

Laporan Praktikum
Mata Kuliah Pemrograman Web (PEMWEB)



Pertemuan 6. Praktikum 6
“Session”

Dosen Pengampu:
Willdan Aprizal Aripin, S.Pd., M.Kom.

Disusun Oleh:
Putri Wahyuni
2310271

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

I. PENDAHULUAN

Session merupakan sebuah mekanisme dalam aplikasi web yang berfungsi untuk menyimpan data pengguna secara sementara selama mereka berinteraksi dengan aplikasi. Di Node.js, session digunakan untuk melacak dan menyimpan status pengguna antara berbagai permintaan HTTP (request) yang terpisah.

Poin penting terkait session:

1. **Penyimpanan Data Sementara:** Saat pengguna login, informasi seperti ID pengguna, username, atau token akan disimpan dalam session sehingga server dapat mengenali pengguna selama sesi berlangsung.
2. **Unik untuk Setiap Pengguna:** Setiap pengguna yang terhubung ke aplikasi memiliki session yang unik. Ketika pengguna membuat request baru (seperti berpindah halaman), session terkait akan diidentifikasi dan diakses.
3. **Menggunakan Cookie:** Session biasanya disimpan di server, dan browser menggunakan cookie untuk melacak session tersebut.

II. ALAT DAN BAHAN

- Laptop
- Aplikasi Visual Studio Code
- Google Chrome
- Xampp

III. PENJELASAN

1. Struktur folder pada praktikum ke-6 ini mengikuti susunan seperti pada gambar di bawah, di mana kita menyesuaikan dengan struktur yang ditunjukkan pada gambar.

```
user-management/  
├── views/  
│   ├── login.ejs  
│   ├── register.ejs  
│   └── profile.ejs  
├── public/  
│   └── styles.css  
├── node_modules/  
├── app.js  
├── config/  
│   └── db.js  
├── routes/  
│   └── auth.js  
└── package.json
```

Dalam praktikum kali ini juga kita sebelumnya harus menginstal yang akan kita perlukan:

2. Inisialisasi Node.js Project

1. Inisialisasi project Node.js:

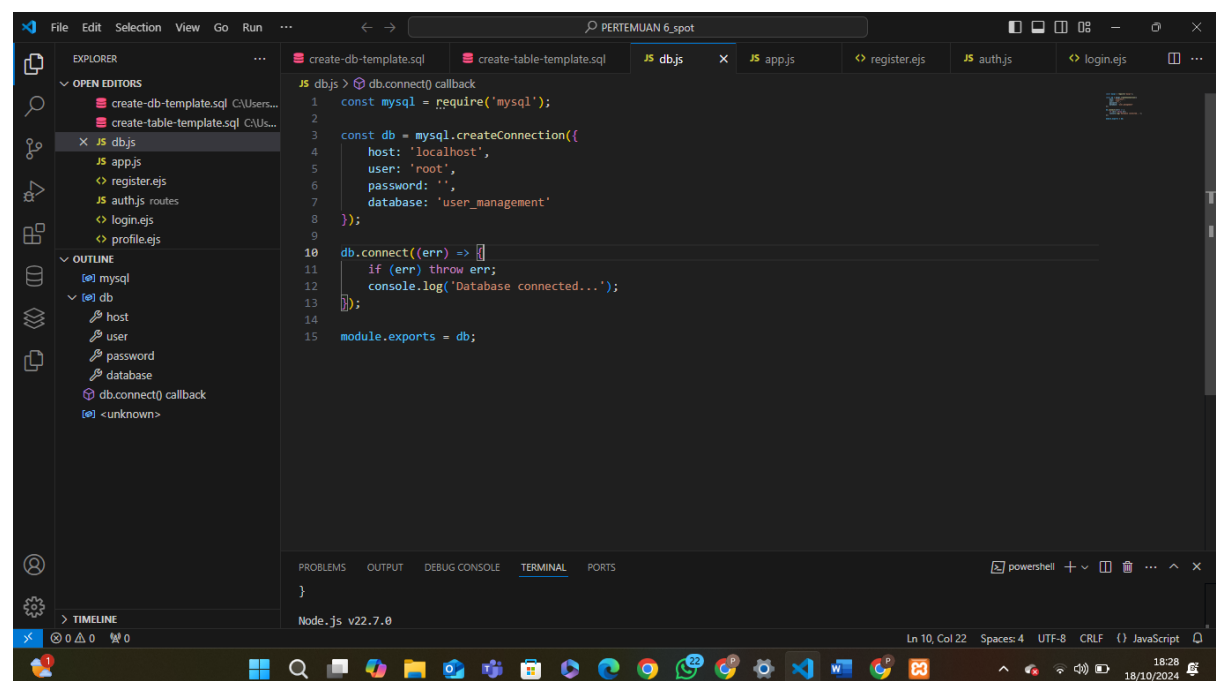
```
npm init -y
```

2. Install dependencies yang diperlukan:

```
npm install express mysql bcryptjs body-parser express-session ejs
```

- express: framework web untuk Node.js.
- mysql: driver untuk koneksi ke MySQL.
- bcryptjs: untuk hashing password.
- body-parser: untuk parsing request body.
- express-session: untuk manajemen sesi.
- ejs : untuk template engine.

