

NIM : 3312411077

Nama : Anisa Frity Amelia

Kelas : IF3A Web Pagi

Link Github : <https://github.com/anisafrityamelia/praktikum8.git>

Praktikum 9 - Matakuliah Pilihan 1 (Web)

Program Studi: Teknik Informatika

Lakukan praktikum dibawah ini, dan buat screenshot untuk pembuktian mengerjakan setiap poin dengan mengisi tabel dibawah, kemudian tunjukkan hasil akhir dari men-share repository github yang telah dibuat.

A. Membuat JSON Web Token (Dynamic Bearer Token)

1. Lanjutkan Project Praktikum 8-9, dengan menggunakan file yang sama (copy)
2. Install library JWT **npm install jsonwebtoken bcryptjs**
3. Tambahkan file [auth.controller.js](#), [auth.middleware.js](#), dan [auth.routes.js](#)
4. Buat file .env disamping [server.js](#) (root folder)
Isi file .env dengan variable sebagai berikut:
JWT_SECRET="KUNCI-RAHASIA"
JWT_EXPIRE=1d
5. Tambahkan script berikut di server.js **require('dotenv').config();**
6. Revisi model sebelumnya pada [user.model.js](#) dengan menambahkan fungsi baru seperti berikut, tambahkan findByEmail

```
delete: (id, callback) => {  
  db.query('DELETE FROM users WHERE id = ?', [id], callback);  
},  
  
// Get user by Email (untuk login)  
findByEmail: (email, callback) => {  
  db.query('SELECT * FROM users WHERE email = ?', [email], callback);  
},  
};
```

7. Masukkan script berikut pada [auth.controller.js](#) yang telah dibuat

```
JS auth.controller.js U X
controllers > JS auth.controller.js > login > login > User.findById() callback
1  const User = require('../models/user.model');
2  const bcrypt = require('bcryptjs');
3  const jwt = require('jsonwebtoken');
4
5  exports.login = (req, res) => {
6    const { email, password } = req.body;
7
8    User.findById(email, (err, results) => {
9      if (err) return res.status(500).json({ message: err.message });
10     if (results.length === 0) return res.status(404).json({ message: "User not found" });
11
12     const user = results[0];
13
14     const match = bcrypt.compareSync(password, user.password);
15     if (!match) return res.status(400).json({ message: "Wrong password" });
16
17     const token = jwt.sign(
18       { id: user.id, email: user.email },
19       process.env.JWT_SECRET,
20       { expiresIn: "7d" }
21     );
22
23     res.json({
24       message: "Login success",
25       token,
26       user: { id: user.id, name: user.name, email: user.email }
27     });
28   });
29 };
30
```

8. Ubah [auth.middleware.js](#) yang sebelumnya menggunakan token biasa, menjadi json web token seperti gambar dibawah ini

```
h.controller.js U    JS user.model.js M    JS auth.middlewares.js M X
middlewares > JS auth.middlewares.js > ...
const jwt = require("jsonwebtoken");
const User = require("../models/user.model");

module.exports = (req, res, next) => {
  const header = req.headers.authorization;

  if (!header || !header.startsWith("Bearer ")) {
    return res.status(401).json({ message: "Unauthorized" });
  }

  const token = header.split(" ")[1];

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);

    // Optional: cek user masih ada
    User.getById(decoded.id, (err, results) => {
      if (err) return res.status(500).json({ message: err.message });
      if (results.length === 0) {
        return res.status(401).json({ message: "Invalid token user" });
      }

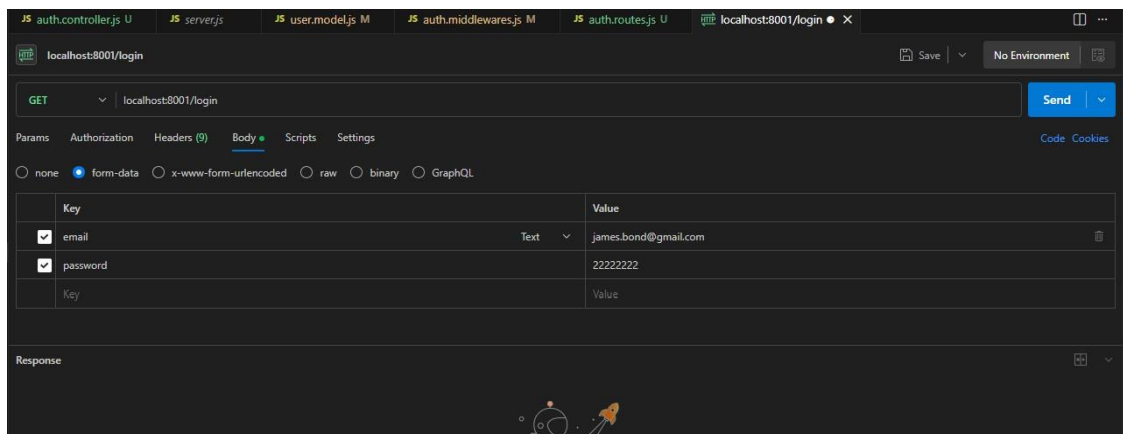
      req.user = results[0];
      next();
    });
  } catch (err) {
    return res.status(401).json({ message: "Invalid token" });
  }
};
```

