

Laporan Praktikum
Mata Kuliah Pemrograman Berorientasi Objek (PBO)



Pertemuan 7. Praktikum 7
“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. Buat struktur folder terlebih dahulu seperti pada gambar di bawah:

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

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

1. Inisialisasi project Node.js:

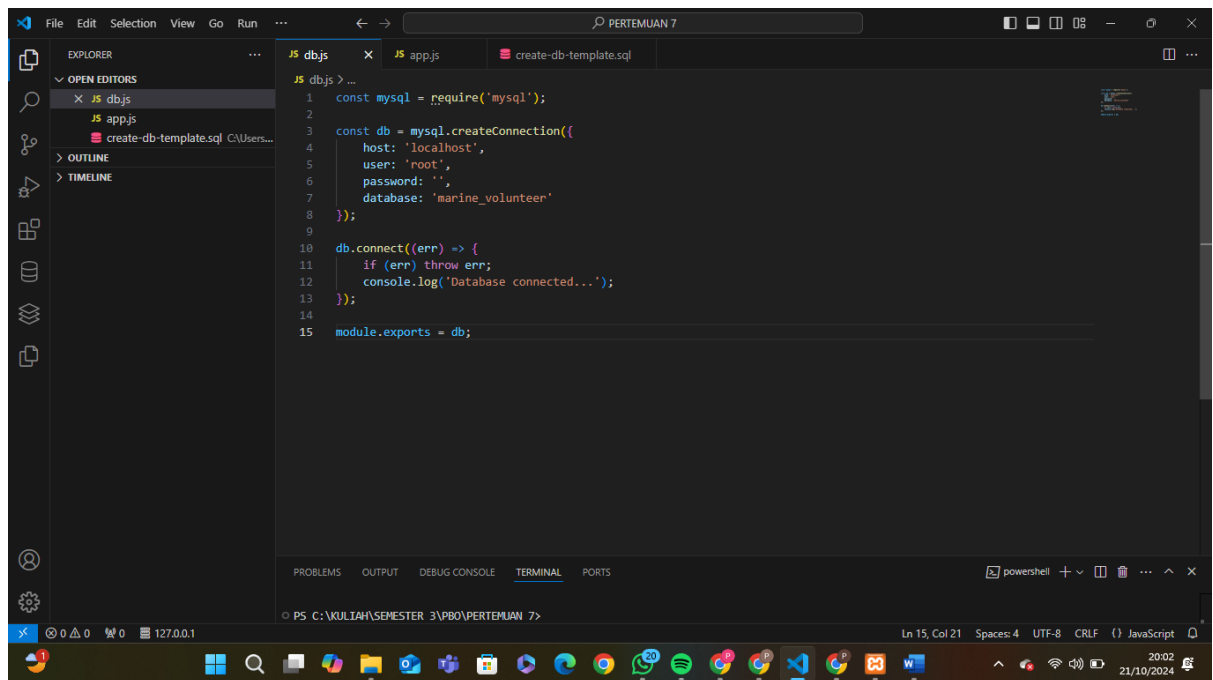
```
npm init -y
```

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



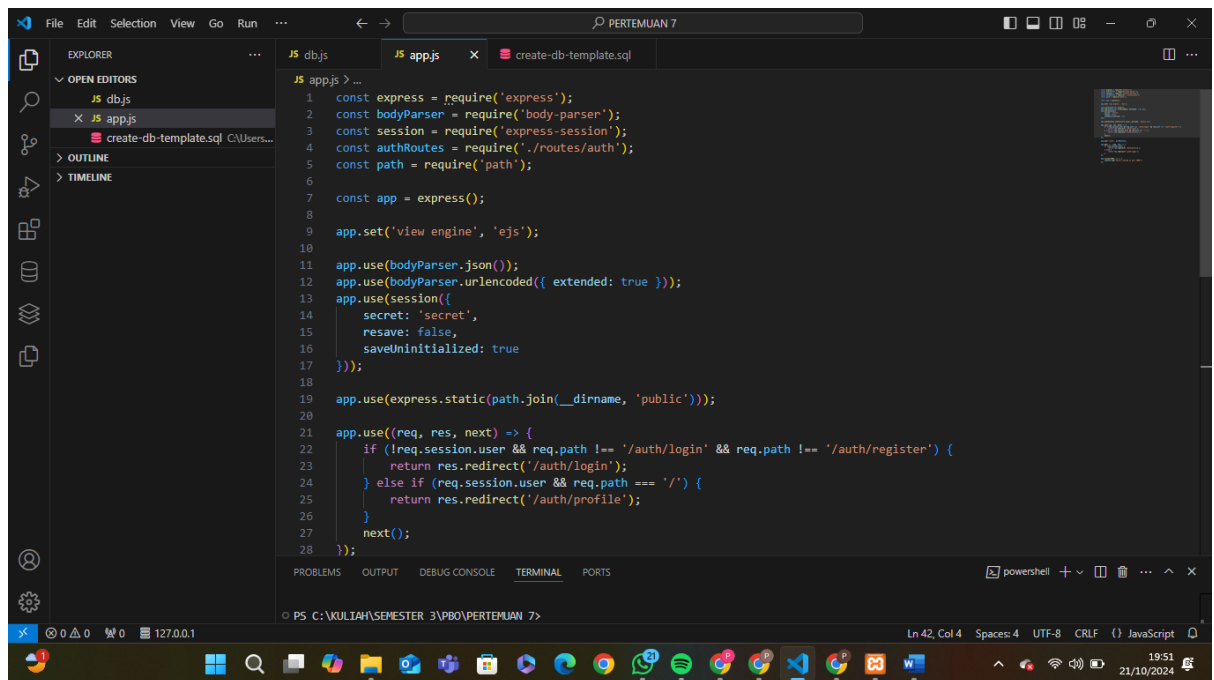
```
1  const mysql = require('mysql');
2
3  const db = mysql.createConnection({
4    host: 'localhost',
5    user: 'root',
6    password: '',
7    database: 'marine_volunteer'
8  });
9
10 db.connect((err) => {
11   if (err) throw err;
12   console.log('Database connected...');
13 });
14
15 module.exports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

Penjelasan kodingan:

Kode tersebut 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 `marine_volunteer`.

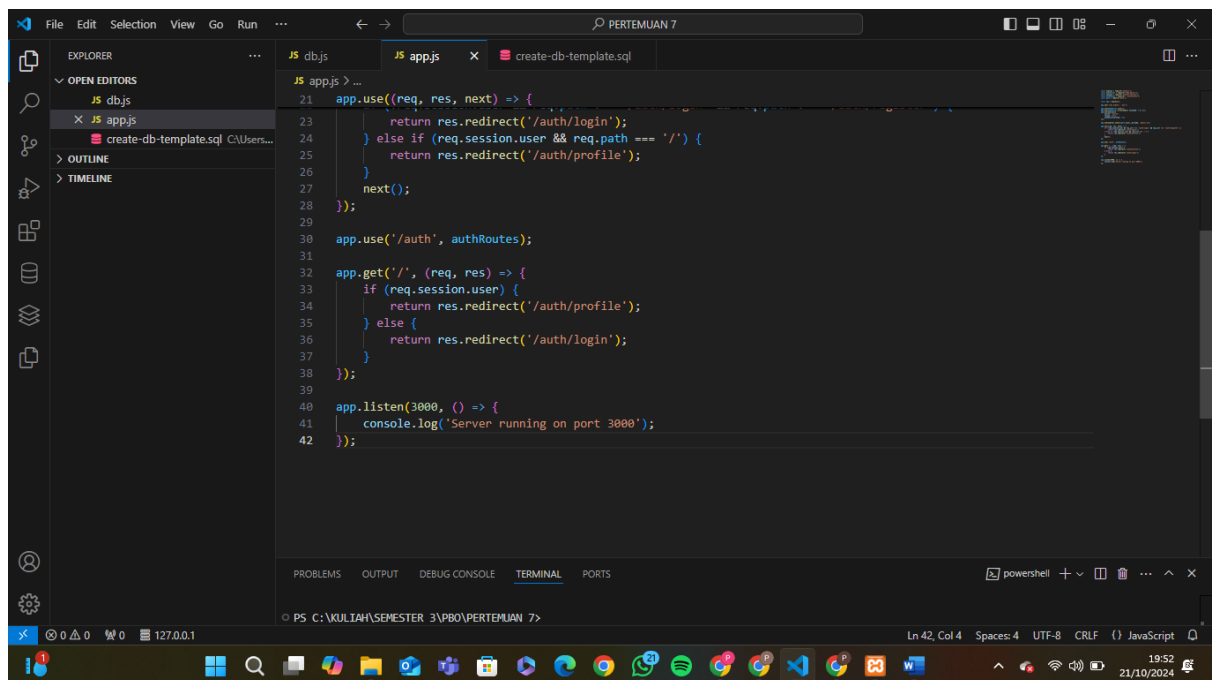
Setelah koneksi dikonfigurasi, fungsi `db.connect()` dipanggil untuk mencoba menyambungkan ke database. Jika terjadi kesalahan, misalnya koneksi gagal atau parameter tidak tepat, kesalahan tersebut akan dilemparkan dengan `throw err`. Jika koneksi berhasil, pesan "Database connected..." akan dicetak 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



The screenshot shows the VS Code editor with the file explorer on the left. The 'app.js' file is open in the editor. The code includes imports for express, body-parser, session, authRoutes, and path. It sets up the Express app with EJS as the view engine, uses body-parser and session middleware, and sets static files. A conditional redirect logic is implemented for the root path based on session status.

```
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: '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.user && req.path !== '/auth/login' && req.path !== '/auth/register') {
23     return res.redirect('/auth/login');
24   } else if (req.session.user && req.path === '/') {
25     return res.redirect('/auth/profile');
26   }
27   next();
28 });
```



The screenshot shows the continuation of the 'app.js' file. It includes a GET route for the root path that redirects to the profile or login page based on session. The server is then configured to listen on port 3000.

