

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

```

js auth.controller.js U X
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

```

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

```
$ 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 dienkripsi 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

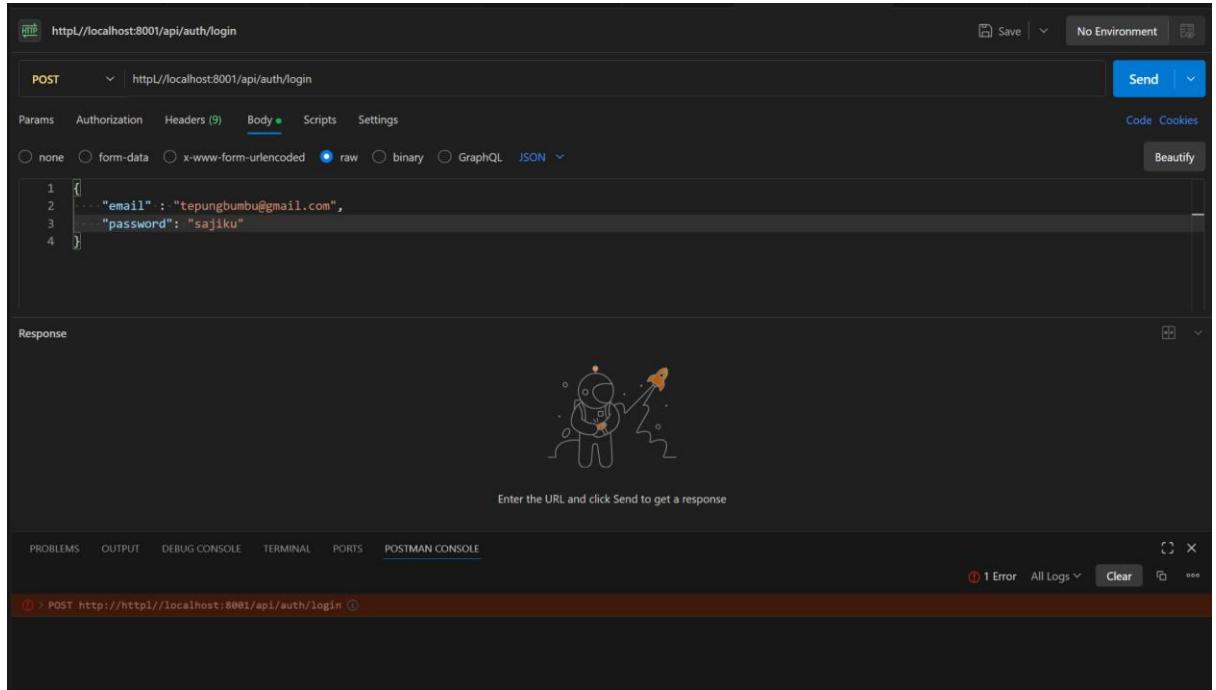
```
const bcrypt = require('bcryptjs');
const hash = bcrypt.hashSync('passwordkamu', 10);
console.log(hash);
```

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 bcryptjs 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
```