9. Tambahkan Routes untuk mengakses login pada auth.routes.js

```
JS auth.controller.js U JS user.model.js M JS auth.middlewares.js M JS auth.routes.js U X
routes > JS auth.routes.js > ...
1  const express = require("express");
2  const router = express.Router();
3  const authController = require("../controllers/auth.controller");
4
5  router.post("/login", authController.login);
6
7  module.exports = router;
```

B. Gunakan POSTMAN dapatkan Token BEARER

1. Install postman di visual code, dan lakukan login berdasarkan email dan password yang terdaftar di database
2. Dapatkan bearer dengan memanggil API endpoints /login

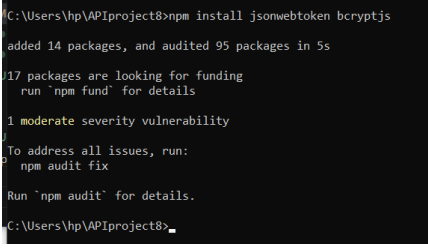
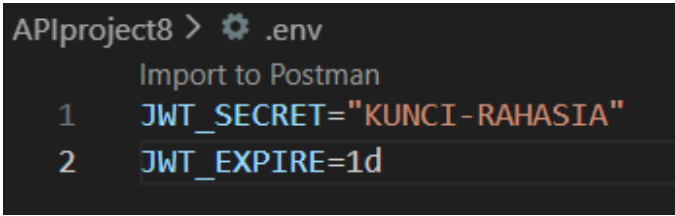
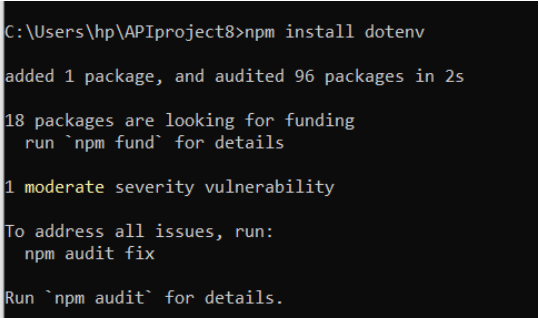


3. Catat bearer yang di dapatkan, lalu gunakan bearer tersebut untuk memanggil endpoints lainnya yang pada praktikum 9 telah di proteksi.