```
21 app.use((req, res, next) => {
22   if (!req.session.user && req.path !== '/auth/login' && req.path !== '/auth/register') {
23     return res.redirect('/auth/login');
24   } else if (req.session.user && 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.user) {
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 });
```

Penjelasan kodingan:

- Mengimpor Modul: Di sini, beberapa modul penting diimpor:
 - express: Framework web untuk Node.js yang memudahkan pengembangan aplikasi web.
 - body-parser: Middleware untuk menguraikan (parse) body dari permintaan HTTP, sehingga bisa mengakses data yang dikirim dari formulir.
 - express-session: Middleware untuk menangani sesi pengguna, memungkinkan penyimpanan data pengguna di server.

- `authRoutes`: Mengimpor rute otentikasi dari file `auth.js` di folder `routes`.
- `path`: Modul bawaan Node.js yang membantu menangani dan mengubah path file javascript

- `const app = express()`: Membuat Aplikasi Express: Baris ini membuat aplikasi Express baru, yang akan digunakan untuk mengatur rute, middleware, dan logika aplikasi.
- Set EJS sebagai template engine
 - `app.set('view engine', 'ejs')`
Menentukan Template Engine: Di sini, EJS (Embedded JavaScript) diatur sebagai template engine. Ini memungkinkan untuk menggunakan file `.ejs` untuk merender halaman HTML dinamis dengan data yang diberikan oleh server.
- Middleware
 - Menguraikan body permintaan dalam format JSON.
 - `bodyParser.urlencoded({ extended: true })`: Menguraikan body permintaan yang dikirim dari formulir HTML. Opsi `extended: true` memungkinkan penguraian data yang lebih kompleks.
- Menggunakan Middleware Sesi: Di sini, middleware sesi diatur dengan opsi:
 - `secret`: String rahasia yang digunakan untuk mengenkripsi sesi.
 - `resave`: Jika `false`, sesi tidak akan disimpan kembali ke penyimpanan sesi jika tidak ada perubahan.
 - `saveUninitialized`: Jika `true`, sesi baru akan disimpan meskipun tidak ada data yang diubah.
- Set static folder
 - `app.use(express.static(path.join(__dirname, 'public')))`
Menentukan Folder Statis: Baris ini menetapkan folder `public` sebagai folder statis. Berkas yang berada di dalam folder ini dapat diakses langsung melalui URL tanpa perlu rute tambahan. Ini sering digunakan untuk menyimpan file CSS, JavaScript, gambar, dan aset lainnya.
- Middleware untuk memeriksa status login
 - Middleware untuk Memeriksa Status Login: Middleware ini memeriksa apakah pengguna sudah login:
 - Jika pengguna tidak memiliki sesi (`!req.session.user`) dan mencoba mengakses rute lain (kecuali login atau register), mereka akan dialihkan ke halaman login.
 - Jika pengguna sudah login (`req.session.user`) dan mencoba mengakses root (`/`), mereka akan dialihkan ke halaman profil.
 - Jika tidak ada kondisi yang terpenuhi, middleware akan memanggil `next()` untuk melanjutkan ke middleware atau rute berikutnya.
- Routes
 - `app.use('/auth', authRoutes);`
Menggunakan Rute Otorisasi: Menetapkan semua rute yang dimulai dengan `/auth` akan diarahkan ke `authRoutes`, yang dikelola dalam file `auth.js`.

- Root route

```
app.get('/', (req, res) => {
  if (req.session.user) {
    return res.redirect('/auth/profile');
  } else {
    return res.redirect('/auth/login');
  }
});
```

-Rute Root: Rute ini menangani permintaan ke URL root (/):
Jika pengguna sudah login, mereka akan dialihkan ke halaman profil.
Jika tidak, mereka akan dialihkan ke halaman login.

- Menjalankan server

```
app.listen(3000, () => {
  console.log('Server running on port 3000');
});
```

Menjalankan Server: Di sini, server diatur untuk mendengarkan permintaan pada port 3000. Ketika server berjalan, akan mencetak pesan ke konsol yang mengonfirmasi bahwa server aktif.

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

- KODINGAN register.ejs

```
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 Volunteer Konservasi Laut</title>
8   <script>
9     function showAlert(message) {
10       alert(message);
11     }
12   </script>
13 </head>
14 <body>
15   <div class="container">
16     <h2>Registrasi Volunteer Konservasi Laut</h2>
17     <form action="/auth/register" method="POST">
18       <label for="username">Username</label>
19       <input type="text" id="username" name="username" required>
20
21       <label for="email">Email</label>
22       <input type="email" id="email" name="email" required>
23
24       <label for="password">Password</label>
25       <input type="password" id="password" name="password" required>
26
27       <button type="submit">Daftar</button>
28     </form>
29     <p>Sudah punya akun? <a href="/auth/login">Login di sini</a></p>
30   </div>
31 </body>
32 </html>
```

Penjelasan kodingan:

- Deklarasi DOCTYPE:

```
<!DOCTYPE html>
```

Baris ini mendeklarasikan bahwa dokumen ini adalah dokumen HTML5.

- Elemen HTML dan Atribut lang:

```
<html lang="en">
```

Elemen <html> adalah elemen akar dari dokumen HTML. Atribut lang="en" menunjukkan bahwa konten halaman ini ditulis dalam bahasa Inggris.

- Bagian <head>:

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<link rel="stylesheet" href="/style.css">
```

```
<title>Register Volunteer Konservasi Laut</title>
```

```
<script>
```

```
function showAlert(message) {
```

```
    alert(message);
```

```
}
```

```
</script>
```

```
</head>
```

Meta Tag:

- <meta charset="UTF-8"> mengatur pengkodean karakter untuk dokumen, memastikan bahwa karakter ditampilkan dengan benar.

- <meta name="viewport" content="width=device-width, initial-scale=1.0"> mengatur tampilan responsif, memastikan bahwa halaman terlihat baik pada perangkat dengan berbagai ukuran layar.

- Link ke CSS:

```
<link rel="stylesheet" href="/style.css">
```

Ini menghubungkan halaman HTML dengan file CSS yang bernama style.css, yang mengatur gaya dan tata letak halaman.

- Judul Halaman:

```
<title>Register Volunteer Konservasi Laut</title>
```

Ini menetapkan judul halaman yang akan ditampilkan di tab browser.

- <script>

```
function showAlert(message) {
```

```
    alert(message);
```

```
}
```

```
</script>
```

Ini mendefinisikan fungsi showAlert yang menerima parameter message dan menampilkan alert dengan pesan tersebut. Fungsi ini bisa digunakan di tempat lain dalam halaman untuk memberi informasi atau peringatan kepada pengguna.

