "Note Deleted",
    });
} catch (error) {
    res.status(error.statusCode || 500).json({
        status: "Error",
        message: error.message,
    });
}
}

```

Tabel 3.3 Konfigurasi NoteController.js

3.2.1.4 Membuat UserController.js

```

import User from "../models/UserModel.js";
import bcrypt from "bcrypt";
import jwt from "jsonwebtoken";

// GET USER
export async function getUsers(req, res) {
    try {
        const users = await User.findAll();
        res.status(200).json({
            status: "Success",
            message: "Users Retrieved",
            data: users,
        });
    } catch (error) {
        res.status(error.statusCode || 500).json({
            status: "Error",
            message: error.message,
        });
    }
}

```

```
// GET USER BY ID
export async function getUserById(req, res) {
  try {
    const user = await User.findOne({ where: { id: req.params.id } });
    if (!user) {
      const error = new Error("User tidak ditemukan !");
      error.statusCode = 400;
      throw error;
    }
    res.status(200).json({
      status: "Success",
      message: "User Retrieved",
      data: user,
    });
  } catch (error) {
    res.status(error.statusCode || 500).json({
      status: "Error",
      message: error.message,
    });
  }
}

// ADD USER
export async function addUser(req, res) {
  try {
    const { name, email, password } = req.body;
    if (!name || !email || !password) {
      const msg = `${
        !name ? "Name" : "Email"
      } field cannot be empty !`;
      const error = new Error(msg);
      error.statusCode = 401;
      throw error;
    }
  }
}
```

```

    }

    const encryptedpass = await bcrypt.hash(password,5);
    await User.create({
      name: name,
      email:email,
      password:encryptedpass
    });
    res.status(201).json({
      status: "Success",
      message: "User Created",
    });
  } catch (error) {
    res.status(error.statusCode || 500).json({
      status: "Error",
      message: error.message,
    });
  }
}

// UPDATE USER
export async function updateUser(req, res) {
  try {
    const { name, email, password } = req.body;
    const ifUserExist = await User.findOne({ where: { id: req.params.id } });

    if (!name || !email || !password) {
      const msg = `${
        !name ? "Name" : "Email"
      } field cannot be empty !`;
      const error = new Error(msg);
      error.statusCode = 401;
      throw error;
    }
  }
}

```

```

    if (!ifUserExist) {
      const error = new Error("User tidak ditemukan !");
      error.statusCode = 400;
      throw error;
    }

    const encryptedpass = await bcrypt.hash(password,5);
    let updatedData = { name,email,encryptedpass };

    await User.update(updatedData, {
      where: { id: req.params.id },
    });

    res.status(200).json({
      status: "Success",
      message: "User Updated",
    });
  } catch (error) {
    res.status(error.statusCode || 500).json({
      status: "Error",
      message: error.message,
    });
  }
}

// DELETE USER
export async function deleteUser(req, res) {
  try {
    const ifUserExist = await User.findOne({ where: { id: req.params.id } });
    if (!ifUserExist) {
      const error = new Error("User tidak ditemukan !");
      error.statusCode = 400;

```



```

    throw error;
  }

  await User.destroy({ where: { id: req.params.id } });
  res.status(200).json({
    status: "Success",
    message: "User Deleted",
  });
} catch (error) {
  res.status(error.statusCode || 500).json({
    status: "Error",
    message: error.message,
  });
}
}

// REGISTER HANDLER
export async function registerHandler(req, res) {
  try {
    const { name, email, password } = req.body;
    if (!name || !email || !password) {
      const msg = `${
        !name ? "Name" : "Email"
      } field cannot be empty !`;
      const error = new Error(msg);
      error.statusCode = 401;
      throw error;
    }
    const encryptedpass = await bcrypt.hash(password, 5);
    await User.create({
      name: name,
      email: email,
      password: encryptedpass
    });
  } catch (error) {
    // ...
  }
}

```

```

    });
    res.status(201).json({
      status: "Success",
      message: "User Created",
    });
  } catch (error) {
    res.status(error.statusCode || 500).json({
      status: "Error",
      message: error.message,
    });
  }
}

// LOGIN HANDLER
export async function loginHandler(req, res){
  try{
    const{email, password} = req.body;
    const user = await User.findOne({
      where : {
        email: email
      }
    });

    if(user){
      const userPlain = user.toJSON();
      const { password: _, refresh_token: __, ...safeUserData } = userPlain;

      const decryptPassword = await bcrypt.compare(password, user.password);
      if(decryptPassword){
        const accessToken = jwt.sign(safeUserData,
process.env.ACCESS_TOKEN_SECRET, {
          expiresIn : '12h'
        });
      }
    }
  }
}

```

```

const refreshToken = jwt.sign(safeUserData,
process.env.REFRESH_TOKEN_SECRET, {
  expiresIn : '1d'
});
await User.update({refresh_token:refreshToken},{
  where:{
    id:user.id
  }
});
res.cookie('refreshToken', refreshToken,{
  httpOnly : true,
  sameSite : 'Strict',
  maxAge : 24*60*60*1000,
  secure:true
});
res.status(200).json({
  status: "Succes",
  message: "Login Berhasil",
  safeUserData,
  accessToken
});
}
else{
  res.status(400).json({
    status: "Failed",
    message: "Password atau email salah",

  });
}
} else{
  res.status(400).json({
    status: "Failed",
    message: "Password atau email salah",

```

```

    });
  }
} catch(error){
  res.status(error.statusCode || 500).json({
    status: "error",
    message: error.message
  })
}
}

// LOGOUT
export async function logout(req,res){
  const refreshToken = req.cookies.refreshToken;
  if(!refreshToken) return res.sendStatus(204);
  const user = await User.findOne({
    where:{
      refresh_token:refreshToken
    }
  });
  if(!user.refresh_token) return res.sendStatus(204);
  const userId = user.id;
  await User.update({refresh_token:null},{
    where:{
      id:userId
    }
  });
  res.clearCookie('refreshToken');
  return res.sendStatus(200);
}

```

Tabel 3.4 Membuat UserController.js

3.2.1.5 Membuat RefreshToken.js

```
import User from "../models/UserModel.js";
import jwt from "jsonwebtoken";

export const refreshToken = async(req, res)=>{
  try{
    const refreshToken = req.cookies.refreshToken;
    console.log({refreshToken})
    if(!refreshToken) return res.sendStatus(401);
    console.log("sudah lewat 401 di authcontroller")
    const user = await User.findOne({
      where:{
        refresh_token:refreshToken
      }
    });
    if(!user.refresh_token) return res.sendStatus(403);
    else jwt.verify(refreshToken, process.env.REFRESH_TOKEN_SECRET,(err,
decoded)=>{
      if(err) return res.sendStatus(403);
      console.log("sudah lewat 403 ke dua di controller")
      const userPlain = user.toJSON();
      const { password: _, refresh_token: __, ...safeUserData } = userPlain;
      const accessToken =
jwt.sign(safeUserData, process.env.ACCESS_TOKEN_SECRET,{
      expiresIn: '30s'
    });
    res.json({ accessToken });
  })
  }catch(error){
    console.log(error);
  }
}
```

Tabel 3.5 Membuat RefreshToken.js

3.2.1.6 Membuat VerifyToken.js

