

## Praktikum 10 - 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. Masukkan 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

```

n.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

```
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;
```

10. Pada [server.js](#) tambahkan kode berikut untuk menambahkan routes

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

## B. SETUP DATABASE

1. Pada tabel user seharusnya password di enkripsi dengan bcryptjs. Oleh karena itu kita buat hasil enkripsinya dengan membuat tools node js seperti berikut
2. Buat file [createpwd.js](#) dan tulis kode seperti dibawah ini

```
const bcrypt = require('bcryptjs');
const hash = bcrypt.hashSync('passwordkamu', 10);
console.log(hash);
```

3. Ganti **passwordkamu** dengan password yang diinginkan, lalu ketik perintah node

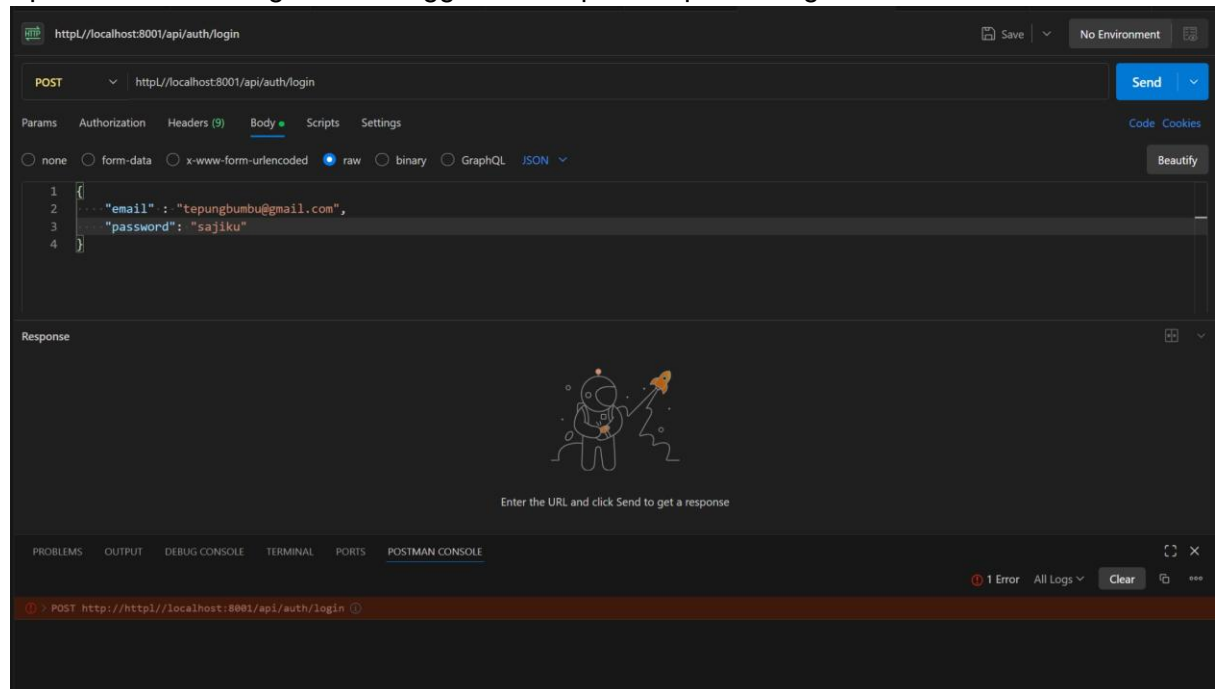
```
PS D:\Node\Praktikum8> node .\createpwd.js
$2b$10$UtTsJEFwY0JfqNOHTI859uf2QADWr9ry4g8w/t3/1fyZNZvwj0nC2
```

4. Ganti isi field password dengan hasil dari hash menggunakan bcryptjs.

Note: Ini adalah simulasi, kedepannya untuk create user dan password seharusnya menggunakan bcrpytsjs agar lebih aman.

## 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 api/auth/login



3. Catat bearer yang di dapatkan, lalu gunakan bearer tersebut untuk memanggil endpoints lainnya yang pada praktikum 9 telah di proteksi.
