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 tunjukan 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

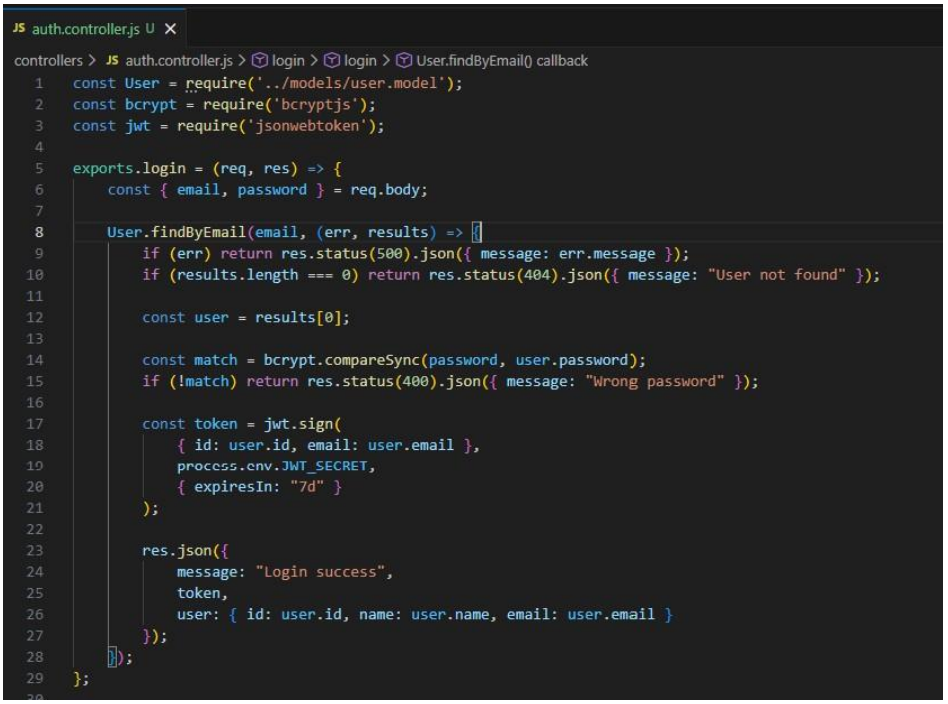
3. Tambahkan file auth.controller.js, auth.middleware.js, dan auth.routes.js
4. Buat file .env disamping server.js (root folder)
   Isi file .env dengan variable sebagai berikut:
   JWT_SECRET=”KUNCI-RAHASIA”
   JWT_EXPIRE=1d

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. Masukan script berikut pada auth.controller.js yang telah dibuat



```
JS auth.controller.js U ×
controllers > JS auth.controller.js > login > login > User.findByEmail() 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.findByEmail(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  };
```

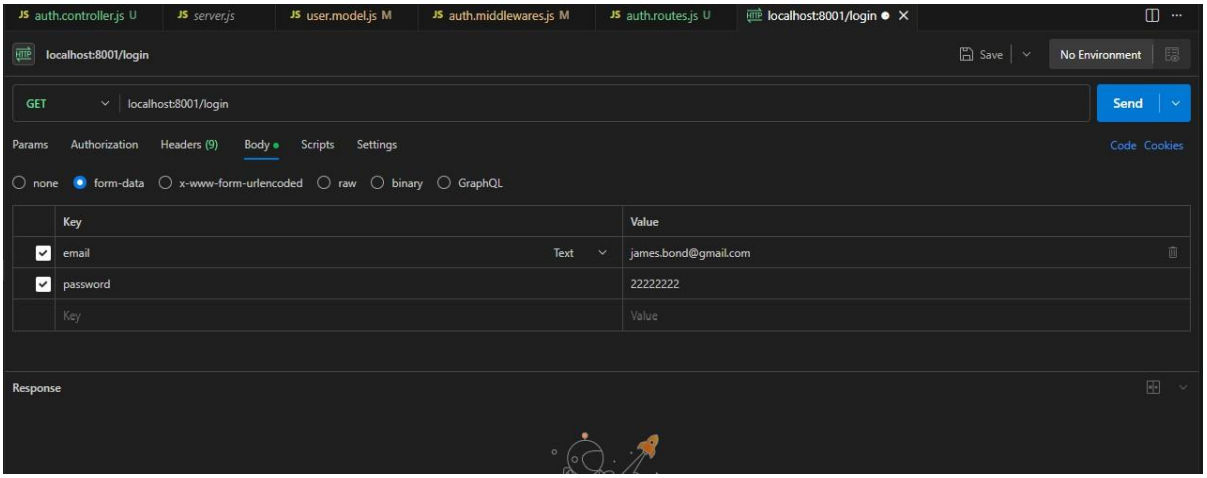
8. Ubah auth.middleware.js yang sebelumnya menggunakan token biasa, menjadi json web token seperti gambar dibawah ini

```
.controller.js U   JS user.model.js M   JS auth.middlewares.js M ×
ewares > 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

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



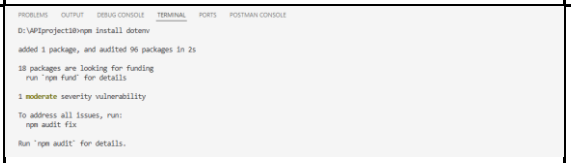
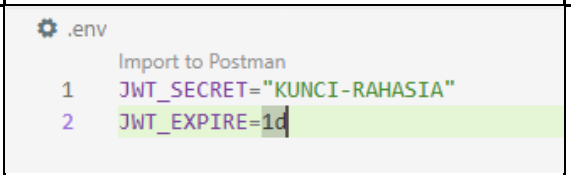
3. Catat bearer yang di dapatkan, lalu gunakan bearer tersebut untuk memanggil endpoints lainya yang pada praktikum 9 telah di proteksi.

F. 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.	Installasi dan Konfigurasi		
1.	1. Lanjutkan Project Praktikum 8-9, dengan menggunakan file yang sama (copy) 2. Install library JWT **npm install jsonwebtoken bcryptjs**		
2.	1. Install **npm install dotenv**		
B.	Github dan Viscode		
1.	1. Tambahkan file auth.controller.js, auth.middleware.js, dan auth.routes.js 2. Buat file .env disamping server.js (root folder) Isi file .env dengan variable sebagai berikut: JWT_SECRET=”KUNCI-RAHASIA” JWT_EXPIRE=1d 3. Tambahkan script berikut di server.js require('dotenv').config();		

4. Revisi model sebelumnya pada user.model.js dengan menambahkan fungsi baru seperti berikut, tambahkan findByEmail

5. Masukan script berikut pada auth.controller.js yang telah dibuat

6. Ubah auth.middleware.js yang sebelumnya menggunakan token biasa, menjadi json
web token seperti gambar dibawah ini

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

server.js