```
import jwt from "jsonwebtoken"

export const verifyToken = (req, res, next)=>{
  const authHeader = req.headers['authorization'];
  console.log("Authorization Header:", req.headers['authorization']);
  const token = authHeader && authHeader.split(' ')[1];
  console.log("masuk verify token: ", {token});
  if(token == null) return res.sendStatus(401);
  console.log("sudah lewat 401 di verify");
  jwt.verify(token, process.env.ACCESS_TOKEN_SECRET,(err, decoded)=>{
```

```

    if(err) return res.sendStatus(403);
    console.log("sudah lewat 403 di verify");
    req.user = decoded;
    next();
  })
}

```

Tabel 3.6 Membuat VerifyToken.js

3.2.1.7 Konfigurasi NoteModel.js dan Membuat UserModel.js

```

import { Sequelize } from "sequelize";
import db from "../config/Database.js";

const Note = db.define('notes', {
  userId: Sequelize.INTEGER,
  title: Sequelize.STRING,
  content: Sequelize.STRING,
}, {
  freezeTableName: true
});

db.sync().then(() => console.log("Database synced"));

export default Note;

```

Tabel 3.7 Konfigurasi NoteModel.js

```

import sequelize, { Sequelize } from "sequelize";
import db from "../config/Database.js";

const User = db.define(
  "user",
  {
    name: Sequelize.STRING,
    email: Sequelize.STRING,
    password: Sequelize.STRING,
    refresh_token: Sequelize.TEXT
  }, {
    freezeTableName: true
  }
);

db.sync().then(() => console.log("Database synced"));

export default User;

```

Tabel 3.8 Membuat UserModel.js

3.2.1.8 Konfigurasi NoteRoute.js

```
import express from "express";
import { addNote, deleteNote, getNote, getNoteById, updateNote } from
"../controllers/NoteController.js";
import { addUser, deleteUser, getUsers, getUserById, updateUser, registerHandler,
loginHandler, logout } from "../controllers/UserController.js";
import { refreshToken } from "../controllers/RefreshToken.js";
import { verifyToken } from "../middleware/VerifyToken.js";

const router = express.Router();

router.get("/token", refreshToken);

router.post("/register", registerHandler);
router.post("/login", loginHandler);
router.delete("/logout", logout);

router.get('/users', verifyToken, getUsers);
router.get('/users/:id', verifyToken, getUserById);
router.post('/users', addUser);
router.patch('/users/:id', verifyToken, updateUser);
router.delete('/users/:id', deleteUser);

router.get('/notes', verifyToken, getNote);
router.get('/notes/:id', verifyToken, getNoteById);
router.post('/notes', verifyToken, addNote);
router.patch('/notes/:id', verifyToken, updateNote);
router.delete('/notes/:id', verifyToken, deleteNote);

export default router;
```

Tabel 3.9 Konfigurasi NoteRoute.js

3.2.1.9 Konfigurasi index.js

```
import express from "express";
import cors from "cors";
import noteRoute from "../routes/NoteRoute.js";
import dotenv from "dotenv";
import cookieParser from "cookie-parser";

const app = express();
app.set("view engine", "ejs");

dotenv.config();

app.use(cookieParser());
```

```

app.use(cors({ credentials:true, origin:'http://localhost:3000' }));
app.use(express.json());
app.get("/", (req, res) => res.render("index"));
app.use(noteRoute);

app.listen(5000, () => console.log("Connected to server"));

```

Tabel 3.10 Konfigurasi index.js

3.2.1.10 Konfigurasi App.js

```

import { BrowserRouter, Route, Routes, Navigate } from "react-router-dom";
import AddNote from "./components/AddNote";
import NoteList from "./components/NoteList";
import EditNote from "./components/EditNote";
import DetailNote from "./components/DetailNote";
import SignIn from "./components/SignIn";
import SignUp from "./components/SignUp";
import { AuthProvider, useAuthContext } from "./auth/AuthProvider";

function App() {
  return (
    <AuthProvider>
      <BrowserRouter>
        <AppRoutes />
      </BrowserRouter>
    </AuthProvider>
  );
}

function AppRoutes() {
  const { accessToken } = useAuthContext();
  const isAuthenticated = !!accessToken;

  return (
    <Routes>
      <Route path="/" element={ <SignIn /> } />
      <Route path="/signin" element={ <SignIn /> } />
      <Route path="/signup" element={ <SignUp /> } />
      <Route path="/notes" element={ isAuthenticated ? <NoteList /> : <Navigate
to="/signin" /> } />
      <Route path="/notes/add" element={ isAuthenticated ? <AddNote /> : <Navigate
to="/signin" /> } />
      <Route path="/edit/:id" element={ isAuthenticated ? <EditNote /> : <Navigate
to="/signin" /> } />
      <Route path="/notes/detail/:id" element={ isAuthenticated ? <DetailNote /> :
<Navigate to="/signin" /> } />
    </Routes>
  );
}

```



```
);
}

export default App;
```

Tabel 3.11 Konfigurasi App.js

3.2.2 Setup Frontend

3.2.2.1 Membuat AxiosInstance.js

```
import axios from "axios";
import { BASE_URL } from "../utils";

const instance = axios.create({
  baseURL: BASE_URL,
  withCredentials: true,
});

export default instance;
```

Tabel 3.11 Membuat AxiosInstance.js

3.2.2.2 Membuat AxiosInterceptor.js

```
import { useEffect } from "react";
import axios from "../AxiosInstance.js";
import useAuth from "../auth/UseAuth.js";

const AxiosInterceptor = () => {
  const { accessToken, refreshAccessToken, logout } = useAuth();

  useEffect(() => {
    const requestInterceptor = axios.interceptors.request.use(
      (config) => {
        if (accessToken) {
          config.headers.Authorization = `Bearer ${accessToken}`;
        }
        return config;
      },
      (error) => Promise.reject(error)
    );

    const responseInterceptor = axios.interceptors.response.use(
      (response) => response,
      async (error) => {
        const originalRequest = error.config;
```

```

    if (error.response?.status === 403) {
      const newToken = await refreshAccessToken();

      if (newToken) {
        originalRequest.headers.Authorization = `Bearer ${newToken}`;
        return axios(originalRequest);
      } else {
        logout();
      }
    }

    return Promise.reject(error);
  }
);

return () => {
  axios.interceptors.request.eject(requestInterceptor);
  axios.interceptors.response.eject(responseInterceptor);
};
}, [accessToken, refreshAccessToken, logout]);

return null;
};

export default AxiosInterceptor;

```

Tabel 3.12 Membuat AxiosInterceptor.js

3.2.2.3 Membuat AuthProvider.js

```

import { createContext, useContext, useState } from "react";
import Cookies from "js-cookie";
import axios from "../api/AxiosInstance.js";
import PropTypes from 'prop-types';
import { BASE_URL } from "../utils.js";

const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
  const [accessToken, setAccessToken] = useState(null);

  const login = async (email, password) => {
    try {
      const res = await axios.post(`${BASE_URL}/login`, {email, password}, {
        withCredentials: true
      });
      setAccessToken(res.data.accessToken);
    }
  }
};

```

```

    Cookies.set("refreshToken", res.data.refreshToken, {
      secure: true,
      sameSite: "None",
      path: "/",
      domain: "frontend-notes-176-dot-xenon-axe-450704-n3.uc.r.appspot.com",
      expires: 5,
    });

    return true;
  } catch (err) {
    console.error("Login failed:", err);
    return false;
  }
};

const logout = () => {
  setAccessToken(null);
  Cookies.remove("refreshToken");
};

const refreshAccessToken = async () => {
  try {
    const res = await axios.get(`${BASE_URL}/token`);
    setAccessToken(res.data.accessToken);
    return res.data.accessToken;
  } catch (err) {
    console.error("Token refresh failed:", err);
    logout();
    return "kosong";
  }
};

return (
  <AuthContext.Provider
    value={{ accessToken, login, logout, refreshAccessToken }}
  >
    {children}
  </AuthContext.Provider>
);
};

