

Laporan Praktikum
Mata Kuliah Pemrograman Web



Pertemuan 7. Praktikum 7

“SESSION”

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

Disusun Oleh :

Nuril Khairiyah

2300231

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA

2024

I. PENDAHULUAN

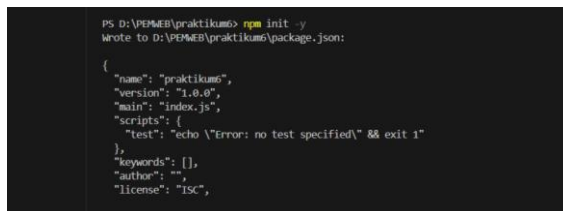
Session adalah mekanisme yang digunakan dalam aplikasi web untuk menyimpan data pengguna secara sementara selama mereka berinteraksi dengan aplikasi. Fungsinya adalah

untuk mempertahankan informasi pengguna di antara permintaan HTTP yang berbeda, memungkinkan aplikasi untuk "mengingat" pengguna selama sesi berlangsung. Dalam Node.js, session digunakan untuk melacak dan menyimpan status pengguna agar data seperti informasi login, preferensi atau pengaturan tertentu tetap tersedia dan konsisten sepanjang penggunaan aplikasi tersebut meskipun terjadi beberapa permintaan atau navigasi halaman.

II. ALAT DAN BAHAN

- Visual Studio Code
- Chrome
- Laptop
- Node.js
- Xampp
- MySQL

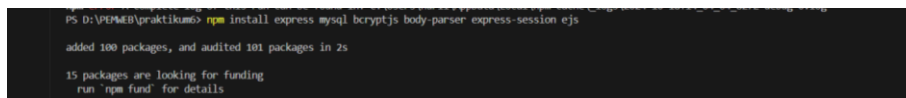
III. PENJELASAN



```
PS D:\PEMWEB\praktikum> npm init -y
Wrote to D:\PEMWEB\praktikum\package.json:

{
  "name": "praktikum",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
}
```

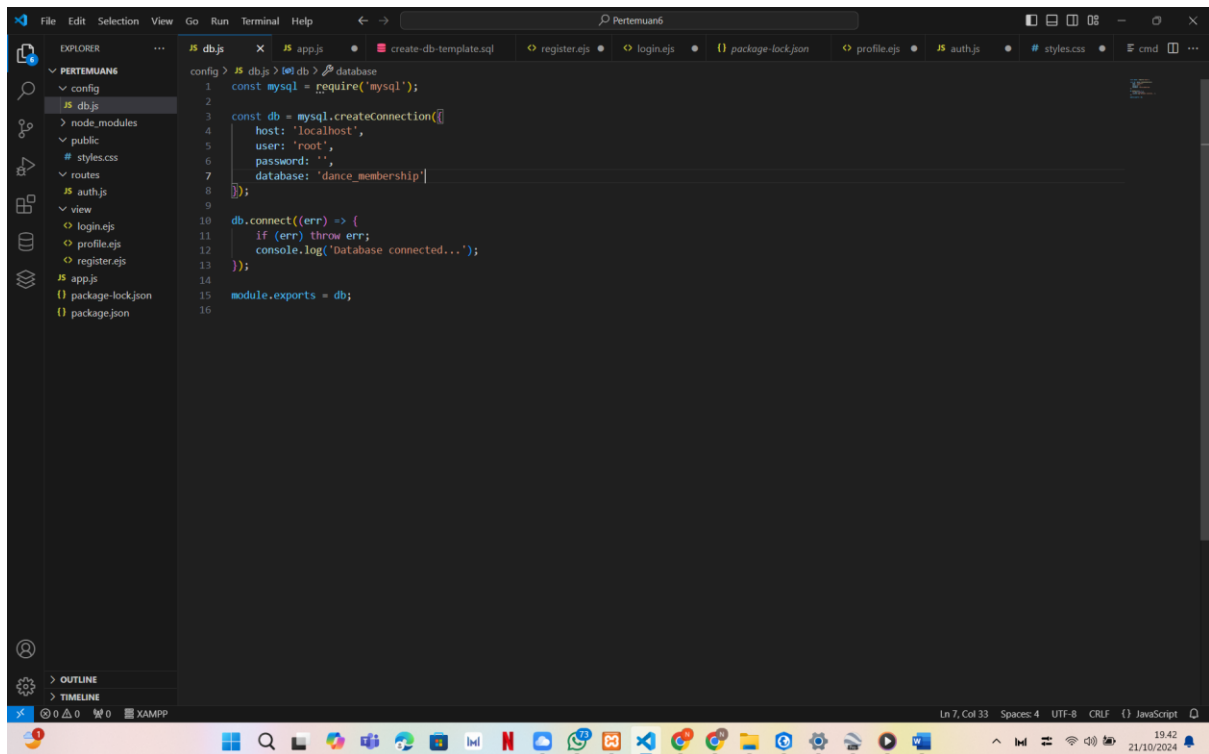
Perintah `npm init -y` digunakan untuk menginisialisasi proyek Node.js dengan membuat file `package.json` secara otomatis.



```
PS D:\PEMWEB\praktikum> npm install express mysql bcryptjs body-parser express-session ejs
added 100 packages, and audited 101 packages in 2s

15 packages are looking for funding
run 'npm fund' for details
```