- Bagian <body>:

```
<body>
```

```
<div class="container">
```

```
<h2>Registrasi Volunteer Konservasi Laut</h2>
```

```
<form action="/auth/register" method="POST">
```

```
<label for="username">Username</label>
```



```
<input type="text" id="username" name="username" required>
```

```
<label for="email">Email</label>
```

```
<input type="email" id="email" name="email" required>
```

```
<label for="password">Password</label>
```

```
<input type="password" id="password" name="password" required>
```

```
<button type="submit">Daftar</button>
```

```
</form>
```

```
<p>Sudah punya akun? <a href="/auth/login">Login di sini</a></p>
```

```
</div>
```

```
</body>
```

Elemen ini adalah bagian utama dari dokumen HTML yang berisi semua konten yang akan ditampilkan di halaman web. Semua elemen yang terlihat oleh pengguna berada di dalam tag ini.

- Kontainer:

```
<div class="container">
```

Elemen <div> dengan kelas container digunakan untuk mengelompokkan elemen di dalamnya dan memudahkan penerapan gaya CSS.

- Judul Halaman:

```
- <h2>Registrasi Volunteer Konservasi Laut</h2>
```

Menampilkan judul sekunder di halaman.

- Formulir Pendaftaran:

```
<form action="/auth/register" method="POST">
```

Elemen <form> ini digunakan untuk mengumpulkan data dari pengguna. Atribut action menunjukkan URL tujuan saat formulir disubmit, dan method="POST" menunjukkan bahwa data akan dikirim menggunakan metode POST.

- Label dan Input: Setiap label diikuti dengan elemen input:

```
<label for="username">Username</label>
```

```
<input type="text" id="username" name="username" required>
```

Ini membuat label untuk input pengguna. Atribut for pada label menghubungkannya dengan input berdasarkan ID, dan required memastikan bahwa input tidak boleh kosong saat formulir disubmit. Ada tiga input: untuk username, email, dan password.

- Tombol Submit:

```
<button type="submit">Daftar</button>
```

Tombol ini digunakan untuk mengirimkan formulir.

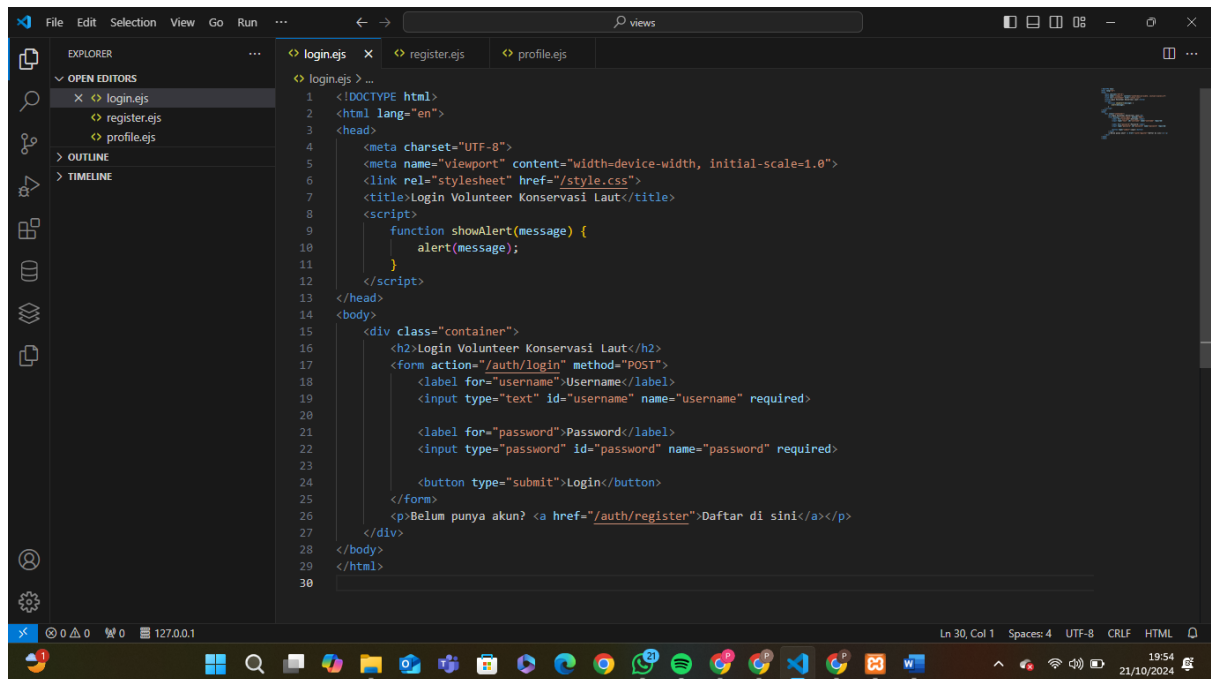
- Link ke Halaman Login:

```
<p>Sudah punya akun? <a href="/auth/login">Login di sini</a></p>
```

Sebuah paragraf yang menyediakan tautan bagi pengguna yang sudah memiliki akun untuk masuk ke halaman login.

Secara keseluruhan, kode tersebut mendefinisikan halaman registrasi untuk menjadi sukarelawan dalam konservasi laut, dengan formulir yang memungkinkan pengguna untuk mendaftar menggunakan username, email, dan password. Ada juga tautan ke halaman login jika pengguna sudah memiliki akun.