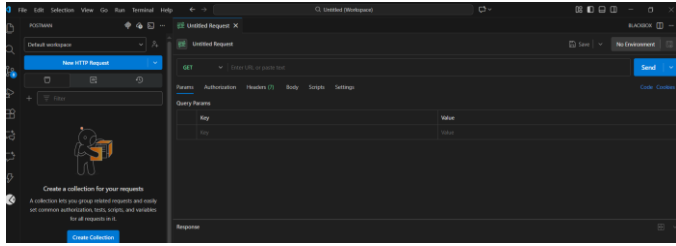
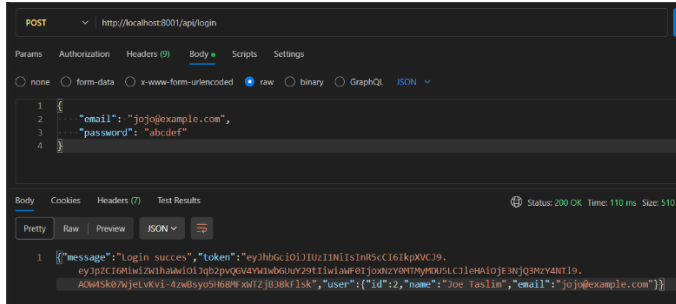
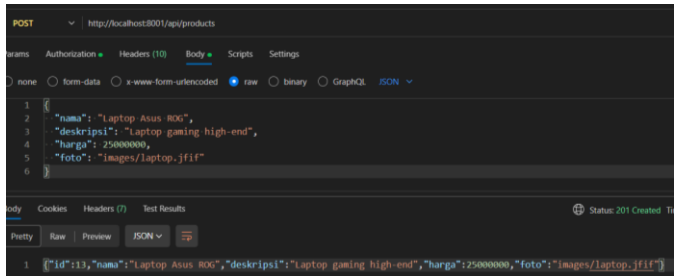
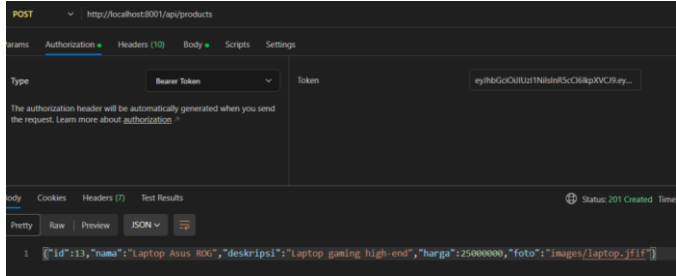
C. Github + Visual Code

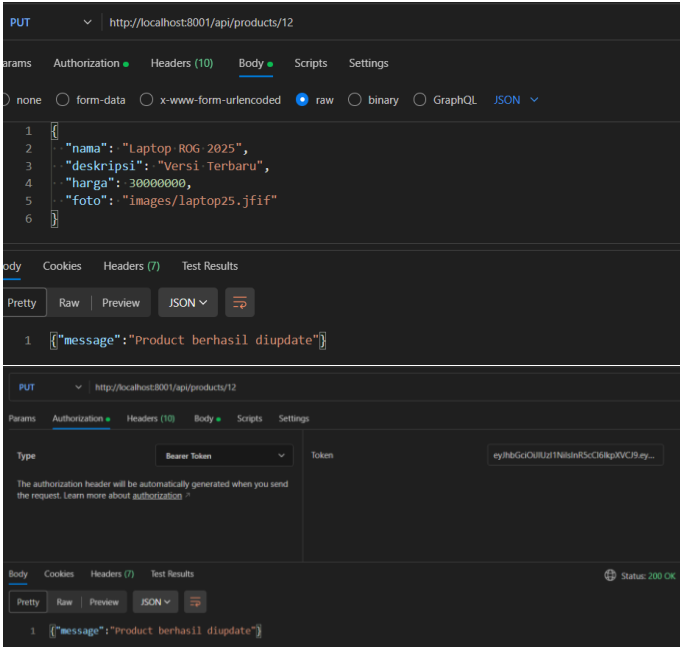
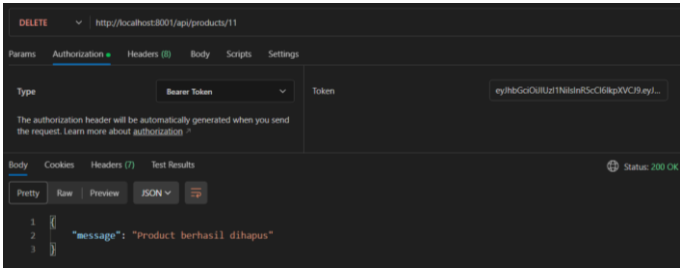
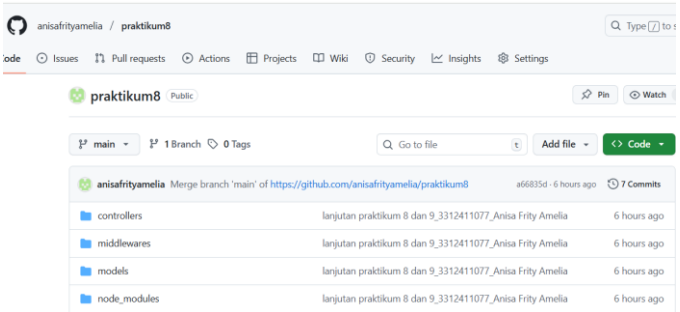
1. Buat proyek di Github dengan nama **Latihan9**
git init
git add .
git commit -m "first commit"
git branch -M main
git remote add origin <https://github.com/agunghakase/Latihan9.git>
git push -u origin main