Perintah ini digunakan untuk menginstal beberapa package atau modul Node.js yang diperlukan dalam sebuah proyek. Yang di install yaitu `express`, `mysql`, `bcryptjs`, `body-parser` dan `express-session`



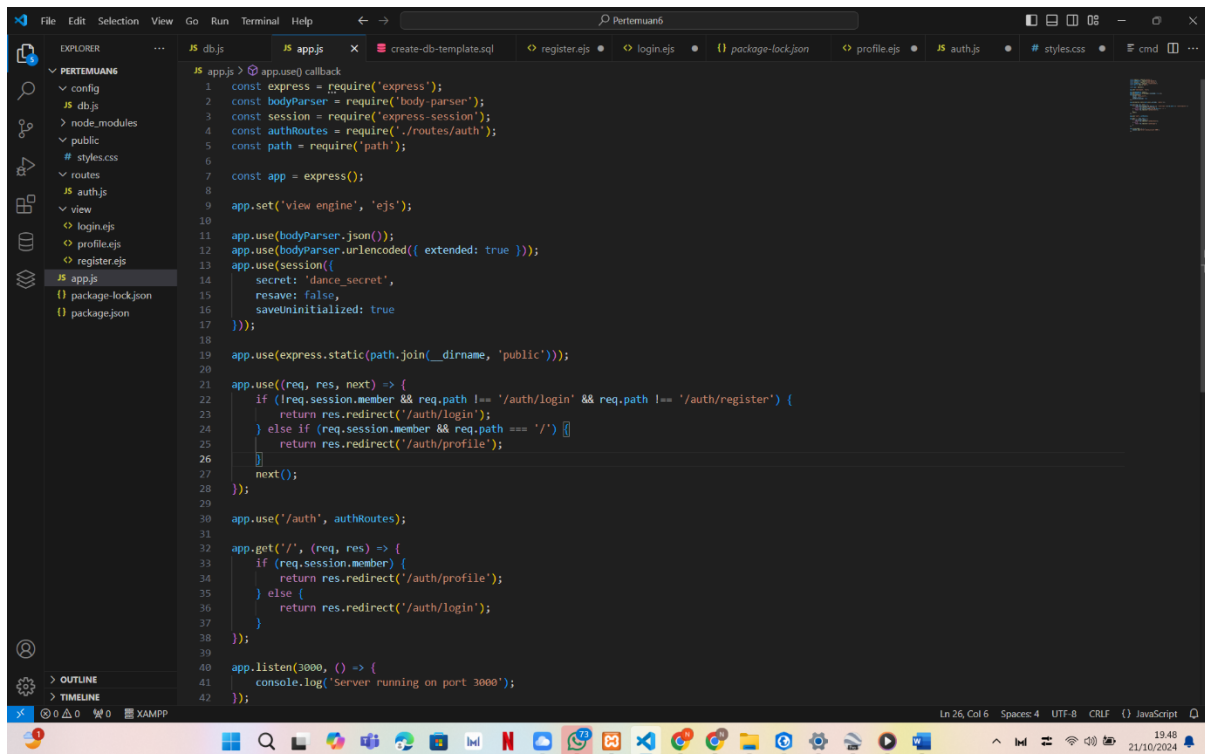
The screenshot shows a VS Code editor window with a project named 'Pertemuan6'. The Explorer sidebar on the left shows the file structure: 'PERTEMUAN6' (containing 'config', 'db.js', 'node_modules', 'public', 'styles.css', 'routes', 'auth.js', 'view', 'login.ejs', 'profile.ejs', 'register.ejs'), 'app.js', 'package-lock.json', and 'package.json'. The main editor area displays the content of 'db.js' with the following code:

```
1 const mysql = require('mysql');
2
3 const db = mysql.createConnection({
4   host: 'localhost',
5   user: 'root',
6   password: '',
7   database: 'dance_membership'
8 });
9
10 db.connect((err) => {
11   if (err) throw err;
12   console.log('Database connected...');
13 });
14
15 module.exports = db;
```

The status bar at the bottom indicates 'Ln 7, Col 33', 'Spaces: 4', 'UTF-8', 'CRLF', and 'JavaScript'.