➤ Kodingan db.js

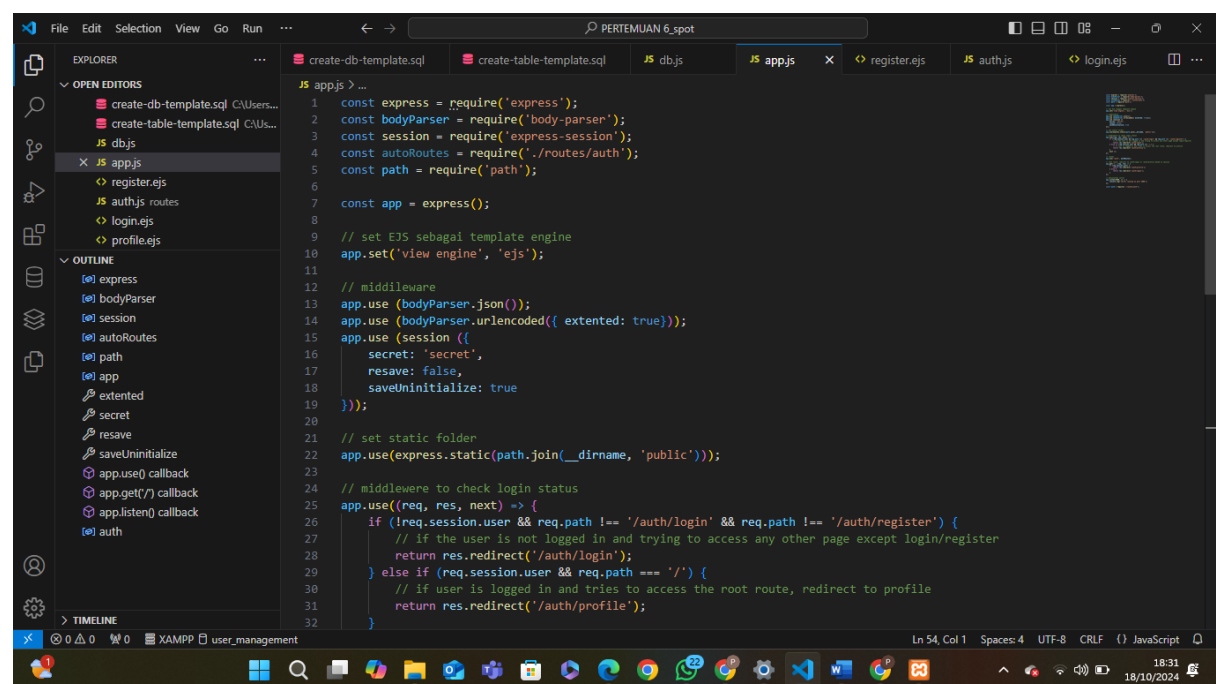


Penjelasan kodingan tersebut:

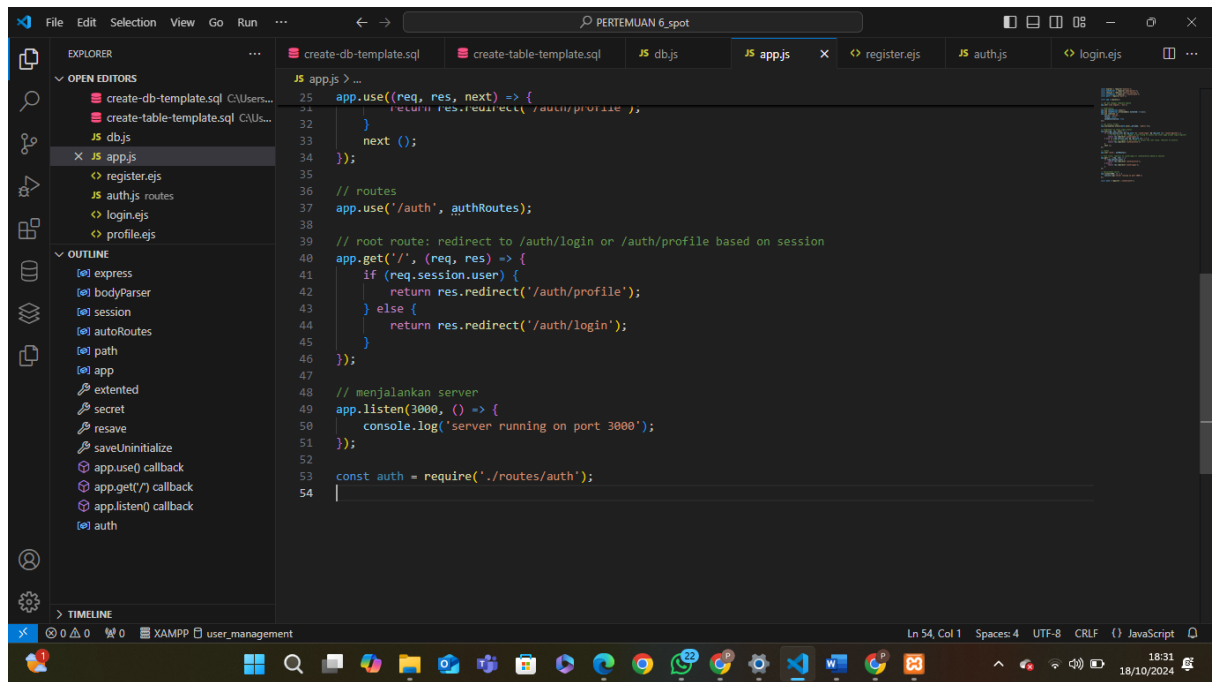
Kode ini berfungsi untuk mengonfigurasi koneksi ke database MySQL menggunakan Node.js. Dimulai dengan mengimpor modul `mysql` yang dibutuhkan untuk menghubungkan aplikasi dengan database MySQL. Koneksi ke database kemudian dibuat dengan memanfaatkan fungsi `mysql.createConnection()`, yang menerima beberapa parameter seperti `host` (diatur ke `localhost` untuk server lokal), `user` (`root` sebagai nama pengguna), dan `password` (dibiarkan kosong karena biasanya tidak dibutuhkan dalam pengaturan pengembangan lokal). Nama database yang akan digunakan adalah `user_management`.

Setelah koneksi dikonfigurasi, fungsi `db.connect()` dipanggil untuk mencoba menyambungkan ke database. Jika terjadi kesalahan, misalnya koneksi gagal atau parameter tidak tepat, kesalahan tersebut akan dilemparkan dengan `throw err`. Jika koneksi berhasil, pesan "Database connected..." akan dicetak di konsol, menandakan bahwa koneksi telah berhasil dibuat. Akhirnya, objek `db` yang berisi koneksi ini diekspor melalui `module.exports`, memungkinkan koneksi ini untuk digunakan di file lain dalam aplikasi guna menjalankan query ke database.