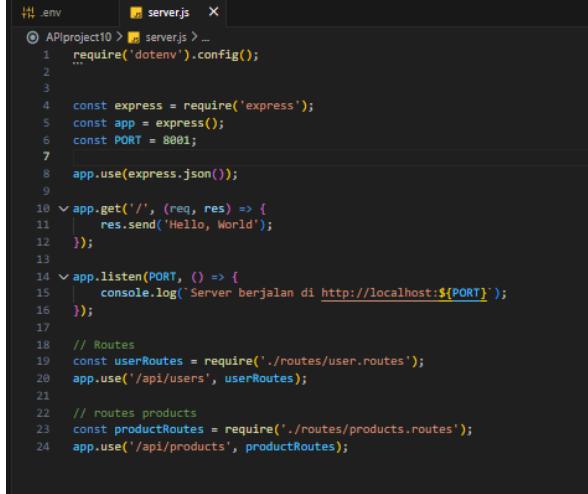
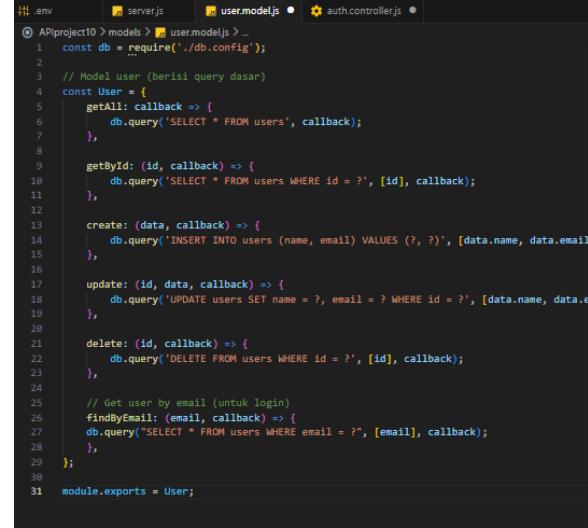
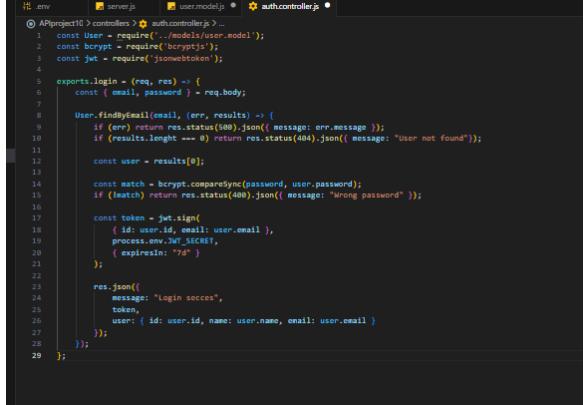
```
git add
```

```
.
```

```
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 Pengeroaan

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

B.	Github and Viscode	
1.	Tambahkan script berikut di server.js <code>require('dotenv').config();</code>	
2.	Revisi user.model.js	
3.	Memasukan codingan ke auth.controller.js	

4.	Ubah auth.middleware.js	<pre> // .env // server.js // user.model.js // auth.controller.js // auth.middleware.js const jwt = require("jsonwebtoken"); const User = require("../models/user.model"); exports.authBearer = (req, res, next) => { const authHeader = req.headers.authorization; if (!authHeader !authHeader.startsWith("Bearer ")) { return res.status(401).json({ message: "Unauthorized" }); } const token = authHeader.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" }); } }; </pre>																
5.	Tambahkan login di auth.routes.js	<pre> // auth.routes.js // API\project10> routes > auth.routes.js ... const express = require("express"); const router = express.Router(); const authController = require("../controllers/auth.controller"); router.post("/login", authController.login); module.exports = router; </pre>																
6.	Membuat file createpwd.js dan memasukan codingan ini dan ganti password yang sesuai diinginkan	<pre> // createpwd.js // API\project10> createpwd.js ... const bcrypt = require('bcryptjs'); const hash = bcrypt.hashSync('ardhiya', 10); console.log(hash); </pre>																
7.	Jalankan perintah di terminal node .\createpwd.js	<pre> PS C:\xampp\htdocs\expressjs\API\project10> node .\createpwd.js \$2b\$10\$Vz.e/hpADSQ0ZXZfkqdq87e3JX2qFdV5Q4ARhcgPAiDp14YuAewGc2 </pre>																
8.	Ganti isi field password dengan hasil dari hash menggunakan bcryptjs	<table border="1"> <thead> <tr> <th>ID</th> <th>Name</th> <th>password</th> <th>created_at</th> <th>updated_at</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Rahmat</td> <td>\$2b\$10\$Vz.e/hpADSQ0ZXZfkqdq87e3JX2qFdV5Q4ARhcgPAiDp14YuAewGc2</td> <td>2023-11-11 10:12:24</td> <td>2023-11-11 10:12:24</td> </tr> <tr> <td>2</td> <td>Arif</td> <td>\$2b\$10\$Vz.e/hpADSQ0ZXZfkqdq87e3JX2qFdV5Q4ARhcgPAiDp14YuAewGc2</td> <td>2023-11-11 10:12:24</td> <td>2023-11-11 10:12:24</td> </tr> </tbody> </table>	ID	Name	password	created_at	updated_at	1	Rahmat	\$2b\$10\$Vz.e/hpADSQ0ZXZfkqdq87e3JX2qFdV5Q4ARhcgPAiDp14YuAewGc2	2023-11-11 10:12:24	2023-11-11 10:12:24	2	Arif	\$2b\$10\$Vz.e/hpADSQ0ZXZfkqdq87e3JX2qFdV5Q4ARhcgPAiDp14YuAewGc2	2023-11-11 10:12:24	2023-11-11 10:12:24	
ID	Name	password	created_at	updated_at														
1	Rahmat	\$2b\$10\$Vz.e/hpADSQ0ZXZfkqdq87e3JX2qFdV5Q4ARhcgPAiDp14YuAewGc2	2023-11-11 10:12:24	2023-11-11 10:12:24														
2	Arif	\$2b\$10\$Vz.e/hpADSQ0ZXZfkqdq87e3JX2qFdV5Q4ARhcgPAiDp14YuAewGc2	2023-11-11 10:12:24	2023-11-11 10:12:24														
9.	Install postman di vscode																	

10.

Gunakan POSTMAN
dapatkan Token BEARER