Dalam kode tersebut, modul `mysql` diimpor menggunakan `require('mysql')`. Ini adalah modul yang menyediakan kemampuan untuk berinteraksi dengan MySQL dari aplikasi Node.js. Kemudian, dibuat sebuah koneksi database dengan menggunakan `mysql.createConnection`, di mana berbagai parameter untuk koneksi database ditentukan, seperti host, user, password dan database.

- host: localhost menunjukkan bahwa server database yang digunakan berada di mesin lokal.
- user: root adalah nama pengguna MySQL default yang sering digunakan untuk koneksi database.
- password: dikosongkan, artinya tidak ada kata sandi yang diperlukan untuk koneksi ini (tergantung pada pengaturan MySQL lokal kamu).
- database: dance_membership menunjukkan nama database yang ingin diakses, yang dalam hal ini adalah dance_membership.



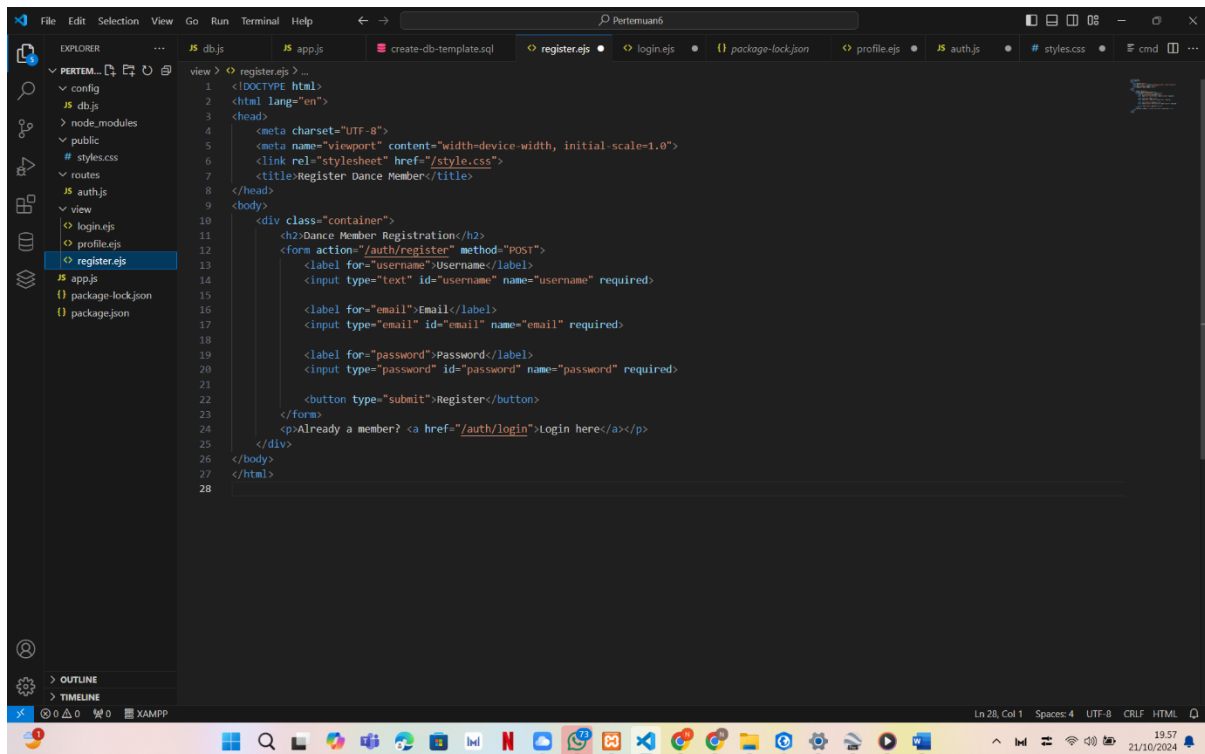
```
1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const session = require('express-session');
4 const authRoutes = require('./routes/auth');
5 const path = require('path');
6
7 const app = express();
8
9 app.set('view engine', 'ejs');
10
11 app.use(bodyParser.json());
12 app.use(bodyParser.urlencoded({ extended: true }));
13 app.use(session({
14   secret: 'dance_secret',
15   resave: false,
16   saveUninitialized: true
17 }));
18
19 app.use(express.static(path.join(__dirname, 'public')));
20
21 app.use((req, res, next) => {
22   if (!req.session.member && req.path !== '/auth/login' && req.path !== '/auth/register') {
23     return res.redirect('/auth/login');
24   } else if (req.session.member && req.path === '/') {
25     return res.redirect('/auth/profile');
26   }
27   next();
28 });
29
30 app.use('/auth', authRoutes);
31
32 app.get('/', (req, res) => {
33   if (req.session.member) {
34     return res.redirect('/auth/profile');
35   } else {
36     return res.redirect('/auth/login');
37   }
38 });
39
40 app.listen(3000, () => {
41   console.log('Server running on port 3000');
42 });
```