➤ Kodingan app.js



```
1  const express = require('express');
2  const bodyParser = require('body-parser');
3  const session = require('express-session');
4  const autoRoutes = require('./routes/auth');
5  const path = require('path');
6
7  const app = express();
8
9  // set EJS sebagai template engine
10 app.set('view engine', 'ejs');
11
12 // middleware
13 app.use(bodyParser.json());
14 app.use(bodyParser.urlencoded({ extended: true }));
15 app.use(session({
16   secret: 'secret',
17   resave: false,
18   saveUninitialized: true
19 }));
20
21 // set static folder
22 app.use(express.static(path.join(__dirname, 'public')));
23
24 // middleware to check login status
25 app.use((req, res, next) => {
26   if (!req.session.user && req.path !== '/auth/login' && req.path !== '/auth/register') {
27     // if the user is not logged in and trying to access any other page except login/register
28     return res.redirect('/auth/login');
29   } else if (req.session.user && req.path === '/') {
30     // if user is logged in and tries to access the root route, redirect to profile
31     return res.redirect('/auth/profile');
32   }
33   next();
34 });
```



Penjelasan kodingannya:

Dalam kodingan mengatur aplikasi web menggunakan Express, ini sebuah framework untuk Node.js, yang digunakan untuk menangani routing, sesi pengguna, dan rendering template EJS.

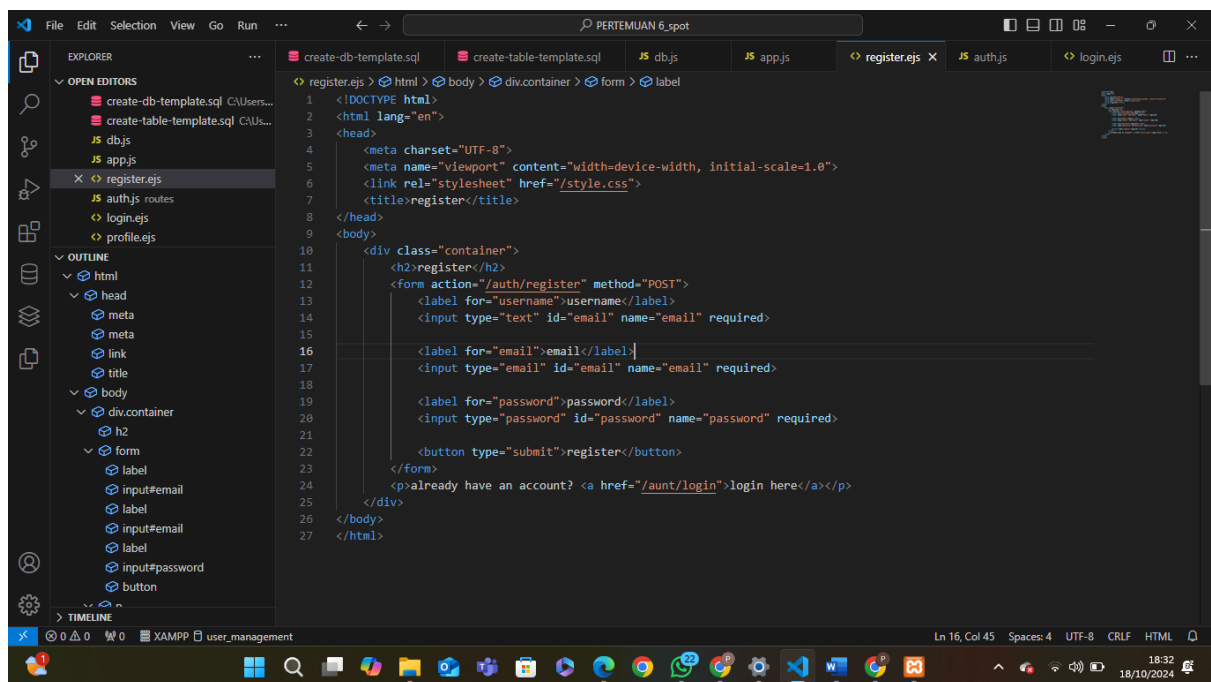
- Impor modul
 - express: Modul ini digunakan untuk membuat aplikasi web.
 - body-parser: Digunakan untuk menangani parsing data dari form (JSON dan URL-encoded).
 - express-session: Mengelola sesi pengguna, memungkinkan aplikasi untuk melacak status pengguna (misalnya login).
 - path: Modul bawaan Node.js untuk menangani dan mengelola jalur file.
 - authRoutes: Mengimpor file auth.js dari folder routes, yang berisi logika untuk autentikasi pengguna.
- Membuat Aplikasi Express: `const app = express();` membuat instance dari aplikasi Express untuk digunakan dalam pengaturan server dan routing.
- Mengatur Template Engine: `app.set('view engine', 'ejs');` mengonfigurasi EJS sebagai template engine. EJS memungkinkan kita untuk menyisipkan data dinamis ke dalam halaman HTML.
- Middleware:
 - `bodyParser.json()` dan `bodyParser.urlencoded({ extended: true })`: Mengaktifkan parsing data dari permintaan HTTP (misalnya form) menjadi format JSON dan URL-encoded.
 - `express-session`: Mengatur konfigurasi sesi. Opsi `secret` digunakan untuk menandatangani sesi, `resave: false` mencegah penyimpanan sesi jika tidak ada perubahan, dan `saveUninitialized: true` menyimpan sesi yang belum diinisialisasi.
 - `express.static`: Mengatur folder public sebagai folder tempat file statis (seperti CSS,

gambar, atau JavaScript) berada. Semua file di dalam folder ini bisa diakses secara publik.

- Middleware Cek Status Login: Middleware ini berfungsi untuk memeriksa apakah pengguna sudah login sebelum mengakses halaman selain /auth/login atau /auth/register. Jika belum login dan mencoba mengakses halaman lain, pengguna akan diarahkan ke halaman login. Jika sudah login dan mencoba mengakses root route ('/'), maka akan diarahkan ke profil pengguna.
- Routing:
 - app.use('/auth', authRoutes): Semua rute yang diawali dengan /auth akan diarahkan ke authRoutes, yang menangani autentikasi (seperti login dan register).
 - app.get('/'): Mengarahkan pengguna ke halaman login atau profil berdasarkan status sesi mereka (login atau belum).
- Menjalankan Server: app.listen(3000, () => { console.log('server running on port 3000'); }); Mengaktifkan server pada port 3000 dan menampilkan pesan "server running on port 3000" di konsol saat server aktif.