- 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>Login Volunteer Konservasi Laut</title>
8   <script>
9     function showAlert(message) {
10       alert(message);
11     }
12   </script>
13 </head>
14 <body>
15   <div class="container">
16     <h2>Login Volunteer Konservasi Laut</h2>
17     <form action="/auth/login" method="POST">
18       <label for="username">Username</label>
19       <input type="text" id="username" name="username" required>
20
21       <label for="password">Password</label>
22       <input type="password" id="password" name="password" required>
23
24       <button type="submit">Login</button>
25     </form>
26     <p>Belum punya akun? <a href="/auth/register">Daftar di sini</a></p>
27   </div>
28 </body>
29 </html>
30
```

Penjelasan kodingan:

Dokumen HTML dimulai dengan deklarasi `<!DOCTYPE html>` yang menunjukkan bahwa ini adalah dokumen HTML5. Elemen `<html>` dengan atribut `lang="en"` mendefinisikan bahasa dokumen sebagai bahasa Inggris. Di dalam elemen `<head>`, terdapat beberapa elemen penting: pertama, `<meta charset="UTF-8">` yang mengatur karakter encoding dokumen ke UTF-8, memungkinkan penggunaan karakter dari berbagai bahasa. Selanjutnya, elemen `<meta name="viewport" content="width=device-width, initial-scale=1.0">` memastikan tampilan responsif pada perangkat mobile dengan menyesuaikan lebar halaman sesuai dengan lebar layar perangkat. Elemen `<link rel="stylesheet" href="/style.css">` menghubungkan dokumen dengan file CSS eksternal untuk styling, sedangkan elemen `<title>` memberikan judul pada halaman, yaitu "Login Volunteer Konservasi Laut".

Di dalam elemen `<script>`, terdapat fungsi JavaScript `showAlert(message)` yang menampilkan pesan dalam bentuk pop-up alert saat dipanggil, meskipun fungsi ini tidak digunakan dalam kode yang disajikan.

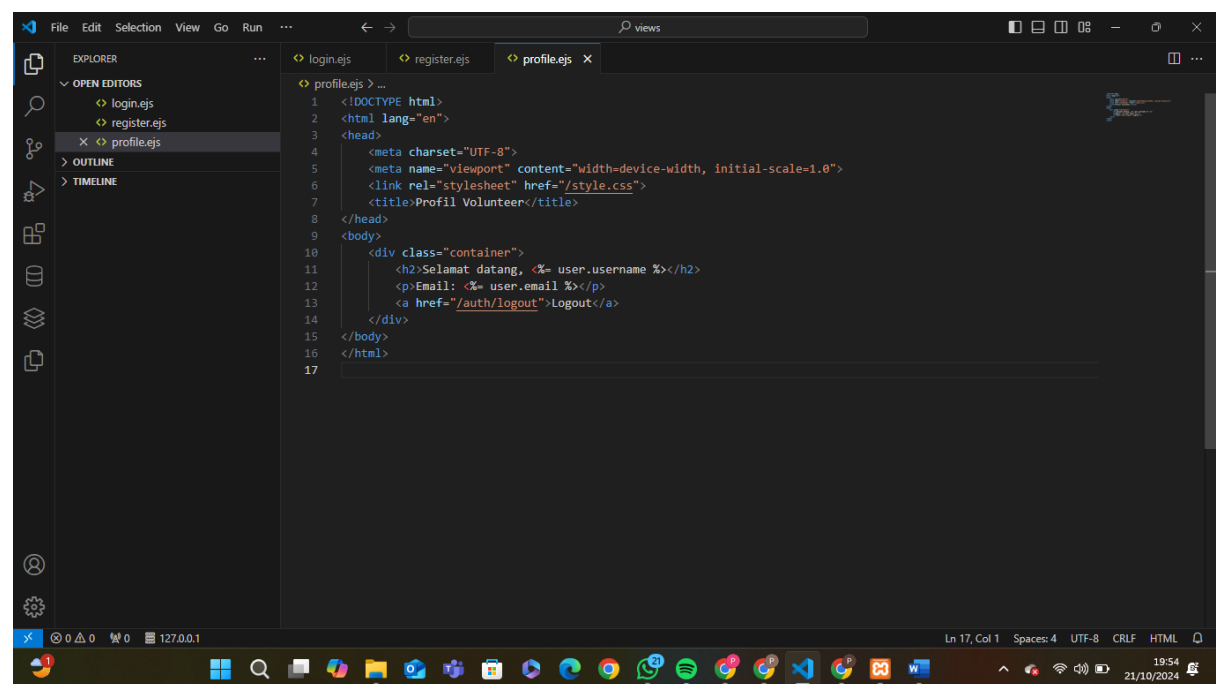
Bagian `<body>` adalah tempat konten yang terlihat oleh pengguna. Terdapat sebuah `<div>` dengan kelas `container`, yang berfungsi sebagai wadah untuk mengelompokkan elemen-elemen di dalamnya. Di dalam div ini, terdapat elemen `<h2>` yang menampilkan judul "Login Volunteer Konservasi Laut", memberi tahu pengguna tentang tujuan halaman.

Selanjutnya, terdapat elemen ``<form>`` yang mengatur aksi pengiriman data ke URL ``/auth/login`` menggunakan metode ``POST``, yang aman untuk mengirimkan data sensitif.

Di dalam formulir, terdapat dua label dan input. Label pertama untuk username dihubungkan dengan input bertipe teks melalui atribut ``for`` dan ``id``, di mana pengguna diminta untuk memasukkan username mereka. Label kedua terkait dengan input bertipe password yang juga menggunakan atribut ``for`` dan ``id`` untuk memberikan instruksi memasukkan password. Kedua input tersebut memiliki atribut ``required``, yang mengharuskan pengguna mengisi kolom tersebut sebelum mengirimkan formulir. Di bawah input, terdapat tombol ``<button>`` dengan tipe ``submit``, yang ketika diklik, akan mengirimkan data formulir ke server.

Terakhir, ada paragraf yang memberikan informasi tambahan kepada pengguna bahwa jika mereka belum memiliki akun, mereka bisa mendaftar dengan mengikuti tautan yang mengarah ke halaman pendaftaran di ``/auth/register``. Tautan ini diimplementasikan menggunakan elemen ``<a>`` yang membuatnya interaktif dan mudah diakses. Secara keseluruhan, dokumen HTML ini dirancang untuk memberikan antarmuka pengguna yang sederhana dan jelas bagi pengguna yang ingin login sebagai sukarelawan dalam program konservasi laut.

- KODINGAN `profile.ejs`



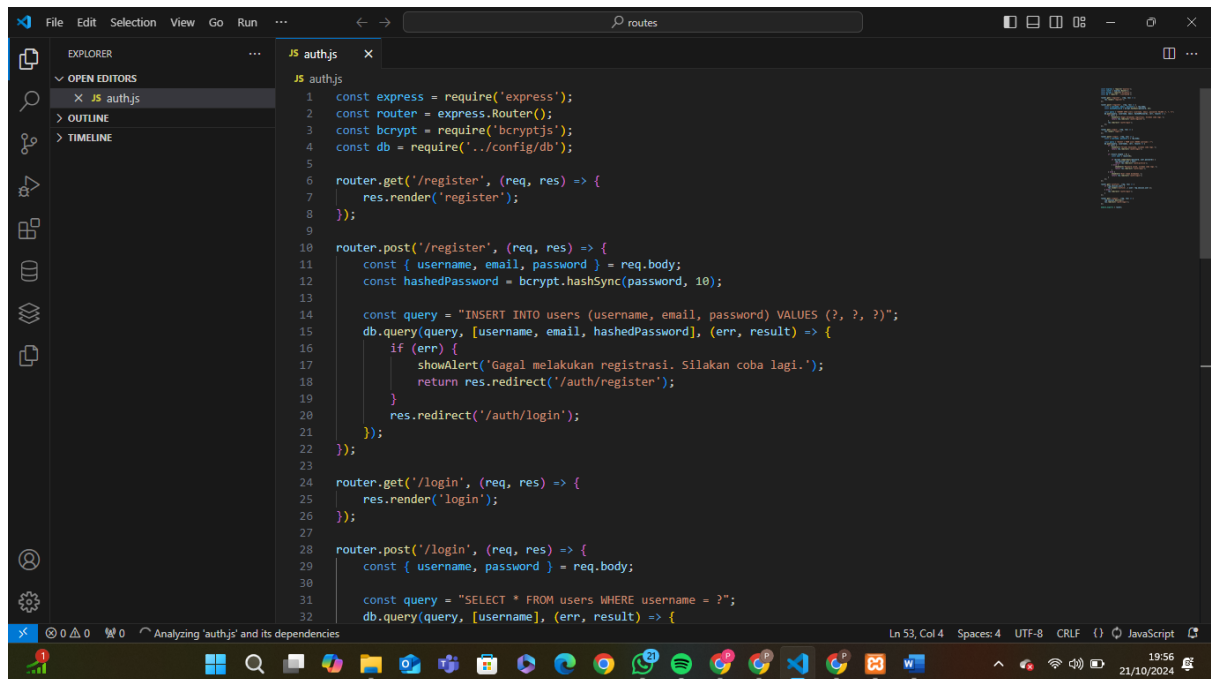
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <link rel="stylesheet" href="/style.css">
7   <title>Profil Volunteer</title>
8 </head>
9 <body>
10   <div class="container">
11     <h2>Selamat datang, <%= user.username %></h2>
12     <p>Email: <%= user.email %></p>
13     <a href="/auth/logout">Logout</a>
14   </div>
15 </body>
16 </html>
17
```