```
require('dotenv').config();
const express = require('express');
const app = express();
const PORT = 8801;

app.use(express.json());

// Default route
app.get('/', (req, res) => {
    res.send('API berjalan...');
});

// Routes user
const userRoutes = require('./routes/user.routes');
app.use('/api/users', userRoutes);

// Routes product
const productRoutes = require('./routes/product.routes');
app.use('/api/products', productRoutes);

// Routes auth (LOGIN)
const authRoutes = require('./routes/auth.routes');
app.use('/api/auth', authRoutes);

// Jalankan server
app.listen(PORT, () => {
    console.log(`Server berjalan di http://localhost:${PORT}`);
});
```

models > user.model.js > User

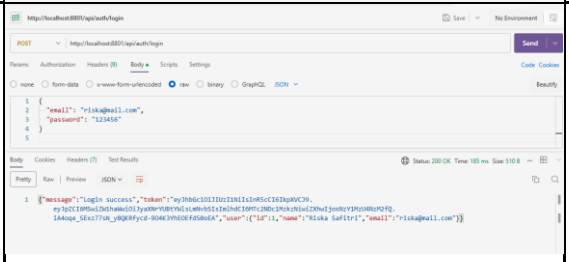
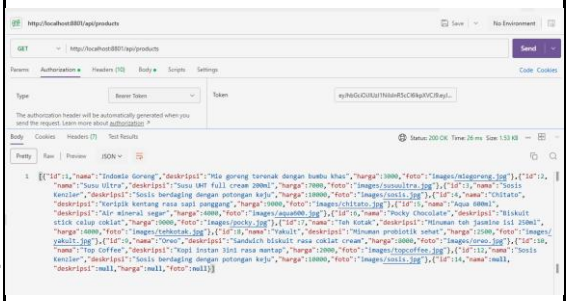
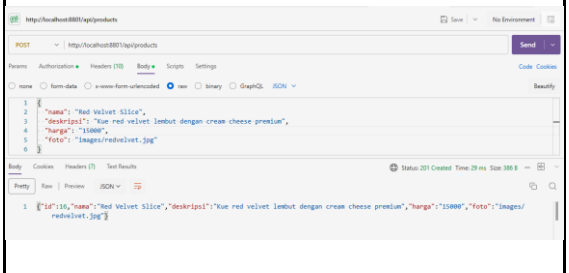
```
const db = require('./db.config');

// Model User (berisi query dasar)
const User = {
    getAll: 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);
    },

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

module.exports = User;
```

controllers > 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);
        //if (!match) return res.status(400).json({ message: "Wrong password" });

        if (password != user.password){
            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 success",
            token,
            user: { id: user.id, name: user.name, email: user.email }
        });
    });
};
```

middlewares > auth.middleware.js > authBearer

```
const jwt = require('jsonwebtoken');
const User = require("../models/user.model");

const authBearer = (req, res, next) => {
    const header = req.headers.authorization; // FIXED

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

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

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

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

            req.user = result[0]; // Simpan user ke request
            next();
        });

    } catch (err) {
        return res.status(401).json({ message: "Invalid or Expired Token" });
    }
};

module.exports = authBearer;
```

		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;	
2.	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 lainya yang pada praktikum 9 telah di proteksi.	   	
3.	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/**anastasyafdg/Latihan9.git** git push -u origin main