4. Token yang di dapat dari login, bisa digunakan untuk mengakses semua API yang diproteksi dalam sistem.

## F. Github + Visual Code

1. Buat proyek di Github dengan nama **Latihan10**

git init

git add

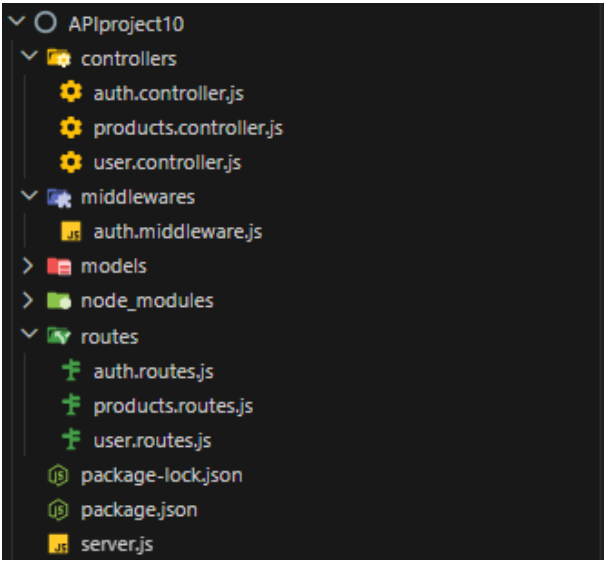
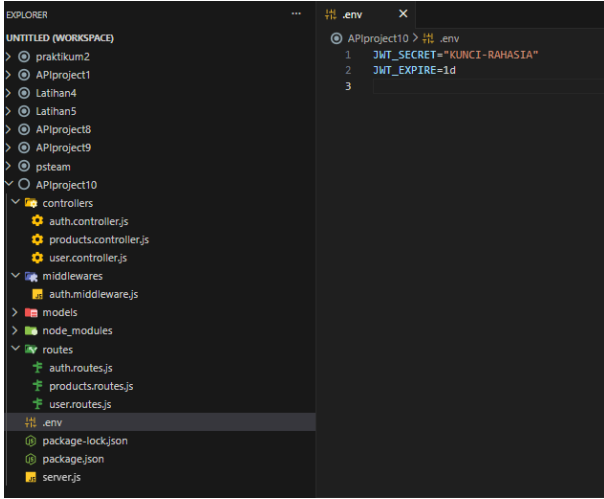
.

git commit -m "first commit" git branch -M main git remote add

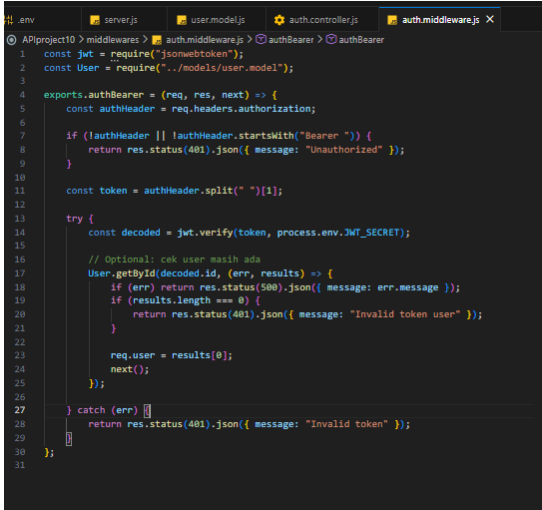
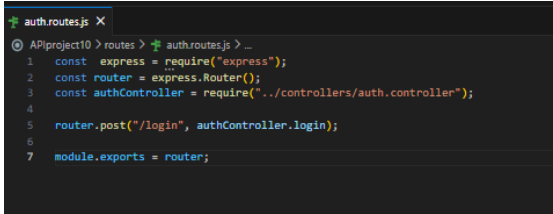
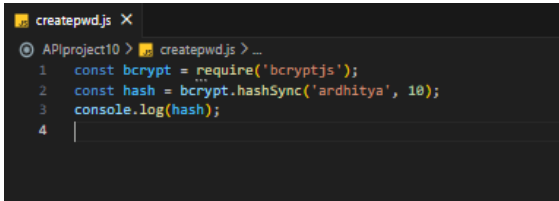
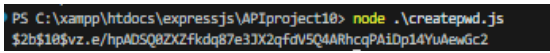
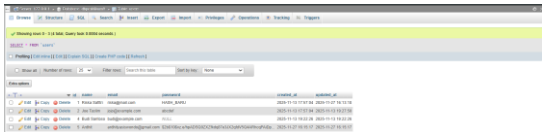
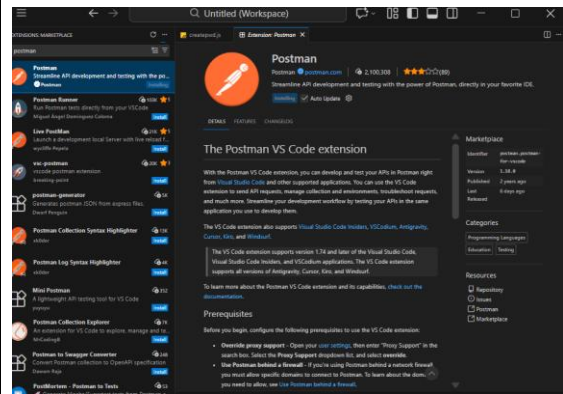
origin https://github.com/agunghakase/Latihan9.git git push -u

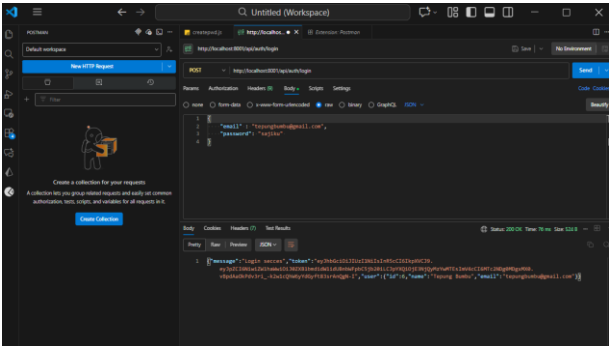
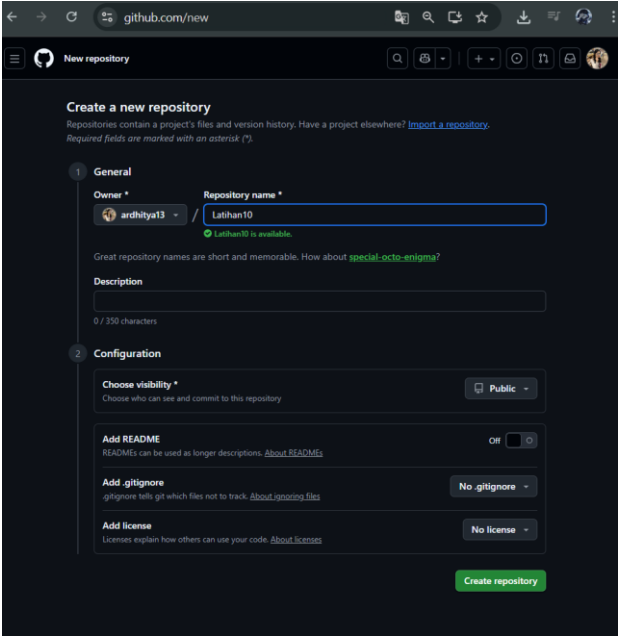
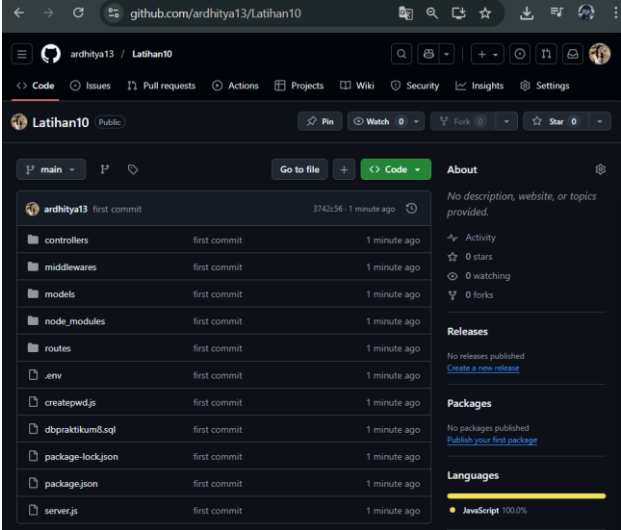
origin main

## Hasil Pengerjaan

No.	Instruksi	Screenshot	Kendala/Saran
A.	Installasi dan Konfigurasi		
1.	Copy project lanjutan		
2.	Install Library JWT		
3.	Tambahkan file <a href="#">auth.controller.js</a> , <a href="#">auth.middleware.js</a> , dan <a href="#">auth.routes.js</a>		
4.	Buat file .env		

B.	Github dan Viscode		
1.	Tambahkan script berikut di server.js <code>require('dotenv').config();</code>	<pre> server.js 1 require('dotenv').config(); 2 3 4 const express = require('express'); 5 const app = express(); 6 const PORT = 8081; 7 8 app.use(express.json()); 9 10 app.get('/', (req, res) =&gt; { 11   res.send('Hello, World!'); 12 }); 13 14 app.listen(PORT, () =&gt; { 15   console.log('Server berjalan di http://localhost:\${PORT}'); 16 }); 17 18 // Routes 19 const userRoutes = require('./routes/user.routes'); 20 app.use('/api/users', userRoutes); 21 22 // routes products 23 const productRoutes = require('./routes/products.routes'); 24 app.use('/api/products', productRoutes); </pre>	