AuthProvider.propTypes = {
  children: PropTypes.node.isRequired,
};

export const useAuthContext = () => useContext(AuthContext);

```

Tabel 3.13 Membuat AuthProvider.js

3.2.2.4 Membuat UseAuth.js

```
import { useAuthContext } from "../AuthProvider";

const useAuth = () => {
  const { accessToken, login, logout, refreshAccessToken } = useAuthContext();

  return {
    accessToken,
    login,
    logout,
    refreshAccessToken,
    isAuthenticated: !!accessToken,
  };
};

export default useAuth;
```

Tabel 3.14 Membuat UseAuth.js

3.2.2.5 Membuat AddNote.js

```
import React, { useState } from 'react';
import axios from 'axios';
import { useNavigate } from 'react-router-dom';
import { BASE_URL } from "../utils";
import useAuth from '../auth/UseAuth';

function AddNote() {
  const [title, setTitle] = useState("");
  const [content, setContent] = useState("");
  const navigate = useNavigate();
  const { accessToken, refreshAccessToken } = useAuth();

  const saveNote = async (e) => {
    e.preventDefault();
    try {
      await axios.post(`${BASE_URL}/notes`, {
        title,
        content
      }, {
        headers: { Authorization: `Bearer ${accessToken}` }
      });
      navigate("/notes");
    } catch (error) {
      console.error("Failed to add note:", error);
    }
  };
}
```

```

    if (error.response && error.response.status === 401) {
      const newAccessToken = await refreshAccessToken();
      if (newAccessToken !== "kosong") {
        await axios.post(`${BASE_URL}/notes`, {
          title,
          content
        }, {
          headers: {
            Authorization: `Bearer ${newAccessToken}`
          }
        });
        navigate("/notes");
      } else {
        console.error("Failed to refresh token. Redirecting to login.");
        navigate("/signin");
      }
    }
  }
}

return (
  <div className="columns mt-6 is-centered">
    <div className="column is-half">
      <h1 className="title has-text-centered mb-5">Add a New Note</h1>
      <form onSubmit={saveNote} className="box">
        <div className="field">
          <label className="label">Title</label>
          <div className="control">
            <input
              type="text"
              className="input"
              value={title}
              onChange={(e) => setTitle(e.target.value)}
              placeholder="Enter the title of your note"
              required
            />
          </div>
        </div>
        <div className="field">
          <label className="label">Content</label>
          <div className="control">
            <textarea
              className="textarea"
              value={content}
              onChange={(e) => setContent(e.target.value)}
              placeholder="Write your note content here..."
              rows="6"
              required
            />
          </div>
        </div>
      </form>
    </div>
  </div>
)

```

```

        />
      </div>
    </div>
    <div className="field is-grouped is-grouped-right mt-4">
      <div className="control">
        <button type="submit" className="button is-success">
          Save Note
        </button>
      </div>
      <div className="control">
        <button type="button" className="button is-light" onClick={() =>
navigate("/notes")}>
          Cancel
        </button>
      </div>
    </div>
  </form>
</div>
</div>
);
}

export default AddNote;

```

Tabel 3.15 Membuat AddNote.js

3.2.2.6 Konfigurasi DetailNote.js

```

import React, { useState, useEffect } from 'react';
import axios from "axios";
import { Link, useNavigate, useParams } from 'react-router-dom';
import { BASE_URL } from "../utils";
import useAuth from '../auth/UseAuth';

function DetailNote() {
  const [notes, setNotes] = useState({});
  const navigate = useNavigate();
  const { id } = useParams();
  const { accessToken, refreshAccessToken } = useAuth();

  useEffect(() => {
    getNoteById();
  }, []);

  const getNoteById = async () => {
    try {
      const response = await axios.get(`${BASE_URL}/notes/${id}`, {
        headers: {

```

```

    Authorization: `Bearer ${accessToken}`
  }
});
setNotes(response.data.data);
} catch (error) {
  console.error("Failed to fetch notes:", error);

  if (error.response?.status === 401) {
    const newAccessToken = await refreshAccessToken();
    if (newAccessToken !== "kosong") {
      const res = await axios.get(`${BASE_URL}/notes/${id}`, {
        headers: {
          Authorization: `Bearer ${newAccessToken}`
        }
      });
      setNotes(res.data.data);
    } else {
      console.error("Failed to refresh token. Redirect to login.");
    }
  }
}
};

const deleteNote = async (id) => {
  try {
    await axios.delete(`${BASE_URL}/notes/${id}`, {
      headers: {
        Authorization: `Bearer ${accessToken}`
      }
    });
    navigate("/notes");
  } catch (error) {
    console.error("Failed to delete note:", error);

    if (error.response?.status === 401) {
      const newAccessToken = await refreshAccessToken();
      if (newAccessToken !== "kosong") {
        await axios.delete(`${BASE_URL}/notes/${id}`, {
          headers: {
            Authorization: `Bearer ${newAccessToken}`
          }
        });
        navigate("/notes");
      } else {
        console.error("Failed to refresh token. Redirect to login.");
      }
    }
  }
};

```

```

return (
  <div className="section">
    <div className="container">
      <div className="box">
        <h1 className="title is-4 has-text-centered">{notes.title}</h1>

        <div className="content mb-5">
          <p>{notes.content}</p>
        </div>

        <div className="buttons is-centered">
          <Link to={`../../edit/${notes.id}`} className="button is-info">
            Edit
          </Link>
          <button
            onClick={() => deleteNote(notes.id)}
            className="button is-danger"
          >
            Delete
          </button>
          <Link to="/notes" className="button is-light">
            Back to Notes
          </Link>
        </div>
      </div>
    </div>
  </div>
);
}

export default DetailNote;

```

Tabel 3.16 Konfigurasi DetailNote.js

3.2.2.7 Konfigurasi EditNote.js

```

import React, { useState, useEffect } from 'react';
import axios from 'axios';
import { useNavigate, useParams } from 'react-router-dom';
import { BASE_URL } from "../../utils";
import useAuth from '../auth/UseAuth';

function EditNote() {
  const [title, setTitle] = useState("");
  const [content, setContent] = useState("");
  const navigate = useNavigate();
  const { id } = useParams();

```



```

const { accessToken, refreshAccessToken } = useAuth();

useEffect(() => {
  getNoteById();
}, []);

const getNoteById = async () => {
  try {
    const response = await axios.get(`${BASE_URL}/notes/${id}`, {
      headers: {
        Authorization: `Bearer ${accessToken}`
      }
    });
    setTitle(response.data.data.title);
    setContent(response.data.data.content);
  } catch (error) {
    console.error("Failed to fetch note:", error);
    if (error.response?.status === 401) {
      const newAccessToken = await refreshAccessToken();
      if (newAccessToken !== "kosong") {
        const response = await axios.get(`${BASE_URL}/notes/${id}`, {
          headers: {
            Authorization: `Bearer ${newAccessToken}`
          }
        });
        setTitle(response.data.data.title);
        setContent(response.data.data.content);
      } else {
        navigate("/signin");
      }
    }
  }
};

const updateNote = async (e) => {
  e.preventDefault();
  try {
    await axios.patch(`${BASE_URL}/notes/${id}`, {
      title,
      content
    }, {
      headers: { Authorization: `Bearer ${accessToken}` }
    });
    navigate(`/notes/detail/${id}`);
  } catch (error) {
    console.error("Failed to update note:", error);
    if (error.response && error.response.status === 401) {
      const newAccessToken = await refreshAccessToken();
      if (newAccessToken !== "kosong") {

```