Penjelasan kodingan:

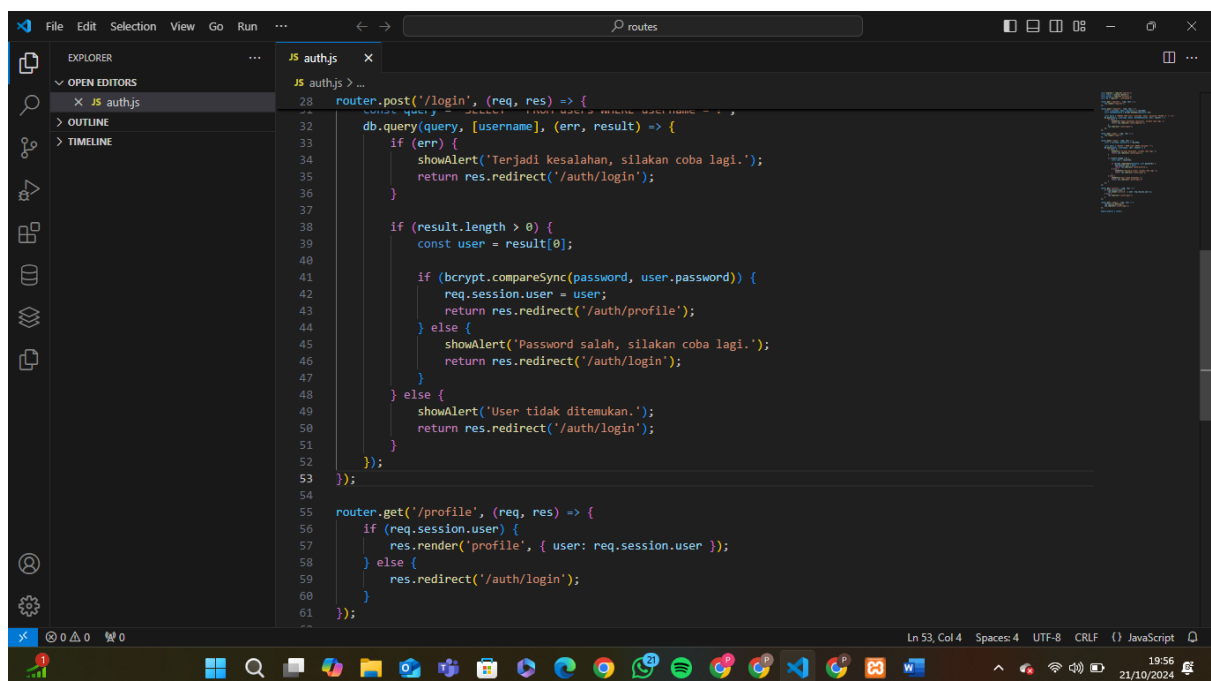
- Deklarasi Dokumen: `<!DOCTYPE html>` menunjukkan bahwa dokumen ini adalah dokumen HTML5.
- Elemen `<html>`:
lang="en" menetapkan bahwa bahasa yang digunakan dalam dokumen ini adalah bahasa Inggris.