➤ Untuk tampilan view nya dimana terdapat: register.ejs, login.ejs, dan profile.ejs

- Kodingan register.ejs



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="/style.css">
7   <title>register</title>
8 </head>
9 <body>
10   <div class="container">
11     <h2>register</h2>
12     <form action="/auth/register" method="POST">
13       <label for="username">username</label>
14       <input type="text" id="email" name="email" required>
15
16       <label for="email">email</label>
17       <input type="email" id="email" name="email" required>
18
19       <label for="password">password</label>
20       <input type="password" id="password" name="password" required>
21
22       <button type="submit">register</button>
23     </form>
24     <p>already have an account? <a href="/auth/login">login here</a></p>
25   </div>
26 </body>
27 </html>
```

Penjelasan kodingan:

- DOCTYPE dan Tag HTML:
 - <!DOCTYPE html>: Menentukan bahwa dokumen ini adalah dokumen HTML5.
 - <html lang="en">: Memulai elemen root HTML, dengan atribut lang="en" yang menyatakan bahwa bahasa yang digunakan adalah bahasa Inggris.
- Elemen Head:

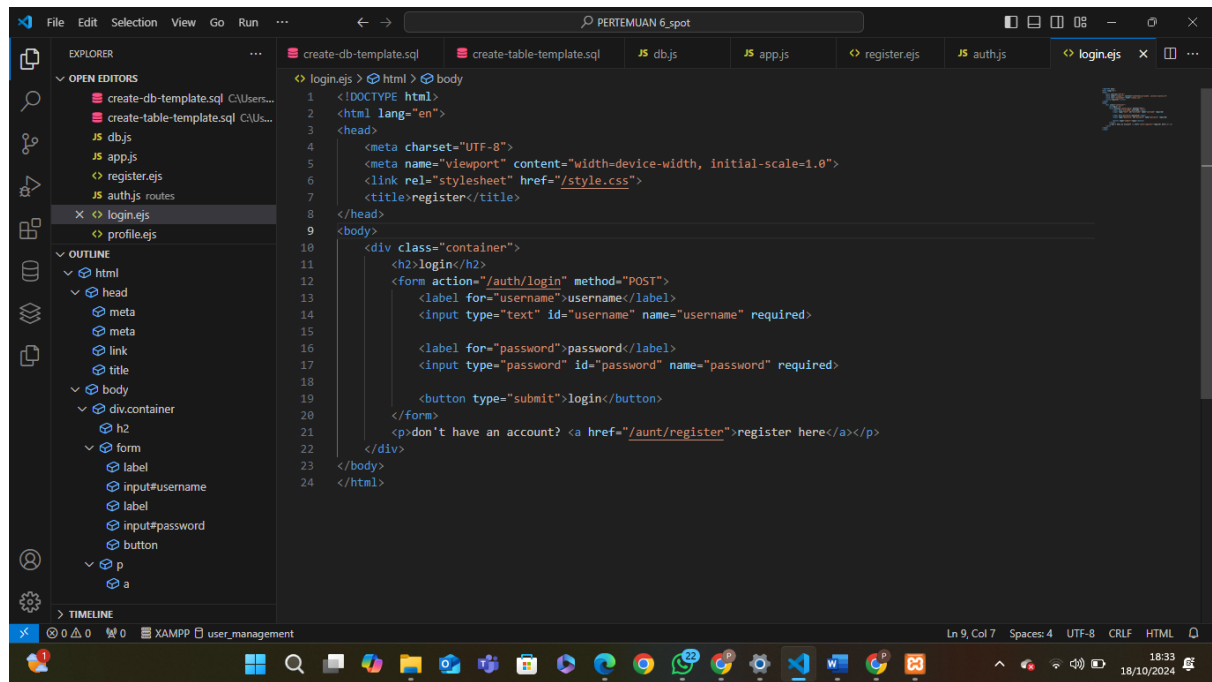
- `<meta charset="UTF-8">`: Mengatur karakter encoding untuk halaman web ini sebagai UTF-8, yang mendukung berbagai karakter internasional.
- `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Memastikan bahwa halaman web diatur agar responsif dan tampil dengan baik di perangkat seluler. `width=device-width` mengatur lebar halaman sesuai dengan lebar layar perangkat, dan `initial-scale=1.0` menetapkan zoom awal pada 100%.
- `<link rel="stylesheet" href="/style.css">`: Menghubungkan halaman dengan file CSS eksternal yang terletak di `/style.css`. File ini akan mengatur tampilan atau styling dari halaman web.
- `<title>register</title>`: Menentukan judul halaman yang akan ditampilkan pada tab browser.
- Elemen Body:
 - `<body>`: Elemen yang memuat semua konten yang akan dilihat oleh pengguna.
 - `<div class="container">`: Sebuah elemen `div` yang mengelompokkan konten, dan diberikan kelas `"container"` untuk keperluan styling. Biasanya, ini akan digunakan untuk menata layout halaman menggunakan CSS.
- Judul dan Formulir Registrasi:
 - `<h2>register</h2>`: Judul halaman yang menunjukkan bahwa ini adalah halaman registrasi.