```

        await axios.patch(`${BASE_URL}/notes/${id}`, {
            title,
            content
        }, {
            headers: {
                Authorization: `Bearer ${newAccessToken}`
            }
        });
        navigate(`/notes/detail/${id}`);
    } else {
        navigate("/signin");
    }
}
}
};

return (
    <div className="columns mt-6 is-centered">
        <div className="column is-half">
            <h1 className="title has-text-centered mb-5">Edit Note</h1>
            <form onSubmit={updateNote} className="box">
                <div className="field">
                    <label className="label">Title</label>
                    <div className="control">
                        <input
                            type="text"
                            className="input"
                            value={title}
                            onChange={(e) => setTitle(e.target.value)}
                            placeholder="Edit the title"
                            required
                        />
                    </div>
                </div>
                <div className="field">
                    <label className="label">Content</label>
                    <div className="control">
                        <textarea
                            className="textarea"
                            value={content}
                            onChange={(e) => setContent(e.target.value)}
                            placeholder="Edit the content"
                            rows="6"
                            required
                        />
                    </div>
                </div>
                <div className="field is-grouped is-grouped-right mt-4">
                    <div className="control">

```

```

        <button type="submit" className="button is-success">
            Update Note
        </button>
    </div>
    <div className="control">
        <button type="button" className="button is-light" onClick={() =>
navigate(`/notes/detail/${id}`)}>
            Cancel
        </button>
    </div>
</div>
</form>
</div>
</div>
);
}

export default EditNote;

```

Tabel 3.17 Konfigurasi EditNote.js

3.2.2.8 Konfigurasi NoteList.js

```

import React, { useState, useEffect } from 'react';
import axios from "axios";
import { Link, useNavigate } from 'react-router-dom';
import { BASE_URL } from "../utils";
import useAuth from '../auth/UseAuth';

function NoteList() {
    const [notes, setNotes] = useState([]);
    const [searchTerm, setSearchTerm] = useState("");
    const navigate = useNavigate();
    const { accessToken, refreshAccessToken, logout } = useAuth();

    const fetchNotes = async () => {
        try {
            const res = await axios.get(`${BASE_URL}/notes`, {
                headers: {
                    Authorization: `Bearer ${accessToken}`
                }
            });
            setNotes(res.data.data);
        } catch (error) {
            console.error("Failed to fetch notes:", error);

            if (error.response && error.response.status === 401) {
                const newAccessToken = await refreshAccessToken();
            }
        }
    };
}

```

```

    if (newAccessToken !== "kosong") {
      const res = await axios.get(`${BASE_URL}/notes`, {
        headers: {
          Authorization: `Bearer ${newAccessToken}`
        }
      });
      setNotes(res.data.data);
    } else {
      console.error("Failed to refresh token. Redirect to login.");
    }
  }
}
};

const handleLogout = async () => {
  await logout();
  navigate("/signin");
};

const handleSearchChange = (e) => {
  setSearchTerm(e.target.value);
};

const filteredNotes = notes.filter(note =>
  note.title.toLowerCase().includes(searchTerm.toLowerCase())
);

useEffect(() => {
  fetchNotes();
}, []);

return (
  <div>
    { /* Navbar */ }
    <nav className="navbar is-dark-grey" role="navigation" aria-label="main navigation">
      <div className="navbar-brand">
        <span className="navbar-item has-text-weight-bold is-size-5">Notes</span>
      </div>
      <div className="navbar-end">
        <div className="navbar-item">
          <button onClick={handleLogout} className="button is-danger is-danger">
            Logout
          </button>
        </div>
      </div>
    </nav>

    { /* Content */ }
  </div>
);

```

```

<div className="section">
  <div className="container">
    <div className="box">

      { /* Search and Add Button */ }
      <div className="level mb-4">
        <div className="level-left">
          <div className="field">
            <div className="control has-icons-left">
              <input
                className="input"
                type="text"
                placeholder="Search notes..."
                value={searchTerm}
                onChange={handleSearchChange}
              />
              <span className="icon is-left">
                <i className="fas fa-search"></i>
              </span>
            </div>
          </div>
          <div className="level-right">
            <Link to={ 'add' } className="button is-info">+ Add Note</Link>
          </div>
        </div>

      { /* Table */ }
      <table className="table is-striped is-fullwidth is-hoverable">
        <thead>
          <tr>
            <th>Title</th>
            <th style={ { width: "100px" } }>Action</th>
          </tr>
        </thead>
        <tbody>
          { filteredNotes.length === 0 ? (
            <tr>
              <td colspan="2" className="has-text-centered has-text-grey">
                No notes found.
              </td>
            </tr>
          ) : (
            filteredNotes.map((note) => (
              <tr key={ note.id }>
                <td>{ note.title }</td>
                <td>
                  <Link to={ `detail/${note.id}` } className="button is-info is-small">See</Link>
                </td>
              </tr>
            )
          )
        </tbody>
      </table>
    </div>
  </div>
</div>

```

```

        </td>
      </tr>
    ))
  })
</tbody>
</table>
</div>
</div>
</div>
</div>
);
}

export default NoteList;

```

Tabel 3.18 Konfigurasi NoteList.js

3.2.2.9 Membuat SignIn.js

```

import React, { useState } from "react";
import { useNavigate, Link } from "react-router-dom";
import useAuth from "../auth/UseAuth.js";

const SignIn = () => {
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const navigate = useNavigate();
  const { login } = useAuth();
  const [error, setError] = useState("");

  const handleLogin = async (e) => {
    e.preventDefault();
    setError("");

    // Logic untuk autentikasi login
    console.log("Login with", { email, password });
    try {
      const result = await login(email, password); // simpan token ke context & cookie
      if (result) {
        navigate("/notes"); // redirect setelah login
      } else {
        alert(result);
      }
    }

    } catch (error) {
      // Log error untuk debug
      console.error("Login Error:", error.response ? error.response.data : error.message);
    }
  }

```

```

    alert("Login failed: " + (error.response ? error.response.data.message :
error.message));
  }
};

return (
  <div
    className="is-flex is-justify-content-center is-align-items-center"
    style={{ height: "100vh", backgroundColor: "#1a1a1a", color: "white" }}
  >
    <div className="box" style={{ width: "320px", backgroundColor: "#2c2c2c" }}>
      <h2 className="title has-text-white has-text-centered">Sign In</h2>
      <input
        className="input mb-3"
        type="text"
        placeholder="Email"
        value={email}
        onChange={(e) => setEmail(e.target.value)}
      />
      <input
        className="input mb-4"
        type="password"
        placeholder="Password"
        value={password}
        onChange={(e) => setPassword(e.target.value)}
      />
      <button className="button is-primary is-fullwidth mb-3"
onClick={handleLogin}>
        Login
      </button>
      <p className="has-text-centered">
        Belum Punya Akun?{" "}
        <Link to="/signup" className="has-text-link">
          Daftar
        </Link>
      </p>
    </div>
  </div>
);
};

export default SignIn;

```

Tabel 3.19 Membuat SignIn.js

3.2.2.10 Membuat SignUp.js