kode JavaScript di file app.js, yang merupakan file utama untuk mengatur aplikasi berbasis Node.js menggunakan Express. Dalam kode ini, beberapa modul dan middleware digunakan untuk mengelola permintaan HTTP dan sesi pengguna, serta mengatur rendering tampilan. Beberapa modul diimpor, termasuk express, body-parser, express-session, path dan rute khusus dari file auth.js. Kemudian, aplikasi Express dibuat menggunakan express() dan view engine diatur menjadi EJS untuk memungkinkan rendering tampilan menggunakan template EJS.

Dua middleware bodyParser digunakan untuk mengurai data yang dikirim melalui form (dalam format JSON dan URL-encoded). Middleware sesi (express-session) digunakan untuk menangani manajemen sesi pengguna, dengan konfigurasi seperti secret, resave dan saveUninitialized. Kode berikutnya adalah middleware untuk memeriksa status login pengguna. Jika pengguna belum login (sesi tidak ada), dan mencoba mengakses halaman lain selain /auth/login atau /auth/register pengguna akan diarahkan ke halaman login. Jika pengguna sudah login dan mencoba mengakses root (/) maka akan diarahkan ke halaman profil (/auth/profile).

Selanjutnya, rute-rute dari auth.js ditambahkan di bawah jalur /auth. Pada rute root (/) terdapat logika untuk mengarahkan pengguna ke halaman profil jika sudah login atau ke halaman login jika belum.

Terakhir server diatur untuk berjalan pada port 3000 dan pesan “Server running on port 3000” dicetak di terminal ketika server berhasil dijalankan.

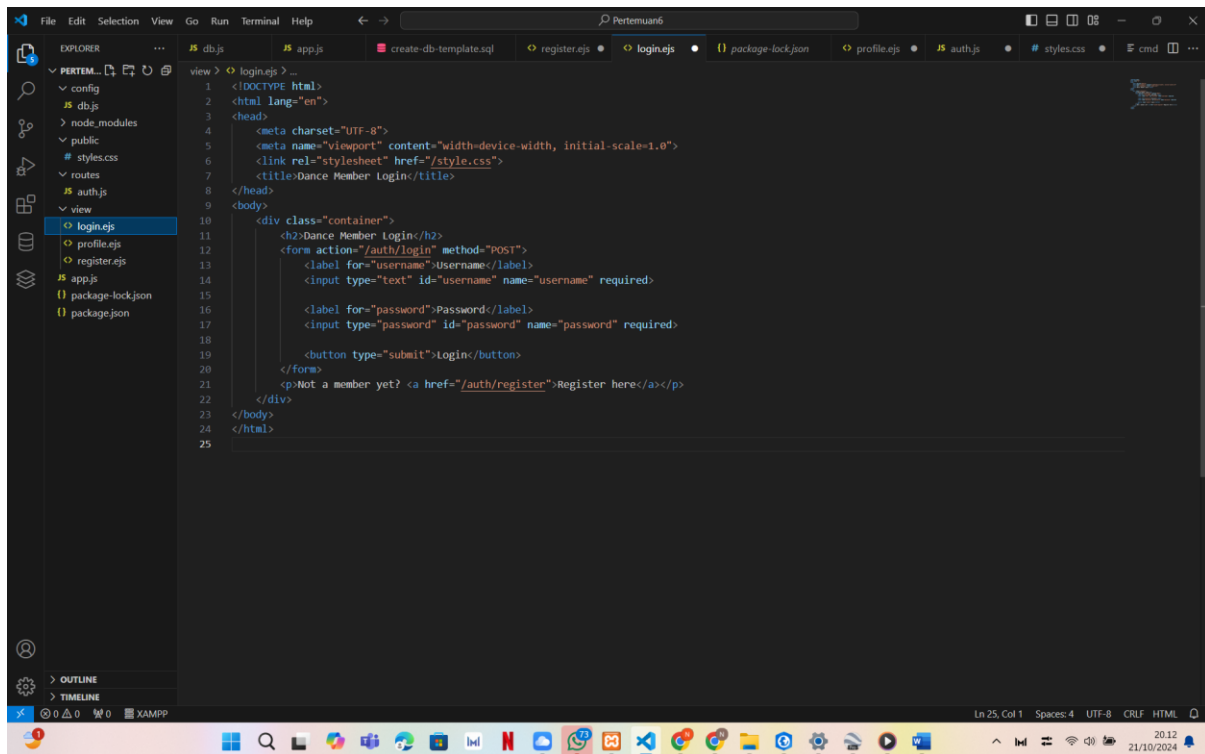


```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="/style.css">
7   <title>Register Dance Member</title>
8 </head>
9 <body>
10  <div class="container">
11    <h2>Dance Member Registration</h2>
12    <form action="/auth/register" method="post">
13      <label for="username">Username</label>
14      <input type="text" id="username" name="username" 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 a member? <a href="/auth/login">Login here</a></p>
25  </div>
26 </body>
27 </html>
28
```