 - `<form action="/auth/register" method="POST">`: Sebuah elemen form yang akan mengirimkan data ke server saat pengguna mendaftar. Atribut `action="/auth/register"` menunjukkan bahwa form ini akan mengirimkan data ke URL `/auth/register`, dan `method="POST"` berarti data akan dikirimkan menggunakan metode HTTP POST, yang lebih aman untuk mengirimkan data sensitif seperti password.
- Label dan Input Field:
 - `<label for="username">username</label>` dan `<input type="text" id="username" name="username" required>`: Label ini terhubung dengan field input yang memungkinkan pengguna memasukkan username mereka. Atribut `required` menunjukkan bahwa field ini wajib diisi sebelum form dapat dikirimkan.
 - `<label for="email">email</label>` dan `<input type="email" id="email" name="email" required>`: Elemen ini memungkinkan pengguna memasukkan alamat email, dan tipe input email memastikan bahwa yang dimasukkan harus berupa format email yang valid.
 - `<label for="password">password</label>` dan `<input type="password" id="password" name="password" required>`: Elemen ini menyediakan tempat untuk memasukkan kata sandi. Tipe input password akan menyembunyikan karakter yang diketikkan.
- Tombol Submit:
 - `<button type="submit">register</button>`: Sebuah tombol yang mengirimkan form ketika diklik.
- Link ke Halaman Login:
 - `<p>already have an account? login here</p>`:

Menyediakan tautan untuk pengguna yang sudah memiliki akun untuk menuju halaman login.

- Penutup Tag:

- Penutup untuk elemen div, body, dan html, yang menandakan akhir dari struktur halaman. Secara keseluruhan, kode ini membangun halaman registrasi yang memungkinkan pengguna untuk memasukkan informasi dasar seperti username, email, dan password, dan kemudian mengirimkannya ke server untuk diproses.

➤ Kodingan login.ejs



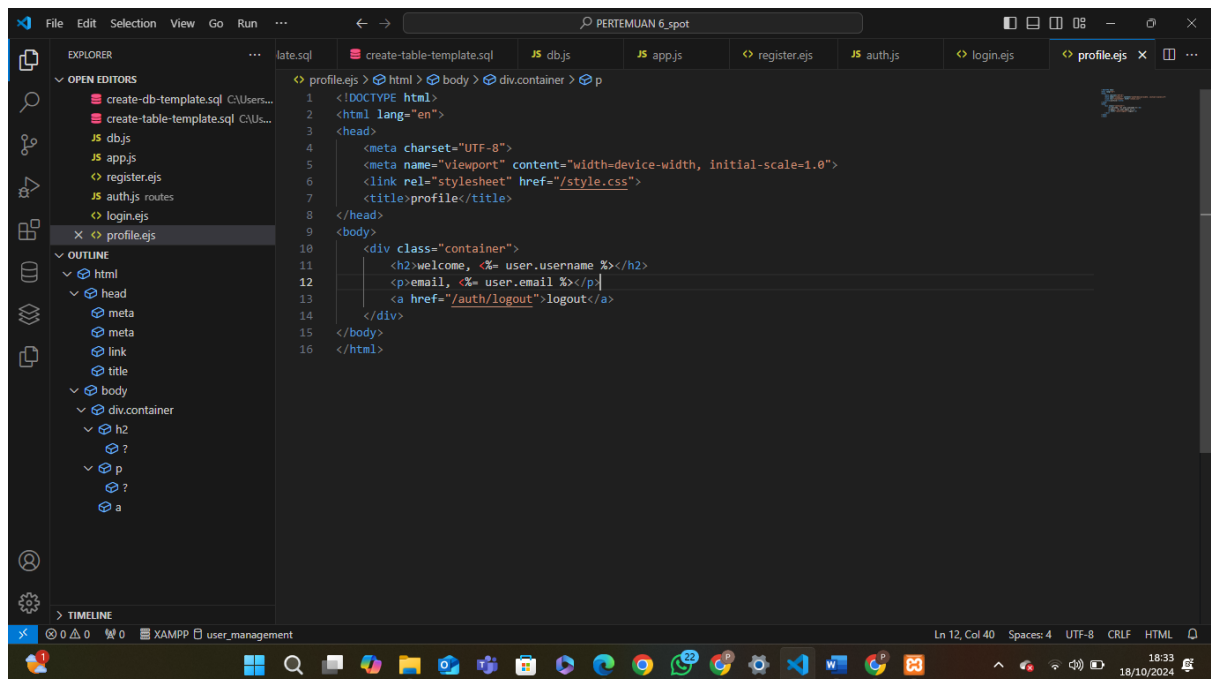
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="/style.css">
7   <title>register</title>
8 </head>
9 <body>
10   <div class="container">
11     <h2>login</h2>
12     <form action="/auth/login" method="POST">
13       <label for="username">username</label>
14       <input type="text" id="username" name="username" required>
15
16       <label for="password">password</label>
17       <input type="password" id="password" name="password" required>
18
19       <button type="submit">login</button>
20     </form>
21     <p>don't have an account? <a href="/auth/register">register here</a></p>
22   </div>
23 </body>
24 </html>
```