```
import React, { useState } from "react";
```

```

import { useNavigate, Link } from "react-router-dom";
import axios from "axios";
import { BASE_URL } from "../utils";

const SignUp = () => {
  const [name, setName] = useState("");
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const navigate = useNavigate();

  const handleSignUp = async () => {
    try {
      const response = await axios.post(`${BASE_URL}/register`, { name, email, password });
      console.log("Registration Response:", response.data);
      navigate("/signin");
    } catch (error) {
      alert("Registration failed. Please try again.");
    }
  };

  return (
    <div
      className="is-flex is-justify-content-center is-align-items-center"
      style={{ height: "100vh", backgroundColor: "#1a1a1a", color: "white" }}
    >
      <div className="box" style={{ width: "384px", backgroundColor: "#2c2c2c" }}>
        <h2 className="title has-text-white has-text-centered">Sign Up</h2>
        <input
          className="input mb-3"
          type="text"
          placeholder="Name"
          value={name}
          onChange={(e) => setName(e.target.value)}
        />
        <input
          className="input mb-3"
          type="email"
          placeholder="Email"
          value={email}
          onChange={(e) => setEmail(e.target.value)}
        />
        <input
          className="input mb-4"
          type="password"
          placeholder="Password"
          value={password}
          onChange={(e) => setPassword(e.target.value)}
        />
      </div>
    </div>
  );
};

```



```

    <button className="button is-success is-fullwidth mb-3"
onClick={handleSignUp}>
    Register
  </button>
  <p className="has-text-centered">
    Sudah Punya Akun?{" "}
    <Link to="/signin" className="has-text-link">
    Sign In
  </Link>
  </p>
</div>
</div>
);
};

export default SignUp;

```

Tabel 3.20 Membuat SignUp.js

3.2.2.11 Konfigurasi App.js

```

import { BrowserRouter, Route, Routes, Navigate } from "react-router-dom";
import AddNote from "../components/AddNote";
import NoteList from "../components/NoteList";
import EditNote from "../components/EditNote";
import DetailNote from "../components/DetailNote";
import SignIn from "../components/SignIn";
import SignUp from "../components/SignUp";
import React from "react";
import { AuthProvider } from "../auth/AuthProvider";

function App() {
  const isAuthenticated = !!localStorage.getItem("token");

  return (
    <AuthProvider>
    <BrowserRouter>
    <Routes>
      <Route path="/" element={<SignIn />} />
      <Route path="/signin" element={<SignIn />} />
      <Route path="/signup" element={<SignUp />} />
      <Route path="/notes" element={isAuthenticated ? <NoteList /> : <Navigate
to="/signin" />} />
      <Route path="/notes/add" element={isAuthenticated ? <AddNote /> : <Navigate
to="/signin" />} />
      <Route path="/edit/:id" element={isAuthenticated ? <EditNote /> : <Navigate
to="/signin" />} />
    </Routes>
    </BrowserRouter>
  </AuthProvider>
  );
}

```

```

    <Route path="/notes/detail/:id" element={isAuthenticated ? <DetailNote /> :
<Navigate to="/signin" />} />
  </Routes>
</BrowserRouter>
</AuthProvider>
);
}

export default App;

```

Tabel 3.21 Konfigurasi App.js

3.2.3 Deployment

3.2.3.1 Membuat Database MySQL di Cloud

Create a database

Database Name *

Must follow the MySQL identifier rules. [Learn more](#)

Character set *

Can be changed later by executing an ALTER DATABASE query.

Collation

Can be changed later by executing an ALTER DATABASE query.

Gambar 3.1 Membuat *database* MySQL di cloud

3.2.3.2 Push Project ke GitHub

```
PS D:\Kuliah\Semester 6\Prak. Cloud Computing\Tugas 7> git init
```

Gambar 3.2 Inisialisasi git

```
PS D:\Kuliah\Semester 6\Prak. Cloud Computing\Tugas 7> git remote add origin git@github.com:irsyadkk/Prak-TCC-tugas-7.git
```

Gambar 3.3 *Connect git remote*

```
PS D:\Kuliah\Semester 6\Prak. Cloud Computing\Tugas 7> git add .
```

Gambar 3.4 *Add file*

```
PS D:\Kuliah\Semester 6\Prak. Cloud Computing\Tugas 7> git commit -m "first commit"
```

Gambar 3.5 *Commit*