Hasil Pengerjaan

No	Instruksi	Screenshot	Kendala/ Saran
A	Membuat JSON Web Token (Dynamic Bearer Token)		
1.	Install library JWT		
2.	Buat file .env di (root folder) ketikkan kode seperti pada gambar disamping		
3.	Tambahkan script require('dotenv').config(); di server.js lihat seperti pada gambar disamping		
4.	Jalankan perintah npm install dotenv		

5.	Menambahkan fungsi baru, findByEmail pada file user.model.js	<pre> ect8 > models > \$S user.model.js > @ User > @ findByEmail const db = require('../db.config'); const User = { getTall: callback => { db.query('SELECT * FROM users', callback); }, getById: (id, callback) => { db.query('SELECT * FROM users WHERE id = ?', [id], callback); }, create: (data, callback) => { db.query('INSERT INTO users (name, email) VALUES (?, ?)', [data.name, data.email], callback); }, update: (id, data, callback) => { db.query('UPDATE users SET name = ?, email = ? WHERE id = ?', [data.name, data.email, id], callback); }, delete: (id, callback) => { db.query('DELETE FROM users WHERE id = ?', [id], callback); }, findByEmail: (email, callback) => { db.query('SELECT * FROM users WHERE email = ?', [email], callback); } }; module.exports = User; </pre>	
6.	Buat file auth.controller.js kemudian tambahkan kode program seperti pada gambar disamping	<pre> ject8 > controllers > \$S auth.controller.js > @ login > @ login > @ User.findByEmail() callback const User = require('../models/user.model'); const bcrypt = require('bcryptjs'); const jwt = require('jsonwebtoken'); exports.login = (req, res) => { const { email, password } = req.body; User.findByEmail(email, (err, results) => { if (err) return res.status(500).json({ message: err.message }); if (results.length === 0) return res.status(404).json({ message: "User not found" }); const user = results[0]; /* const match = bcrypt.compareSync(password, user.password); */ const match = password === user.password; if (!match) return res.status(400).json({ message: "Wrong password" }); const token = jwt.sign({ id: user.id, email: user.email }, process.env.JWT_SECRET, { expiresIn: "7d" }); res.json({ message: "Login succes", token, user: { id: user.id, name: user.name, email: user.email } }); }); }; </pre>	
7.	Ubah isi kode program pada file auth.middleware.js yang sebelumnya menggunakan token biasa, menjadi json web token, lihat seperti gambar disamping	<pre> ject8 > middlewares > \$S auth.middleware.js > ... const jwt = require("jsonwebtoken"); const User = require("../models/user.model"); const authBearer = (req, res, next) => { const header = req.headers.authorization; if (!header !header.startsWith("Bearer ")) { return res.status(401).json({ message: "Unauthorized" }); } const token = header.split(" ")[1]; try { const decoded = jwt.verify(token, process.env.JWT_SECRET); User.getById(decoded.id, (err, results) => { if (err) return res.status(500).json({ message: err.message }); if (!results results.length === 0) { return res.status(401).json({ message: "Invalid token user" }); } req.user = results[0]; next(); }); } catch (err) { return res.status(401).json({ message: "Invalid token" }); } }; module.exports = { authBearer }; </pre>	

8.	Buat file auth.routes.js, kemudian ketikkan kode program seperti gambar disamping	<pre> ject8 > routes > JS auth.routes.js > ... const express = require("express"); const router = express.Router(); const authController = require("../controllers/auth.controller"); router.post("/", authController.login); module.exports = router; </pre>	
B	Gunakan POSTMAN dapatkan Token BEARER		
1.	Install postman di visual code		
2.	Dengan menggunakan postman, dapatkan bearer dengan memanggil endpoints /api/login (method: POST)		
3.	Catat bearer yang di dapatkan, lalu gunakan bearer tersebut untuk memanggil endpoints (POST, PUT, DELETE)	<p>POST</p>  <p>PUT</p> 	

		 <p>DELETE</p> 	
C	Github + Visual Code		
1.	Melakukan commit proyek ke dalam Github		
2.	Link Github	https://github.com/anisafrityamelia/praktikum8.git	