Penjelasan kodingan:

Dokumen dimulai dengan deklarasi `<!DOCTYPE html>` dan elemen `<html lang="en">` untuk menetapkan bahasa Inggris. Di dalam `<head>`, terdapat tag `<meta>` untuk dukungan karakter dan responsivitas, serta link ke file CSS eksternal. Dalam `<body>`, konten utama berada di dalam `<div class="container">`, dengan judul `<h2>login</h2>` dan formulir login menggunakan `<form action="/auth/login" method="POST">`. Pengguna harus mengisi username dan password yang bersifat wajib sebelum menekan tombol `<button type="submit">login</button>` untuk mengirim data. Terdapat juga tautan bagi pengguna yang belum memiliki akun untuk mendaftar. Struktur halaman ditutup dengan elemen `</body>` dan `</html>`, memungkinkan pengguna untuk masuk atau mendaftar.

Secara keseluruhan, kode ini menghasilkan halaman login yang memungkinkan pengguna untuk memasukkan username dan password, serta menyediakan tautan untuk pendaftaran jika pengguna belum memiliki akun.

➤ Kodingan profile.ejs



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="/style.css">
7   <title>profile</title>
8 </head>
9 <body>
10   <div class="container">
11     <h2>welcome, <%= user.username %></h2>
12     <p>email, <%= user.email %></p>
13     <a href="/auth/logout">logout</a>
14   </div>
15 </body>
16 </html>
```

Penjelasan kodingan:

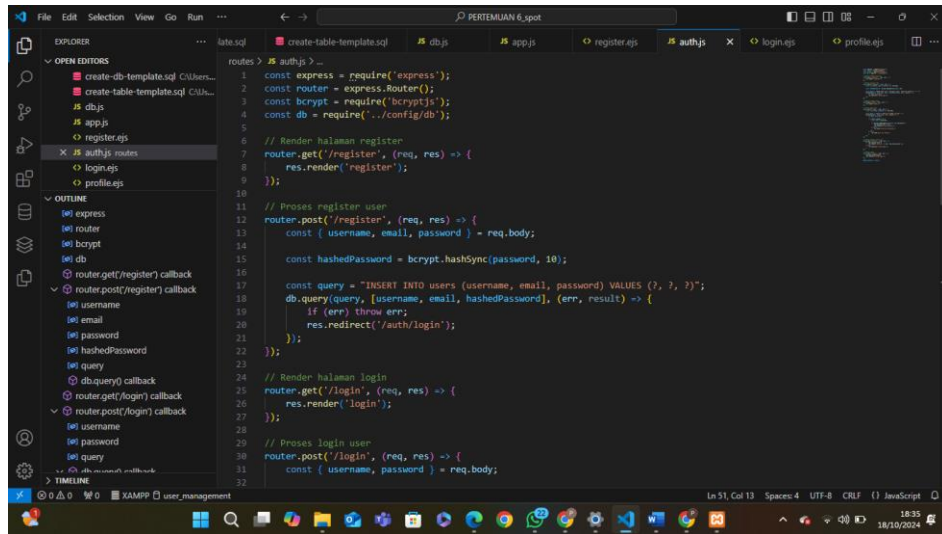
Diawali dengan deklarasi `<!DOCTYPE html>` yang menandakan dokumen ini adalah HTML5, halaman ini kemudian membuka elemen HTML dengan `<html lang="en">`, menetapkan bahasa sebagai Inggris. Dalam bagian kepala (`<head>`), terdapat pengaturan karakter encoding menggunakan `<meta charset="UTF-8">` dan pengaturan responsivitas viewport dengan `<meta name="viewport" content="width=device-width, initial-scale=1.0">`. Halaman ini menghubungkan file CSS eksternal melalui `<link rel="stylesheet" href="/style.css">`, dan menetapkan judul halaman menjadi "profile" dengan `<title>profile</title>`.

Di bagian tubuh (`<body>`), konten utama dibungkus dalam `<div class="container">`, di mana terdapat pesan sambutan yang menampilkan username pengguna dengan sintaks templating `<%= user.username %>`. Selanjutnya, email pengguna juga ditampilkan menggunakan sintaks templating yang sama. Terakhir, terdapat tautan untuk logout yang mengarah ke endpoint `/auth/logout`. Penutupan halaman dilakukan dengan elemen `</body>` dan `</html>`.

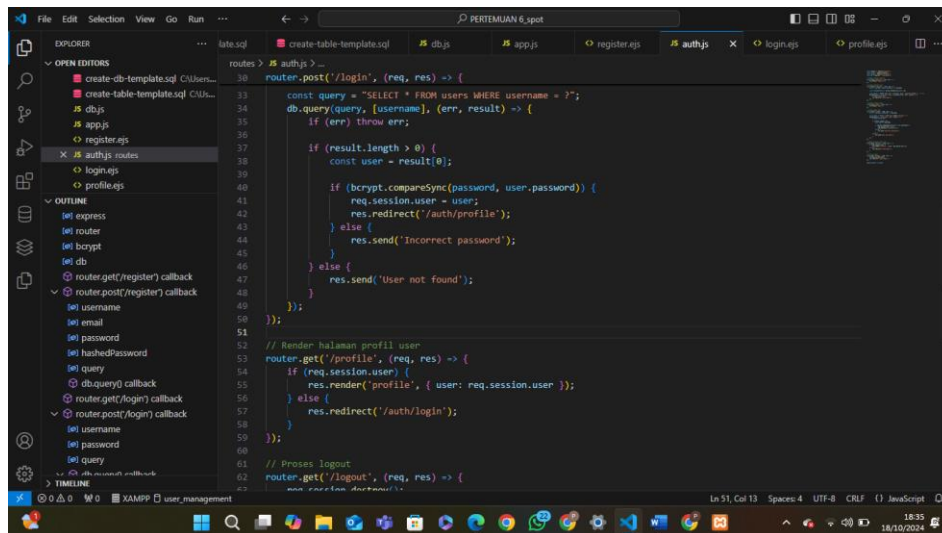
Secara keseluruhan, kode ini menghasilkan halaman profil yang menyambut pengguna dengan username dan email mereka, serta menyediakan tautan untuk logout. Halaman ini menggunakan templating untuk dinamis menampilkan informasi pengguna.

➤ Dalam folder routes

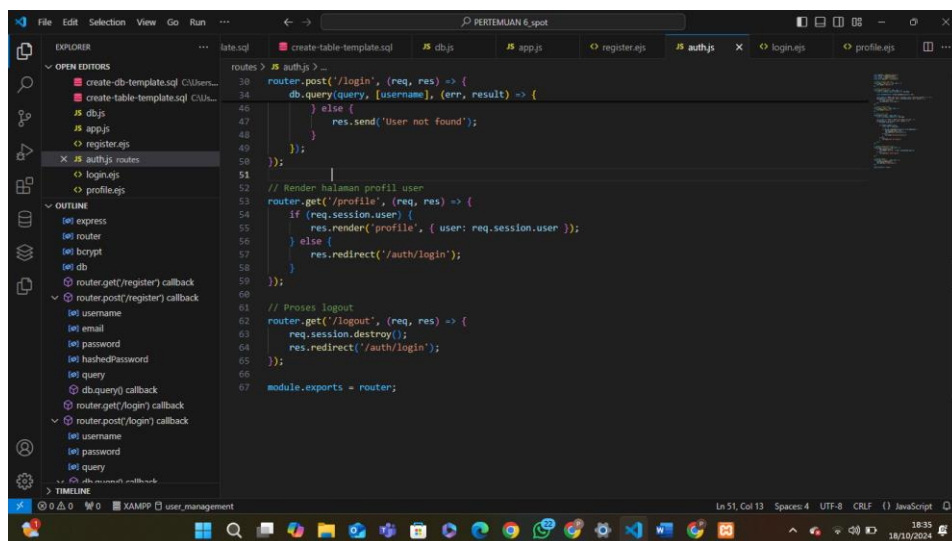
1. Kodingan auth.js



```
1 const express = require('express');
2 const router = express.Router();
3 const bcrypt = require('bcryptjs');
4 const db = require('../config/db');
5
6 // Render halaman register
7 router.get('/register', (req, res) => {
8   res.render('register');
9 });
10
11 // Proses register user
12 router.post('/register', (req, res) => {
13   const { username, email, password } = req.body;
14
15   const hashedPassword = bcrypt.hashSync(password, 10);
16
17   const query = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";
18   db.query(query, [username, email, hashedPassword], (err, result) => {
19     if (err) throw err;
20     res.redirect('/auth/login');
21   });
22 });
23
24 // Render halaman login
25 router.get('/login', (req, res) => {
26   res.render('login');
27 });
28
29 // Proses login user
30 router.post('/login', (req, res) => {
31   const { username, password } = req.body;
```



```
32
33   router.post('/login', (req, res) => {
34     const query = "SELECT * FROM users WHERE username = ?";
35     db.query(query, [username], (err, result) => {
36       if (err) throw err;
37
38       if (result.length > 0) {
39         const user = result[0];
40
41         if (bcrypt.compareSync(password, user.password)) {
42           req.session.user = user;
43           res.redirect('/auth/profile');
44         } else {
45           res.send('Incorrect password');
46         }
47       } else {
48         res.send('User not found');
49       }
50     });
51   });
52
53   // Render halaman profil user
54   router.get('/profile', (req, res) => {
55     if (req.session.user) {
56       res.render('profile', { user: req.session.user });
57     } else {
58       res.redirect('/auth/login');
59     }
60   });
61
62   // Proses logout
63   router.get('/logout', (req, res) => {
64     req.session.destroy();
65     res.redirect('/auth/login');
```