```
PS D:\Kuliah\Semester 6\Prak. Cloud Computing\Tugas 7> git push origin master
```

Gambar 3.6 *Push*

3.2.3.3 Buat Trigger Cloud Build

[←](#) Create trigger

Name *

tugas7-be-irsyad

Must be unique within the project's region

Region *

us-central1 (Iowa)

Description

Tags

Event

Repository event that invokes trigger

☒ Push to a branch

☐ Push new tag

☐ Pull request

Not available for Cloud Source Repositories

Or in response to

☐ Manual invocation

☐ Pub/Sub message

☐ Webhook event

Source

Repository generation

☒ 1st gen

☐ 2nd gen

Repository *

irsyadkk/Prak-TCC-tugas-7 (GitHub App)

Select the repository to watch for events and clone when the trigger is invoked

Branch *

*master\$

Trigger only for a branch that matches the given regular expression [Learn more](#)

☐ Invert Regex

Matches the branch: master

Included files filter (glob)

backend/** X glob pattern example: src/**

Changes affecting at least one included file will trigger builds

Ignored files filter (glob)

Changes only affecting ignored files won't trigger builds

[^ Hide included and ignored files filters](#)

Configuration

Type

☐ Autodetected

A cloudbuild.yaml or Dockerfile will be detected in the repository

☒ Cloud Build configuration file (yaml or json)

Gambar 3.7 Trigger backend

Location

- ☒ **Repository**
insyadkk/Prak-TCC-tugas-7 (GitHub App)
- ☐ **Inline**
Write inline YAML

Cloud Build configuration file location *
/ cloudbuild.backend.yaml

Specify the path to a Cloud Build configuration file in the Git repo. [Learn more](#)

Advanced

Substitution variables

Substitutions allow re-use of a cloudbuild.yaml file with different variable values. Use bash string manipulation to combine variables and bindings to access arbitrary data in the JSON payload of the webhook. [Learn more](#)

Variable 1 * _DB_NAME	Value 1 notes_prakcc
Variable 2 * _DB_USERNAME	Value 2 root
Variable 3 * _DB_PASSWORD	Value 3
Variable 4 * _DB_HOST	Value 4 34.122.227.61
Variable 5 * _ACCESS_TOKEN_SECRET	Value 5 SECRETTOKEN
Variable 6 * _REFRESH_TOKEN_SECRET	Value 6 SECRETTOKEN

[+ Add variable](#)

Approval

Builds created by this trigger will require approval before they execute. Any user with a Cloud Build Approver role for the project can approve a build. [Learn more](#)

☐ Require approval before build executes

Build logs

Build logs will be visible to any GitHub user with read access to this repository.

☐ Send build logs to GitHub

Service account

Select a user-managed service account to use when executing a build with this trigger. [Learn more](#)

Service account *
559917148272-compute@developer.gserviceaccount.com



This service account may have very broad permissions by default. We strongly recommend selecting a service account with only the necessary permissions for this build's execution.

[Learn more](#)

Gambar 3.8 Lanjutan *trigger backend*

[←](#) Create trigger

Name *
tugas7-fe-irsyad
Must be unique within the project's region

Region *
us-central1 (Iowa) ▼

Description

Tags [?](#)

Event

Repository event that invokes trigger

- ☒ Push to a branch
- ☐ Push new tag
- ☐ Pull request
Not available for Cloud Source Repositories

Or in response to

- ☐ Manual invocation
- ☐ Pub/Sub message
- ☐ Webhook event

Source

Repository generation

- ☒ 1st gen
- ☐ 2nd gen

Repository *
irsyadkk/Prak-TCC-tugas-7 (GitHub App) ▼
Select the repository to watch for events and clone when the trigger is invoked

Branch *
^master\$
Trigger only for a branch that matches the given regular expression [Learn more](#)

☐ Invert Regex

Matches the branch: master

Included files filter (glob)
frontend/** ✕ glob pattern example: src/**
Changes affecting at least one included file will trigger builds

Ignored files filter (glob)
Changes only affecting ignored files won't trigger builds

[^](#) Hide included and ignored files filters

Configuration

Type

- ☐ Autodetected
A cloudbuild.yaml or Dockerfile will be detected in the repository
- ☒ Cloud Build configuration file (yaml or json)

Gambar 3.9 *Trigger frontend*

Location

☒ **Repository**
insyadik/Prak-TOC-tugas-7 (GitHub App)

☐ **Inline**
Write inline YAML

Cloud Build configuration file location *

/ cloudbuild.frontend.yaml

Specify the path to a Cloud Build configuration file in the Git repo. [Learn more](#)

Advanced

Substitution variables

Substitutions allow re-use of a cloudbuild.yaml file with different variable values. Use bash string manipulation to combine variables and bindings to access arbitrary data in the JSON payload of the webhook. [Learn more](#)

[+ Add variable](#)

Approval

Builds created by this trigger will require approval before they execute. Any user with a Cloud Build Approver role for the project can approve a build. [Learn more](#)

☐ Require approval before build executes

Build logs

Build logs will be visible to any GitHub user with read access to this repository.


☐ Send build logs to GitHub

Service account

Select a user-managed service account to use when executing a build with this trigger. [Learn more](#)

Service account *

559917148272-compute@developer.gserviceaccount.com

 This service account may have very broad permissions by default. We strongly recommend selecting a service account with only the necessary permissions for this build's execution.

[Learn more](#)

[Create](#) [Cancel](#)

Gambar 3.10 Lanjutan *trigger frontend*

3.2.3.4 Run Trigger



✕

Run trigger

Trigger

tugas7-be-irsyad

Repository

 [irsyadkk/Prak-TCC-tugas-7](#) 

Revision type


☒ Branch

☐ Commit hash

Branch

master

Substitution variables


Substitutions allow re-use of a cloudbuild.yaml file with different variable values. Use bash string manipulation to combine variables and bindings to access arbitrary data in the JSON payload of the webhook. [Learn more](#) 

Variable 1 *

_ACCESS_TOKEN_SECRET

Value 1

SECRETTOKEN




Variable 2 *

_DB_HOST

Value 2

34.122.227.61




Variable 3 *

_DB_NAME

Value 3


notes_prakcc



Variable 4 *

_DB_PASSWORD

Value 4




Variable 5 *

_DB_USERNAME

Value 5

root




Variable 6 *

_REFRESH_TOKEN_SECRET

Value 6


SECRETTOKEN



Gambar 3.11 Run trigger backend

✕
Run trigger

Trigger
tugas7-fe-irsyad

Repository
 [irsyadkk/Prak-TCC-tugas-7](#)

Revision type
☒ Branch
☐ Commit hash

Branch

Run trigger Cancel

Gambar 3.12 *Run trigger frontend*

3.2.3.5 Konfigurasi *Link Frontend*

✓ backendnotes-176 Region: us-central1 URL: <https://backendnotes-176-559917148272.us-central1.run.app> Scaling: Auto (Min: 0)

Gambar 3.13 *Link backend*

```
import axios from "axios";
import { BASE_URL } from "../utils";

const instance = axios.create({
  baseURL: BASE_URL,
  withCredentials: true,
});

export default instance;
```

Tabel 3.22 Konfigurasi AxiosInstance.js

```
export const BASE_URL = "https://backendnotes-176-559917148272.us-central1.run.app";
```

Tabel 3.23 Konfigurasi utils.js

3.2.3.6 Konfigurasi *Link Backend*

```
import express from "express";
import cors from "cors";
import noteRoute from "../routes/NoteRoute.js";
import dotenv from "dotenv";
import cookieParser from "cookie-parser";

const app = express();
app.set("view engine", "ejs");

dotenv.config();

app.use(cookieParser());
app.use(cors({
  origin: 'https://frontend-notes-176-dot-xenon-axe-450704-n3.uc.r.appspot.com',
  credentials: true,
}));
app.use(express.json());
app.get("/", (req, res) => res.render("index"));
app.use(noteRoute);

app.listen(5000, () => console.log("Connected to server"));
```

Tabel 3.24 Konfigurasi index.js

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil

Berikut adalah hasil Integrasi autentikasi dengan Jason Web Token.

4.1.1 Struktur *Database*

Pada *project* ini *database* yang digunakan bukan *database* lokal pada perangkat, melainkan *database* SQL milik *google cloud*. Sehingga semua orang yang tidak berada di jaringan lokal dapat mengaksesnya. Untuk melihat struktur *database*, dapat menggunakan *shell* dan *connect* ke SQL. Setelah itu dapat memilih *database* dan menggunakan *command* “describe” untuk menampilkan struktur *table*.

```
mysql> show tables;
+-----+
| Tables_in_notes_prakcc |
+-----+
| notes                  |
| user                   |
+-----+
2 rows in set (0.21 sec)
```

Gambar 4.1 List Tables

```
mysql> describe notes;
+-----+-----+-----+-----+-----+-----+
| Field      | Type          | Null | Key | Default | Extra           |
+-----+-----+-----+-----+-----+-----+
| id         | int           | NO   | PRI | NULL    | auto_increment |
| userId     | int           | YES  |     | NULL    |                 |
| title      | varchar(255)  | YES  |     | NULL    |                 |
| content    | varchar(255)  | YES  |     | NULL    |                 |
| createdAt  | datetime      | NO   |     | NULL    |                 |
| updatedAt  | datetime      | NO   |     | NULL    |                 |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.24 sec)
```

Gambar 4.2 Struktur Table Notes

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(255)	YES		NULL	
email	varchar(255)	YES		NULL	
password	varchar(255)	YES		NULL	
refresh_token	text	YES		NULL	
createdAt	datetime	NO		NULL	
updatedAt	datetime	NO		NULL	

7 rows in set (0.21 sec)

Gambar 4.3 Struktur *Table User*

4.1.2 API

API yang diintegrasikan dengan autentikasi membutuhkan *access token* untuk mengaksesnya. Sehingga tidak sembarang orang yang bisa mengakses data atau API.