- Bagian `<head>`:
 - `<meta charset="UTF-8">`: Mengatur karakter encoding dokumen ke UTF-8, yang mendukung berbagai karakter.
 - `<meta name="viewport" content="width=device-width, initial-scale=1.0">`: Menjamin bahwa halaman akan responsif, menyesuaikan lebar dengan layar perangkat.
 - `<link rel="stylesheet" href="/style.css">`: Menghubungkan dokumen dengan file CSS eksternal yang digunakan untuk styling halaman.
 - `<title>Profil Volunteer</title>`: Mengatur judul halaman yang ditampilkan di tab browser.
- `<script>`:
 - Mendeklarasikan fungsi JavaScript `showAlert(message)`, yang menampilkan alert dengan pesan yang diberikan sebagai parameter. Fungsi ini dapat dipanggil di bagian lain dalam kode untuk menunjukkan informasi kepada pengguna.
- Bagian `<body>`:
 - `<div class="container">`: Membungkus konten di dalam elemen `div` dengan kelas `container`, yang umumnya digunakan untuk styling dan tata letak.
 - `<h2>Selamat datang, <%= user.username %></h2>`: Menampilkan pesan sambutan kepada pengguna dengan nama pengguna mereka. Sintaks `<%= user.username %>` adalah sintaks EJS yang menyisipkan nilai `username` dari objek `user` yang terdaftar di sesi.
 - `<p>Email: <%= user.email %></p>`: Menampilkan alamat email pengguna dengan cara yang sama menggunakan sintaks EJS untuk menyisipkan nilai email dari objek `user`.
 - `Logout`: Menyediakan tautan bagi pengguna untuk keluar dari akun mereka. Tautan ini mengarah ke URL `/auth/logout`, yang akan memproses permintaan logout ketika diklik.

- Dalam folder routes
- 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 router.get('/register', (req, res) => {
7   res.render('register');
8 });
9
10 router.post('/register', (req, res) => {
11   const { username, email, password } = req.body;
12   const hashedPassword = bcrypt.hashSync(password, 10);
13
14   const query = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";
15   db.query(query, [username, email, hashedPassword], (err, result) => {
16     if (err) {
17       showAlert('Gagal melakukan registrasi. Silakan coba lagi.');
```