2.	Revisi user.model.js	<pre> server.js user.model.js auth.controller.js 1 const db = require('./db.config'); 2 3 // Model user (berisi query dasar) 4 const User = { 5   getAll: callback =&gt; { 6     db.query('SELECT * FROM users', callback); 7   }, 8   getById: (id, callback) =&gt; { 9     db.query('SELECT * FROM users WHERE id = ?', [id], callback); 10  }, 11  create: (data, callback) =&gt; { 12    db.query('INSERT INTO users (name, email) VALUES (?, ?)', [data.name, data.email], callback); 13  }, 14  update: (id, data, callback) =&gt; { 15    db.query('UPDATE users SET name = ?, email = ? WHERE id = ?', [data.name, data.email], callback); 16  }, 17  delete: (id, callback) =&gt; { 18    db.query('DELETE FROM users WHERE id = ?', [id], callback); 19  }, 20  // Get user by email (untuk login) 21  findByEmail: (email, callback) =&gt; { 22    db.query('SELECT * FROM users WHERE email = ?', [email], callback); 23  }, 24 }; 25 module.exports = User; </pre>	
3.	Memasukan codingan ke auth.controller.js	<pre> server.js user.model.js auth.controller.js 1 const User = require('./models/user.model'); 2 const bcrypt = require('bcryptjs'); 3 const jwt = require('jsonwebtoken'); 4 5 exports.login = (req, res) =&gt; { 6   const { email, password } = req.body; 7 8   User.findByEmail(email, (err, results) =&gt; { 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 succes", 25      token, 26      user: { id: user.id, name: user.name, email: user.email } 27    }); 28  }); 29 }; </pre>	

4.	Ubah auth.middleware.js	 <pre> 1 const jwt = require("jsonwebtoken"); 2 const User = require("../models/user.model"); 3 4 exports.authBearer = (req, res, next) =&gt; { 5   const authHeader = req.headers.authorization; 6 7   if (!authHeader    !authHeader.startsWith("Bearer ")) { 8     return res.status(401).json({ message: "Unauthorized" }); 9   } 10 11   const token = authHeader.split(" ")[1]; 12 13   try { 14     const decoded = jwt.verify(token, process.env.JWT_SECRET); 15 16     // Optional: cek user masih ada 17     User.findById(decoded.id, (err, results) =&gt; { 18       if (err) return res.status(500).json({ message: err.message }); 19       if (results.length === 0) { 20         return res.status(401).json({ message: "Invalid token user" }); 21       } 22 23       req.user = results[0]; 24       next(); 25     }); 26   } catch (err) { 27     return res.status(401).json({ message: "Invalid token" }); 28   } 29 } 30 31 </pre>	
5.	Tambahkan login di auth.routes.js	 <pre> 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; </pre>	
6.	Membuat file createpwd.js dan memasukan codingan ini dan ganti password yang sesuai diinginkan	 <pre> 1 const bcrypt = require('bcryptjs'); 2 const hash = bcrypt.hashSync('ardhitya', 10); 3 console.log(hash); 4 </pre>	
7.	Jalankan perintah di terminal node .\createpwd.js	 <pre> PS C:\xampp\htdocs\expressjs\APIproject10&gt; node .\createpwd.js \$2b\$10\$vz.e/hpAD5Q8ZKZFKdg87e3JX2qFdv5Q4ARhcqPA1Op14YvAewGc2 </pre>	
8.	Ganti isi field password dengan hasil dari hash menggunakan bcryptjs		
9.	Install postman di vscode		

10.	Gunakan POSTMAN dapatkan Token BEARER		
11.	Membuat Repository Latihan10		
12.	Push Codingan ke github	<pre>PS C:\xampp\htdocs\expressjs\API\project10&gt; git remote add origin https://github.com/ardhiya13/Latihan10.git PS C:\xampp\htdocs\expressjs\API\project10&gt; git push -u origin main git: 'credential-manager-core' is not a git command. See 'git --help'. Enumerating objects: 1142, done. Counting objects: 100% (1142/1142), done. Delta compression using up to 12 threads Compressing objects: 100% (1066/1066), done. Writing objects: 100% (1142/1142), 1.20 MiB   679.00 KiB/s, done. Total 1142 (delta 176), reused 0 (delta 0), pack-reused 0 (from 0) remote: Resolving deltas: 100% (176/176), done. To https://github.com/ardhiya13/Latihan10.git  * [new branch] main -&gt; main branch 'main' set up to track 'origin/main'.</pre>	
13.	Berhasil memasukan ke github		



14.	Link github	<a href="https://github.com/ardhitya13/Latihan10">https://github.com/ardhitya13/Latihan10</a>	
-----	-------------	---	--