```

1 HTTP/1.1 401 Unauthorized
2 X-Powered-By: Express
3 Access-Control-Allow-Origin: https://frontend-notes-176-dot-xenon-axe-450704-n3.uc.r.appspot.com
4 Vary: Origin
5 Access-Control-Allow-Credentials: true
6 Content-Type: text/plain; charset=utf-8
7 Content-Length: 12
8 ETag: W/"c-dAuDFQrdjs3hezqxDTNgW7A0lYk"
9 Date: Mon, 19 May 2025 11:47:52 GMT
10 Connection: close
11
12 Unauthorized

```

Gambar 4.4 Pengguna Mengambil Data dari API Tanpa *Token*

```

e-450704-n3.uc.r.appspot.com
Vary: Origin
Access-Control-Allow-Credentials: true
Content-Type: application/json; charset=utf-8
Content-Length: 208
ETag: W/"d0-iuNFW18TaFBXjo06qtsM82F30m8"
Date: Mon, 19 May 2025 11:49:36 GMT
Connection: close

{
  "status": "Success",
  "message": "Notes Retrieved",
  "data": [
    {
      "id": 6,
      "userId": 5,
      "title": "Hi",
      "content": "Hello with authentication !",
      "createdAt": "2025-05-19T11:40:34.000Z",
      "updatedAt": "2025-05-19T11:40:34.000Z"
    }
  ]
}