File template register.ejs yang digunakan untuk menampilkan halaman registrasi anggota. File ini ditulis dalam format HTML dengan menggunakan sintaks EJS dan bertujuan untuk menyediakan form pendaftaran bagi pengguna baru yang ingin mendaftar sebagai anggota.

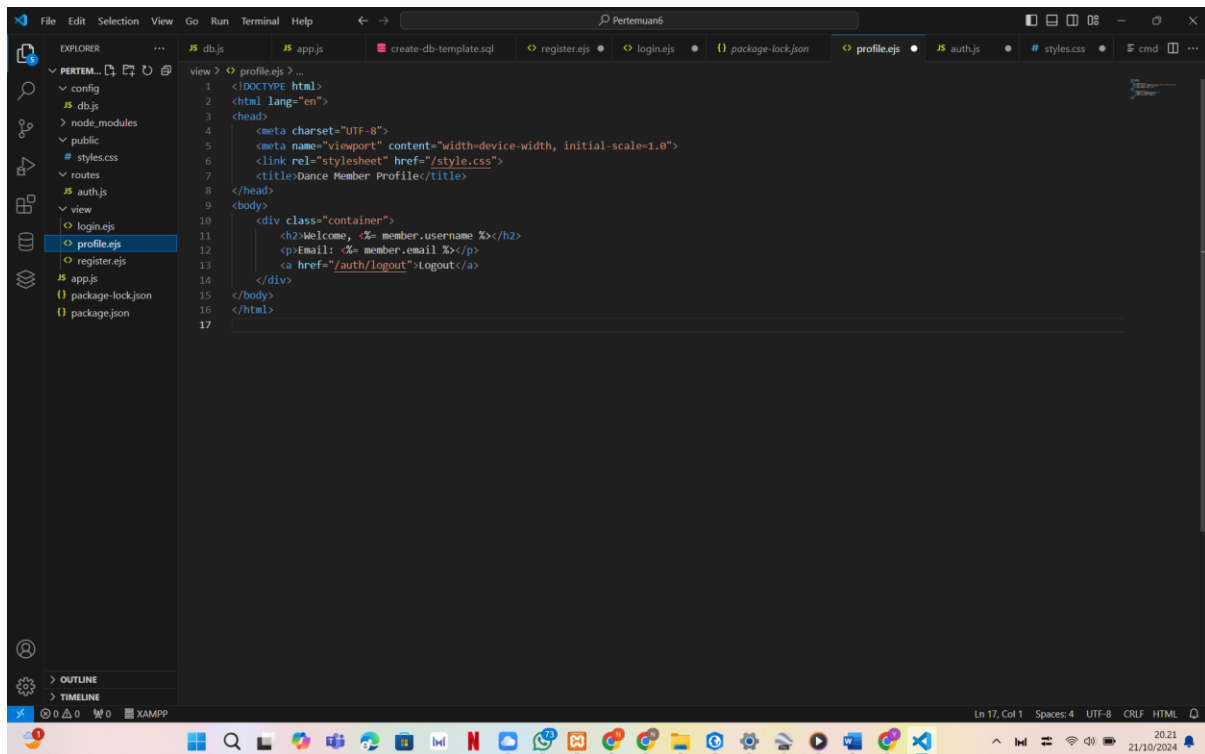
Di dalam kode, elemen-elemen dasar HTML disusun dengan struktur yang rapi. Di bagian `<head>` ada beberapa elemen penting seperti deklarasi dokumen HTML dengan tipe `<!DOCTYPE html>` dan elemen `<meta>` yang mengatur pengaturan responsif halaman agar sesuai dengan lebar layar perangkat. Terdapat juga `<link>` yang menghubungkan file CSS eksternal (style.css) untuk memberikan gaya pada halaman. Pada bagian `<body>`, kode menggunakan `<div>` dengan kelas container untuk membungkus seluruh konten halaman, yang memberikan struktur dan memungkinkan penggunaan styling lebih mudah. Di dalamnya, terdapat judul 'Dance Member Registration' yang ditampilkan menggunakan tag `<h2>`.