```
66
67   module.exports = router;
```

Penjelasan kodingan:

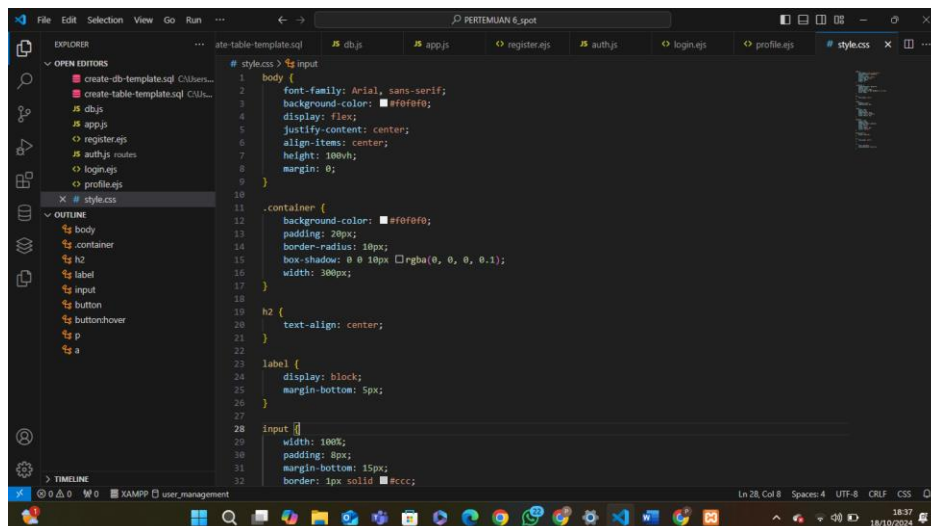
- Import modul
 - `const express = require('express');`: Mengimpor modul Express, yang merupakan framework untuk aplikasi web di Node.js.
 - `const router = express.Router();`: Membuat instance router untuk mendefinisikan rute-rute aplikasi.
 - `const bcrypt = require('bcryptjs');`: Mengimpor bcrypt, sebuah library untuk melakukan hashing password.
 - `const db = require('../config/db');`: Mengimpor konfigurasi database yang diperlukan untuk menjalankan kueri SQL.
- Render Halaman Registrasi:
 - `router.get('/register', (req, res) => {...});`: Mendefinisikan rute untuk menampilkan halaman registrasi. Ketika pengguna mengakses /register, server akan merender halaman register.
- Proses Registrasi Pengguna
 - `router.post('/register', (req, res) => {...});`: Mendefinisikan rute untuk memproses data registrasi yang dikirim melalui formulir. Data username, email, dan password diambil dari `req.body`.
 - `const hashedPassword = bcrypt.hashSync(password, 10);`: Meng-hash password yang dimasukkan pengguna menggunakan bcrypt, dengan tingkat kompleksitas 10.
 - `const query = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";`: Mempersiapkan kueri SQL untuk memasukkan data pengguna baru ke dalam tabel users.
 - `db.query(query, [username, email, hashedPassword], (err, result) => {...});`: Menjalankan kueri. Jika berhasil, pengguna akan diarahkan ke halaman login.
- Render Halaman Login:
 - `router.get('/login', (req, res) => {...});`: Mendefinisikan rute untuk menampilkan halaman login. Ketika pengguna mengakses /login, server akan merender halaman login.
- Proses Login Pengguna:
 - `router.post('/login', (req, res) => {...});`: Mendefinisikan rute untuk memproses data login. Data username dan password diambil dari `req.body`.
 - `const query = "SELECT * FROM users WHERE username = ?";`: Mempersiapkan kueri SQL untuk mencari pengguna berdasarkan username.
 - `db.query(query, [username], (err, result) => {...});`: Menjalankan kueri. Jika pengguna ditemukan, password yang dimasukkan dibandingkan dengan password yang di-hash di database menggunakan `bcrypt.compareSync`. Jika password cocok, sesi pengguna disimpan, dan mereka diarahkan ke halaman profil. Jika tidak, pesan kesalahan ditampilkan.
- Render Halaman Profil Pengguna:

- `router.get('/profile', (req, res) => {...});`: Mendefinisikan rute untuk menampilkan halaman profil. Jika sesi pengguna ada, halaman profil dirender dengan data pengguna. Jika tidak, pengguna diarahkan kembali ke halaman login.
- Proses Logout:
 - `router.get('/logout', (req, res) => {...});`: Mendefinisikan rute untuk logout. Sesi pengguna dihapus, dan mereka diarahkan kembali ke halaman login.
- Ekspor Router:
 - `module.exports = router;`: Mengekspor router untuk digunakan dalam aplikasi utama.

Secara keseluruhan, kode ini menangani pendaftaran, login, profil, dan logout pengguna dengan menyimpan dan memverifikasi data pengguna dalam database.

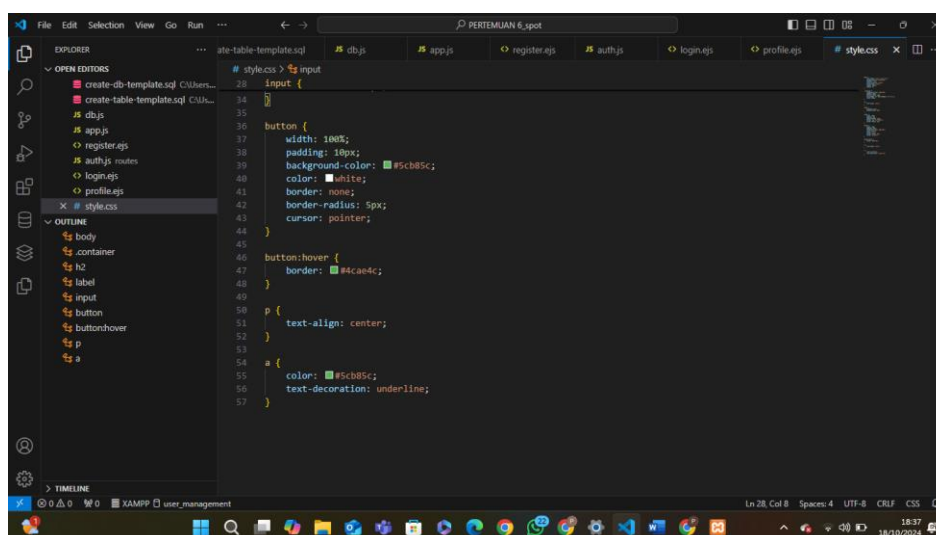
➤ style.css

Untuk kodingan CSS (Cascading Style Sheets) digunakan untuk mengatur dan mengubah tampilan visual halaman web agar web terlihat agar lebih menarik.