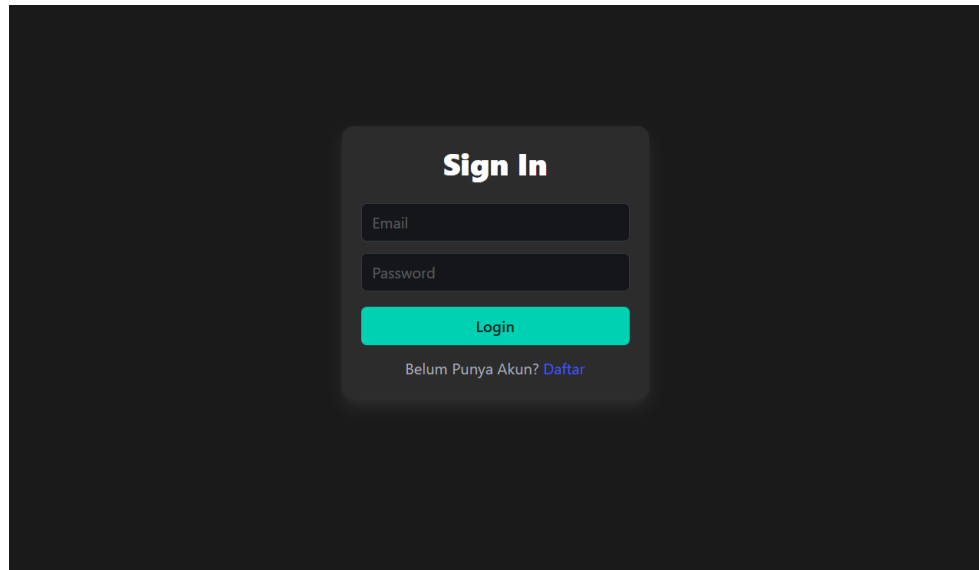
```

Gambar 4.5 Pengguna Mengambil Data dari API Dengan *Token*

4.1.3 Tampilan Web

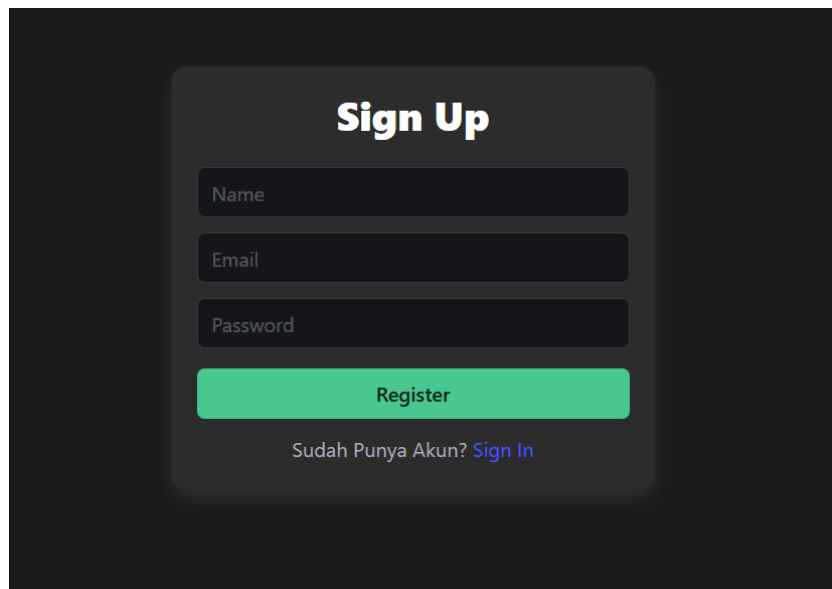
Web yang sudah dirun di *app engine* memiliki beberapa *page*. Yang dimana *page – page* tersebut memiliki fungsinya masing – masing. *Page -page* tersebut antara lain adalah *page* signin, *page* signup, *page* listnotes, *page* addnote, *page* detailnote, dan *page* editnote.

Page sign in berfungsi untuk login pengguna ke dalam sistem, sehingga pengguna dapat mengakses dashboard atau *page* berikutnya. Pengguna juga harus login ketika ingin memanipulasi data, seperti menambahkan, mengedit, melihat, dan menghapus *note*. *Page* ini memiliki 2 *input text* untuk mengisi email dan *password* serta 1 tombol untuk *login* dan ada *link* yang menuju ke *page* pendaftaran atau *sign up*.



Gambar 4.6 *Sign In Page*

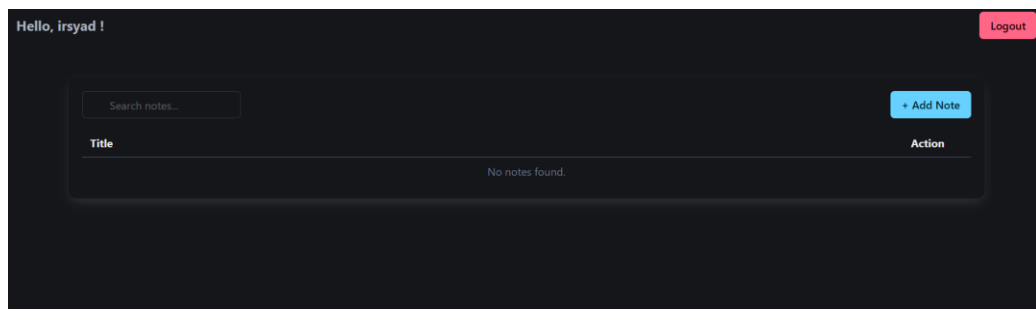
Page sign up berfungsi untuk mendaftarkan pengguna baru, supaya mereka terdaftar di dalam sistem. *Page* ini memiliki 3 *text input* yaitu nama, email, dan *password*. Serta terdapat 1 tombol untuk *register*. Yang terakhir ada *link* yang menuju ke *login* ketika pengguna ingin Kembali ke *login page*.



Gambar 4.7 *Sign Up Page*

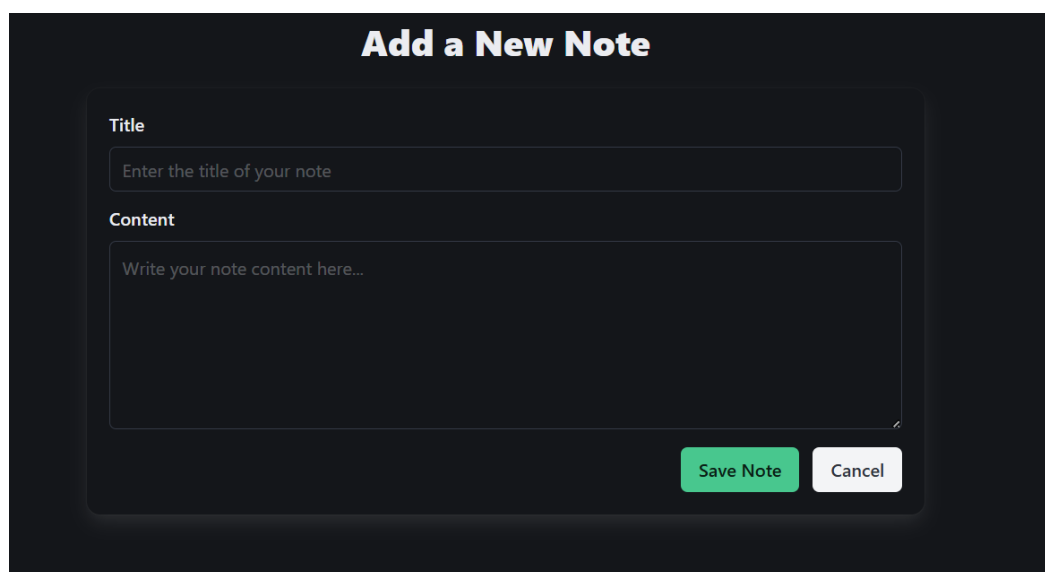
Page listnote berfungsi untuk melihat semua *notes* yang ada. Pada *page* listnote terdapat *navbar* yang berisikan kata “Hello, (nama pengguna) !” pada bagian kiri *navbar*. Terdapat juga tombol *logout* pada bagian kanan *navbar* yang

berfungsi untuk keluar dari akun tersebut dan akan diarahkan ke *page login*. Selanjutnya pada *page listnote* terdapat 1 *table* yang berisikan semua *list notes* yang ada di dalam *database*. Pada *table* tersebut terdapat 2 kolom, yaitu kolom “Title” untuk menampilkan judul *notes* dan kolom “Action” yang memiliki komponen *button* untuk melihat *notes* secara *detail*. Terdapat juga *button* untuk melakukan penambahan *note* pada bagian atas *table*. *Table* tersebut juga dilengkapi dengan *search bar* untuk mencari *notes* dengan nama.



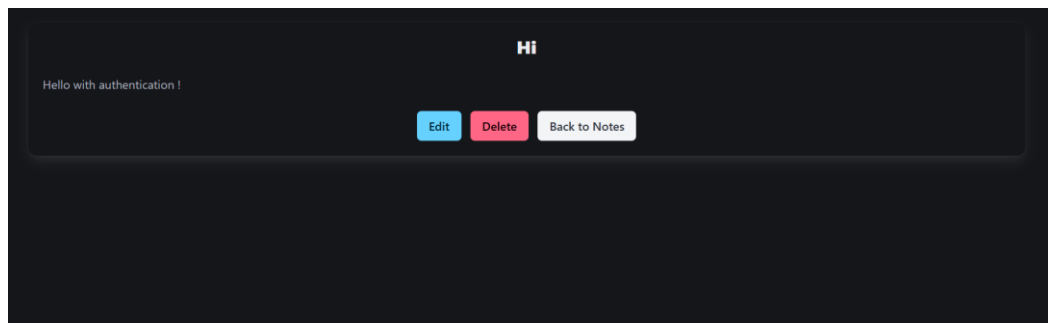
Gambar 4.8 *List Notes Page*

Page addnote berfungsi untuk menambahkann *note* ke dalam *database*. Pada *page addnote* terdapat 2 *textfield* untuk mengisi judul dan konten dari *note* yang ingin dibuat. Terdapat juga 1 *button* untuk melakukan *save* sehingga *textfield* yang sudah diisi akan masuk ke dalam *database*. Serta *button cancel* untuk Kembali ke *page listnote*.



Gambar 4.9 *Add Note Page*

Page detailnote berfungsi untuk melihat *detail* atau konten dari sebuah *note*. Yang sebelumnya pada *page listnote* hanya terdapat judulnya saja, pada *page* ini terdapat judul dan konten dari *note* tersebut. Pada *page* ini terdapat 3 *button* yang dimana setiap *button* tersebut memiliki fungsinya masing – masing. *Button edit* memiliki fungsi untuk mengubah *note* yang sedang diinspeksi sekarang. *Button delete* berfungsi untuk menghapus *note* yang sedang diinspeksi sekarang. Lalu *button Back To Notes* untuk kembail ke *page listnote*.



Gambar 4.10 *Detail Note page*

Page editnote berfungsi untuk mengubah *note* yang terdapat di dalam *database*. Pada *page* ini terdapat 2 *textfield* untuk mengubah judul dan konten *note*. Terdapat juga 1 *button* untuk melakukan *update* pada *database* atau menyimpan *data* yang sudah diinputkann pada *textfield*. Serta *button cancel* untuk kembali ke *page detailnote*.

Gambarr 4.11 *Edit Note Page*

4.2 Pembahasan

Berdasarkan hasil integrasi autentikasi, serta percobaan fitur yang telah dilakukan, terlihat bahwa semua fitur bekerja dengan baik, tanpa mengubah fungsi dari setiap fitur.

Meskipun API sudah dilengkapi dengan autentikasi. Masih terdapat banyak kekurangan yang harus di evaluasi. Beberapa diantaranya adalah pengoptimalan kinerja *web* dan pengembangan UI. *Web* yang ada saat ini masih belum berkerja secara optimal, masih terasa sekali *delay* yang cukup lama untuk melakukan suatu operasi, sehingga diperlukan adanya pengoptimalan baik dari segi *logic source code*, ataupun *service* yang digunakan. UI yang ada saat ini juga masih sederhana, sehingga dapat dilakukan pengembangan lebih lanjut.

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil integrasi autentikasi dengan Jason Web Token yang telah dilakukan, dapat disimpulkan bahwa hal tersebut sudah berhasil dilakukan. Pengintegrasian serta *pendeployan* sudah dilakukan secara keseluruhan tanpa mengubah setiap fungsi yang terdapat didalamnya.

Meskipun pengintegrasian ini sudah memenuhi tujuan awal, masih terdapat banyak evaluasi yang perlu dilakukan, seperti pengoptimalan lebih lanjut dan pengembangan UI.

5.2 Saran

Berdasarkan pembahasan yang telah dilakukan pada BAB – BAB sebelumnya. Terdapat saran yang dapat dilakukan untuk pengembangan lebih lanjut:

5.2.1 Pengoptimalan Lebih Lanjut

Web yang ada saat ini masih belum berkerja secara optimal, masih terasa sekali *delay* yang cukup lama untuk melakukan suatu operasi, sehingga diperlukan adanya pengoptimalan baik dari segi *logic source code*, ataupun *service* yang digunakan.

5.2.2 Pengembangan UI

UI yang ada saat ini sangatlah sederhana, dapat dilakukan pengembangan lebih lanjut untuk meningkatkan kepuasan pengguna. Dengan dilakukannya saran – saran diatas, aplikasi akan menjadi lebih baik dan harapannya dapat berguna bagi para pengguna aplikasi *notes* ini.

DAFTAR PUSTAKA

Wijdan. Dokuemntasi Prak CC Tugas 7 - Video Yapping [Video]. Youtube
<https://www.youtube.com/watch?v=3u4VIWtvvIo&t=1s>

Link Project :

<https://github.com/irsyadkk/Prak-TCC-tugas-7.git> (Github)