```

POST /api/login
{
  "email": "reygiant@gmail.com",
  "password": "123456"
}
{
  "message": "Success login", "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJsb2dpbi5nb29nbGUuY29tIiwidWFpbCI6ImFkbWluQGdtYWlsLmNvbSIsImlhdCI6MTc2NTU0OTQzOH0.g5tGM4gc-w4kcgk1m6rstfpvcA5RU45idrnjzu_ks"
}
  
```

11

Catat bearer yang di
dapatkan, lalu gunakan
bearer tersebut untuk
memanggil endpoints
(POST, PUT, DELETE)

POST

```

POST /api/products
{
  "nama": "Sari Roti",
  "deskripsi": "Sari Roti 100g",
  "harga": 12000,
  "foto": "images/sari_roti.jpg"
}
{
  "id": 15,
  "nama": "Sari Roti",
  "deskripsi": "Sari Roti 100g",
  "harga": 12000,
  "foto": "images/sari_roti.jpg"
}
  
```

Wajib Masukan Bearer Token dulu baru bisa send

Type: Bearer To...
The authorization header will be automatically generated when you send the request. Learn more about [authorization](#).

Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJsb2dpbi5nb29nbGUuY29tIiwidWFpbCI6ImFkbWluQGdtYWlsLmNvbSIsImlhdCI6MTc2NTU0OTQzOH0.g5tGM4gc-w4kcgk1m6rstfpvcA5RU45idrnjzu_ks

PUT

The screenshot shows a POST request to `http://localhost:8001/api/products/15`. The request body contains the following JSON:

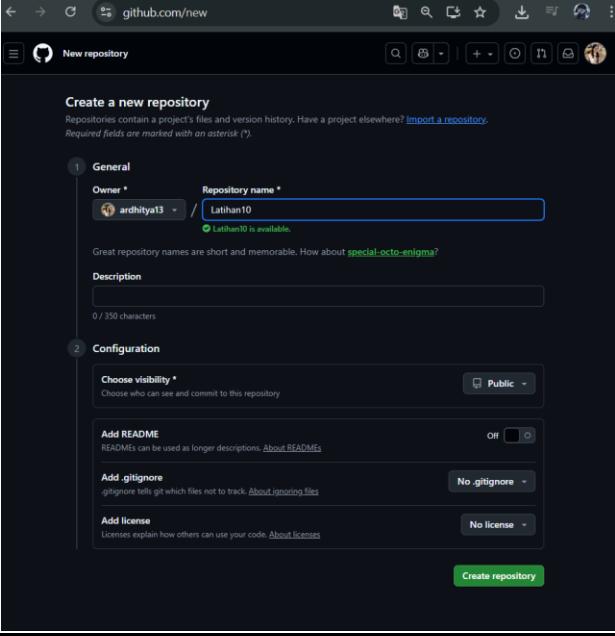
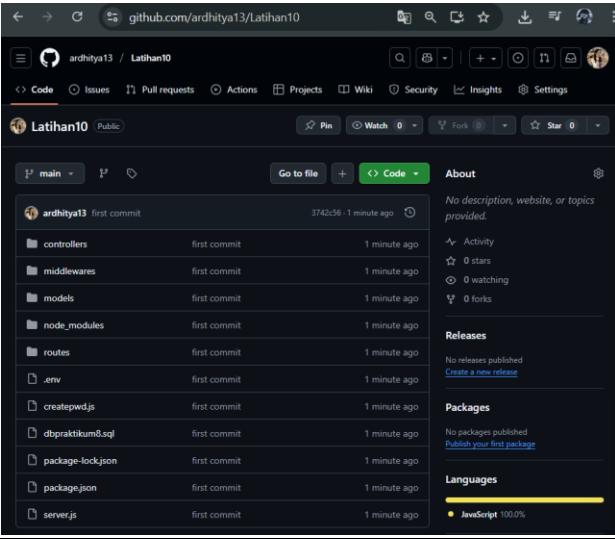
```
1 [ { 2 "nama" : "Sari Roti", 3 "deskripsi" : "Sari Roti 100g", 4 "harga" : 12000, 5 "foto" : "images/sari_roti.jpg" 6 } ]
```

The response status is 200 OK, with a message: "Produk diupdate".

DELETE

The screenshot shows a DELETE request to `http://localhost:8001/api/products/15`. The request body contains the same JSON as the previous update request.

The response status is 200 OK, with a message: "Produk dihapus".

12.	Membuat Repository Latihan10		
13.	Push Codingan ke github	<pre>PS C:\xampp\htdocs\expressjs\API\project10> git remote add origin https://github.com/ardhitya13/Latihan10.git PS C:\xampp\htdocs\expressjs\API\project10> 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), 1.28 MB 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/ardhitya13/Latihan10.git * [new branch] main -> main branch 'main' set up to track 'origin/main'.</pre>	
14.	Berhasil memasukan ke github		
15.	Link github	https://github.com/ardhitya13/Latihan10	