```

1 body {
2   font-family: Arial, sans-serif;
3   background-color: #f0f0f0;
4   display: flex;
5   justify-content: center;
6   align-items: center;
7   height: 100vh;
8   margin: 0;
9 }
10
11 .container {
12   background-color: #f0f0f0;
13   padding: 20px;
14   border-radius: 10px;
15   box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
16   width: 300px;
17 }
18
19 h2 {
20   text-align: center;
21 }
22
23 label {
24   display: block;
25   margin-bottom: 5px;
26 }
27
28 input {
29   width: 100%;
30   padding: 8px;
31   margin-bottom: 15px;
32   border: 1px solid #ccc;
  
```

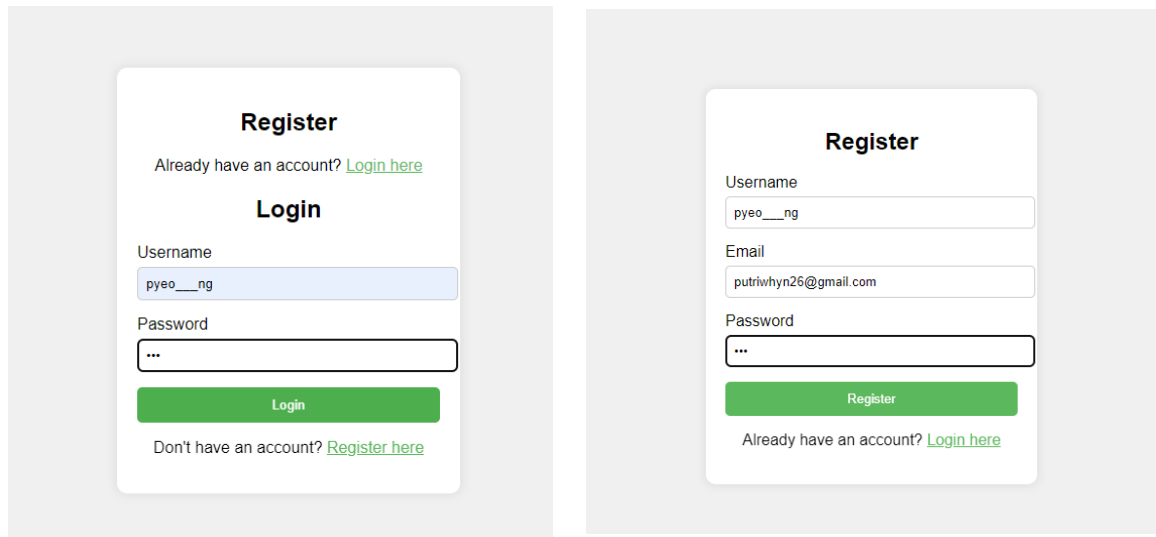


```

33 }
34
35 button {
36   width: 100%;
37   padding: 10px;
38   background-color: #5cb85c;
39   color: white;
40   border: none;
41   border-radius: 5px;
42   cursor: pointer;
43 }
44
45 button:hover {
46   border: 2px solid #4cae4c;
47 }
48
49 p {
50   text-align: center;
51 }
52
53 a {
54   color: #5cb85c;
55   text-decoration: underline;
56 }
57
  
```

➤ Tampilan:

Setelah dijalankan di server localhost:3000 maka akan muncul seperti gambar dibawah tersebut. Ketika kita mencoba untuk login tanpa memiliki akun, akan muncul kehalaman “user not found”. Ini berarti kita harus registrasi terlebih dahulu setelah ter register maka login kembali, maka akan masuk ke halaman profile. Di halaman profile nanti akan ada tampilan untuk logout, lalu ketika kita mengklik logout, maka tampilan akan kembali kehalaman login kembali.



The image displays two side-by-side screenshots of a web application's authentication interface. The left screenshot shows a 'Login' form with the title 'Login' and a 'Register here' link. The right screenshot shows a 'Register' form with the title 'Register' and a 'Login here' link. Both forms have a green button at the bottom.

Left Screenshot (Login Form):

- Title: **Login**
- Link: [Register here](#)
- Fields: Username (pyeo__ng), Password (masked with ...)
- Button: **Login**
- Link: [Don't have an account? Register here](#)

Right Screenshot (Register Form):

- Title: **Register**
- Fields: Username (pyeo__ng), Email (putriwhyn26@gmail.com), Password (masked with ...)
- Button: **Register**
- Link: [Already have an account? Login here](#)

IV. KESIMPULAN

Session merupakan sebuah mekanisme dalam aplikasi web yang berfungsi untuk menyimpan data pengguna secara sementara selama mereka berinteraksi dengan aplikasi. Dalam sesi pembelajaran ini, kita telah melihat bagaimana session digunakan untuk menyimpan data pengguna setelah mereka berhasil login. Saat pengguna melakukan registrasi dan login, informasi mereka disimpan dalam session (`req.session.user`), yang memungkinkan mereka mengakses halaman profil dan merasakan pengalaman yang dipersonalisasi. Apabila pengguna mencoba mengakses halaman tertentu tanpa login, middleware yang diterapkan akan memeriksa status session dan mengarahkan mereka kembali ke halaman login jika belum terautentikasi. Hal ini menambahkan lapisan keamanan, memastikan bahwa hanya pengguna terdaftar yang bisa mengakses informasi pribadi mereka.

Dengan memahami dan menerapkan konsep session, pengembang dapat membuat aplikasi web yang lebih dinamis dan responsif, serta memberikan pengalaman pengguna yang lebih baik melalui pengelolaan login dan data pengguna yang efisien.