Formulir pendaftaran dibuat menggunakan elemen `<form>` yang mengatur action sebagai `/auth/register` dengan metode POST. Ini berarti ketika form dikirimkan data akan dikirim ke rute `/auth/register` di server menggunakan metode HTTP POST. Di dalam form tersebut ada tiga input utama: username, email, dan password, masing-masing dilengkapi dengan label untuk memudahkan pengguna dalam mengisi form. Input username, email dan password masing-masing memiliki atribut `required` yang berarti pengguna harus mengisinya sebelum dapat mengirimkan formulir. Setiap input juga diberi tipe data yang sesuai (text, email dan password) untuk memastikan validasi yang benar. Di bagian bawah form terdapat tombol submit dengan label 'Register' yang digunakan untuk mengirimkan data form.



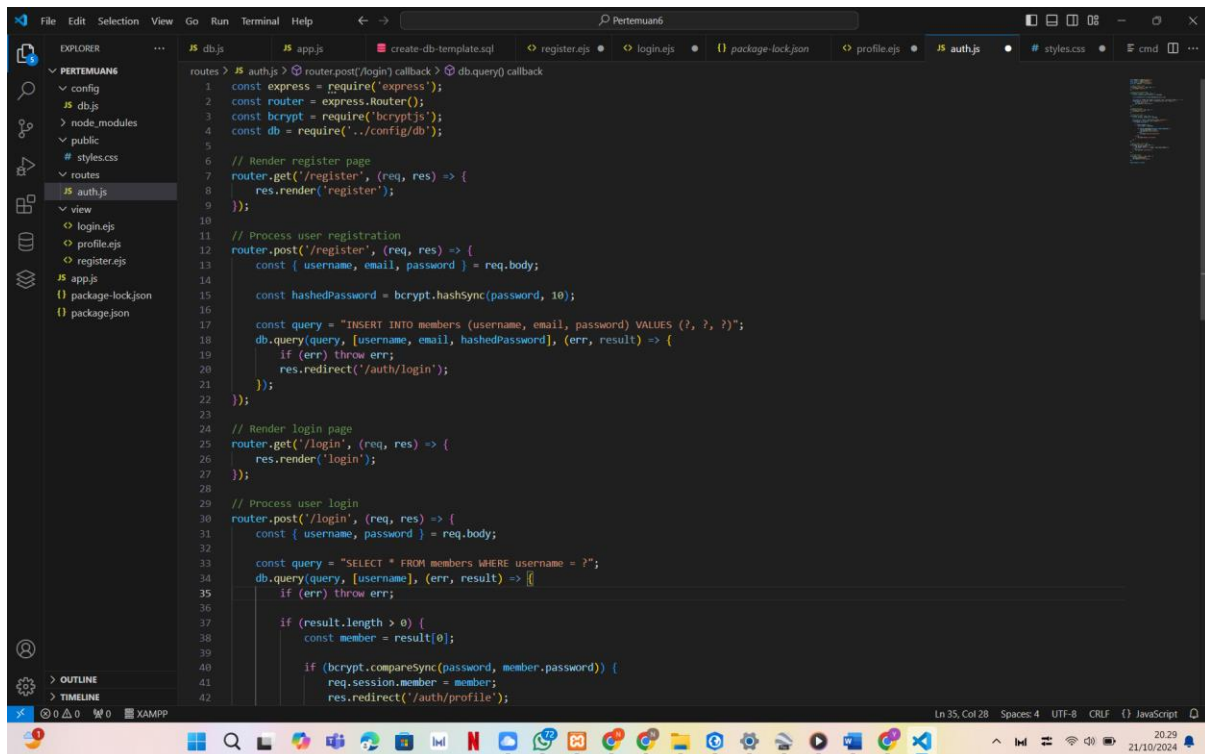
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="/style.css">
7   <title>Dance Member Login</title>
8 </head>
9 <body>
10   <div class="container">
11     <h2>Dance Member 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>Not a member yet? <a href="/auth/register">Register here</a></p>
22   </div>
23 </body>
24 </html>
25
```

Bagian pertama dari kode adalah tag `<!DOCTYPE html>` yang memberi tahu browser bahwa dokumen ini adalah dokumen HTML5. Setelah itu terdapat tag `<html>` yang menandakan awal dari halaman HTML. Dalam tag `<head>` beberapa metadata disertakan, seperti pengaturan charset untuk UTF-8 dan viewport agar tampilan halaman responsif di perangkat seluler. Link ke file CSS eksternal (styles.css) juga ditambahkan untuk menerapkan gaya pada halaman. Di dalam tag `<body>` terdapat div dengan class `container` yang tampaknya digunakan untuk mengatur tata letak halaman login. Judul halaman ditampilkan dengan tag `<h2>` yang berbunyi “Dance Member Login”. Ada form dengan action menuju `/auth/login` menggunakan metode POST yang berarti data yang dimasukkan oleh pengguna akan dikirim ke URL tersebut untuk diproses. Dalam form tersebut terdapat dua input field, satu untuk username dan satu lagi untuk password. Keduanya memiliki atribut `required` yang memaksa pengguna untuk mengisinya sebelum mengirimkan form ada juga tombol login yang dibuat menggunakan elemen `<button type="submit">` yang berfungsi untuk mengirimkan data saat diklik.



```
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>Dance Member Profile</title>
8 </head>
9 <body>
10   <div class="container">
11     <h2>Welcome, <%= member.username %></h2>
12     <p>Email: <%= member.email %></p>
13     <a href="/auth/logout">Logout</a>
14   </div>
15 </body>
16 </html>
17
```

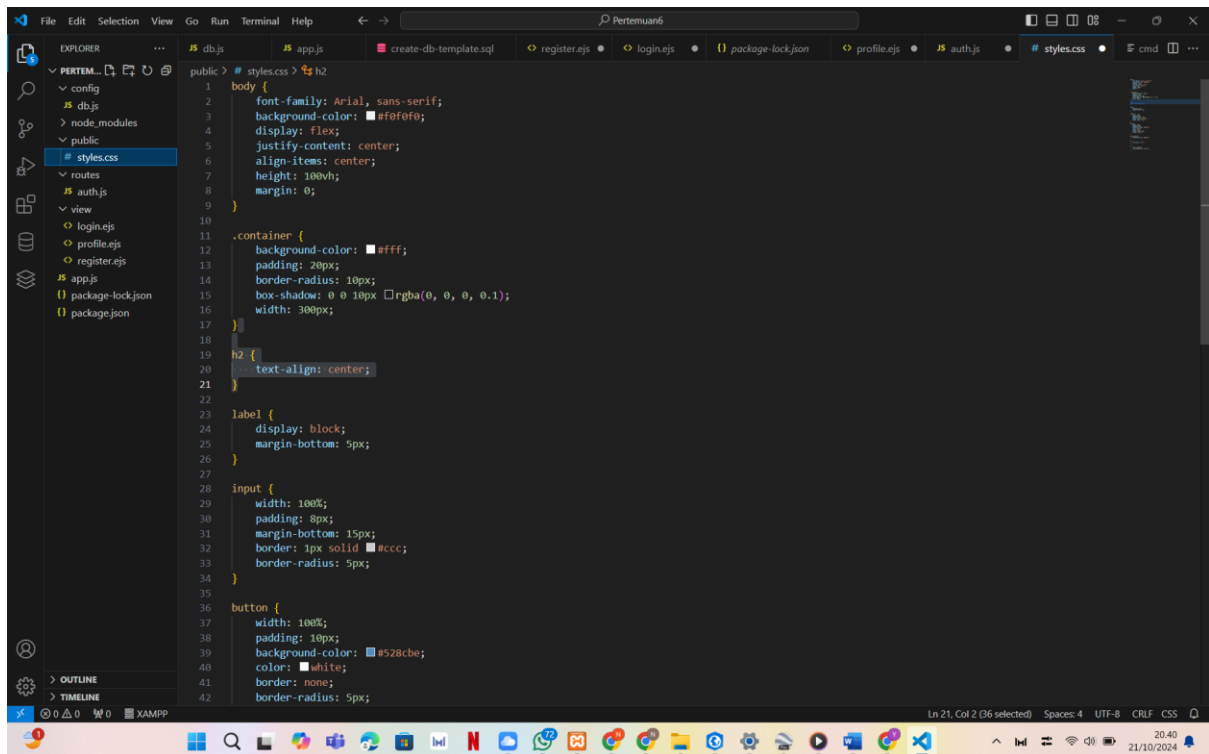
File ini menggunakan format EJS (Embedded JavaScript yang memungkinkan kita untuk menyisipkan kode JavaScript di dalam HTML untuk menghasilkan halaman dinamis. Struktur dasar dimulai dengan tag `<!DOCTYPE html>` dan tag `<html>` yang menunjukkan bahwa ini adalah halaman HTML5. Pada bagian `<head>` ada pengaturan charset untuk UTF-8 dan pengaturan viewport agar halaman responsif di berbagai perangkat. Link menuju file CSS eksternal `styles.css` digunakan untuk mengatur tampilan visual halaman. Pada tag `<title>` judul halaman diatur menjadi “Dance Member Profile” yang akan ditampilkan di tab browser. Di dalam tag `<body>` terdapat div dengan class `container` yang berfungsi sebagai pembungkus elemen-elemen konten di dalam halaman. Tag `<h2>` digunakan untuk menampilkan sambutan kepada pengguna yang sudah login. Nama pengguna diambil dari objek `member.username` menggunakan sintaks EJS untuk menyisipkan data dinamis ke dalam HTML. Di bawahnya tag `<p>` menampilkan alamat email pengguna diambil dari objek `member.email`. Terakhir terdapat tautan Logout yang mengarah ke URL `/auth/logout` untuk mengizinkan pengguna keluar dari akun mereka.



```
1 const express = require('express');
2 const router = express.Router();
3 const bcrypt = require('bcryptjs');
4 const db = require('../config/db');
5
6 // Render register page
7 router.get('/register', (req, res) => {
8   res.render('register');
9 });
10
11 // Process user registration
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 members (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 login page
25 router.get('/login', (req, res) => {
26   res.render('login');
27 });
28
29 // Process user login
30 router.post('/login', (req, res) => {
31   const { username, password } = req.body;
32
33   const query = "SELECT * FROM members WHERE username = ?";
34   db.query(query, [username], (err, result) => {
35     if (err) throw err;
36
37     if (result.length > 0) {
38       const member = result[0];
39
40       if (bcrypt.compareSync(password, member.password)) {
41         req.session.member = member;
42         res.redirect('/auth/profile');
```

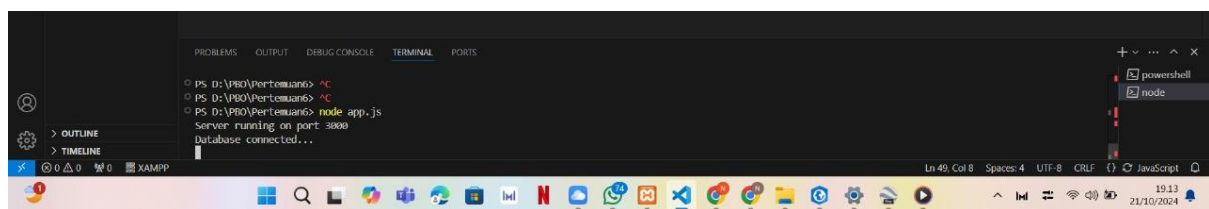
Kode tersebut digunakan untuk menangani rute (routes) otentikasi dalam aplikasi Node.js menggunakan Express. Impor modul pada express yaitu Modul Express digunakan untuk membuat aplikasi server, bcryptjs digunakan untuk melakukan hashing password sehingga password pengguna tidak disimpan dalam teks biasa, db Menghubungkan ke database (ini mungkin merupakan file konfigurasi database lokal), router.get('/register', ...) Rute ini akan menangani permintaan GET ke /register dan merespons dengan merender halaman register. Halaman tersebut akan ditampilkan kepada pengguna saat mereka mengakses URL /register.

router.post(/register, ...) Rute ini menangani permintaan POST ke '/register' yang digunakan untuk memproses registrasi pengguna baru. Data (username, email, password) diambil dari 'req.body' (yang berarti data diambil dari form yang diisi oleh pengguna). Password pengguna di-hash menggunakan bcrypt.hashSync dengan salt 10, sehingga password yang disimpan dalam database sudah terenkripsi. Setelah itu sebuah query SQL 'INSERT INTO' digunakan untuk menyimpan data pengguna (username, email, hashedPassword) ke dalam tabel members. Setelah registrasi berhasil pengguna diarahkan ke halaman login dengan res.redirect('/auth/login'). Router.get(/login, ...) Rute ini akan merender halaman login ketika pengguna mengakses URL /login. router.post(/login, ...): Rute ini menangani permintaan POST untuk login. Data username dan password diambil dari req.body. Query SQL SELECT FROM members WHERE username = ? dijalankan untuk mencari pengguna dengan username yang diberikan. jika username ditemukan (result.length > 0), maka password pengguna yang diinputkan dibandingkan dengan password yang tersimpan di database menggunakan bcrypt.compareSync. Jika password cocok, sesi pengguna (req.session.member) akan diset dengan data pengguna, dan mereka diarahkan ke halaman profil (res.redirect('/auth/profile')). Jika tidak cocok login gagal meskipun tidak ada penanganan eksplisit untuk itu di potongan kode ini.



```
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: #fff;
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   border-radius: 5px;
34 }
35
36 button {
37   width: 100%;
38   padding: 10px;
39   background-color: #528cbe;
40   color: white;
41   border: none;
42   border-radius: 5px;
```

Berikut adalah styles.css yang digunakan



```
PS D:\P80\pertemuan6> .C
PS D:\P80\pertemuan6> .C
PS D:\P80\pertemuan6> node app.js
Server running on port 3000
Database connected...
```

Menjalankan server : node app.js

HASIL