```
33   db.query(query, [username], (err, result) => {
34     if (err) {
35       showAlert('Terjadi kesalahan, silakan coba lagi.');
```

Penjelasan kodingan:

Kode tersebut adalah bagian dari routing untuk aplikasi web yang menggunakan Express, berfokus pada autentikasi pengguna, termasuk registrasi, login, dan pengelolaan sesi pengguna. Pertama, modul `express` diimpor dan router dibuat menggunakan `express.Router()`. Selain itu, `bcryptjs` digunakan untuk hashing password, dan koneksi ke database diimpor dari file konfigurasi `db.js`.

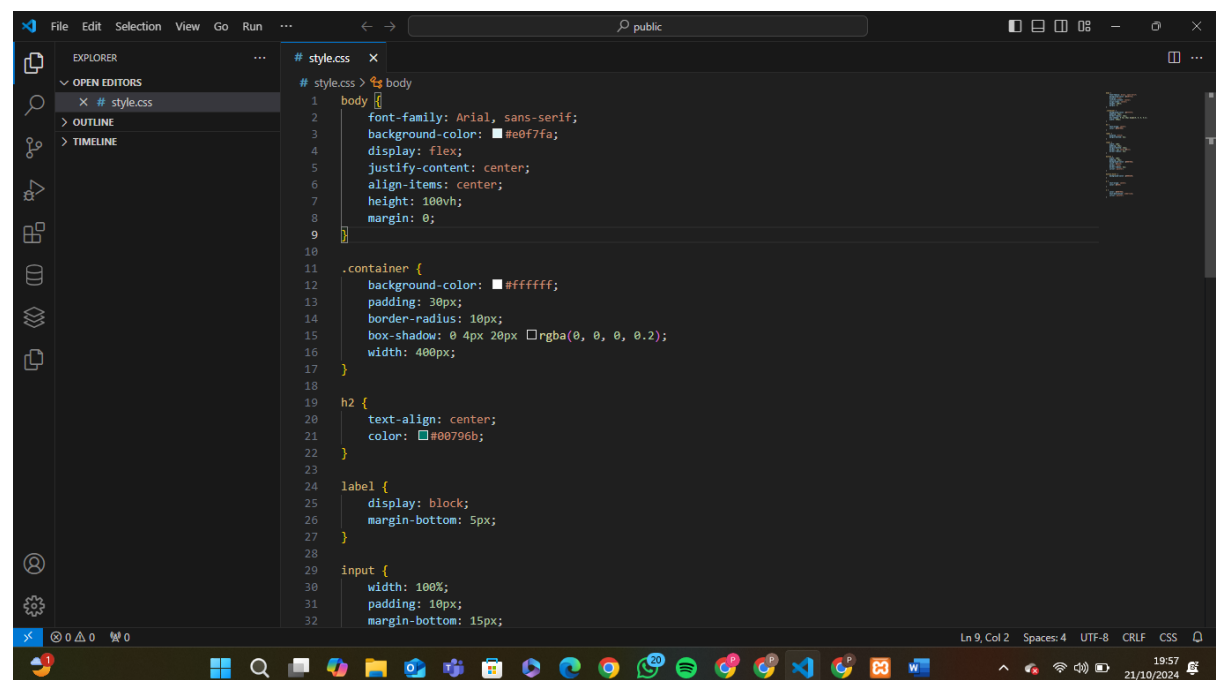
Fungsi pertama adalah route untuk menampilkan halaman registrasi. Ketika pengguna mengakses `/auth/register`, aplikasi akan merender halaman registrasi yang sesuai. Untuk proses registrasi, route POST di `/auth/register` menerima data dari form. Data seperti username, email, dan password diambil dari `req.body`. Password kemudian di-hash menggunakan `bcrypt.hashSync()` dengan tingkat kompleksitas 10 sebelum disimpan ke dalam database. Jika penyimpanan berhasil, pengguna akan dialihkan ke halaman login; jika terjadi kesalahan, pesan kesalahan akan ditampilkan dan pengguna akan tetap berada di halaman registrasi.

Fungsi selanjutnya menangani login pengguna. Ketika pengguna mengakses `/auth/login`, aplikasi akan merender halaman login. Ketika data login dikirim melalui form, route POST di `/auth/login` akan mencari pengguna berdasarkan username yang dimasukkan. Jika pengguna ditemukan, password yang dimasukkan akan dibandingkan dengan password yang di-hash yang tersimpan di database menggunakan `bcrypt.compareSync()`. Jika password cocok, objek pengguna disimpan dalam sesi dan pengguna dialihkan ke halaman profil; jika tidak cocok atau pengguna tidak ditemukan, pesan kesalahan akan ditampilkan, dan pengguna akan tetap di halaman login.

Route untuk halaman profil akan merender halaman profil pengguna jika sesi pengguna ada. Jika tidak, pengguna akan dialihkan kembali ke halaman login. Terakhir, route untuk logout menghapus sesi pengguna dengan `req.session.destroy()`, dan pengguna akan diarahkan kembali ke halaman login.

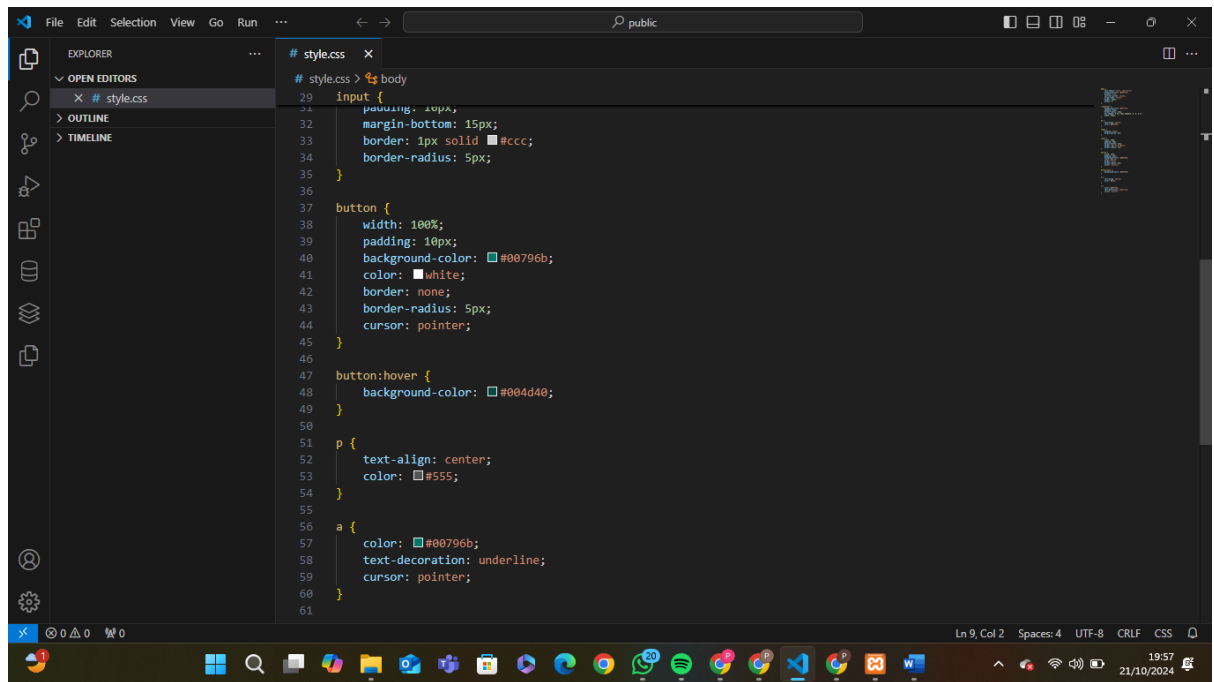
➤ KODINGAN style.css

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

A screenshot of a code editor window with a dark theme. The editor is open to a file named 'style.css'. The left sidebar shows the 'EXPLORER' view with 'style.css' selected. The main editor area contains the following CSS code:

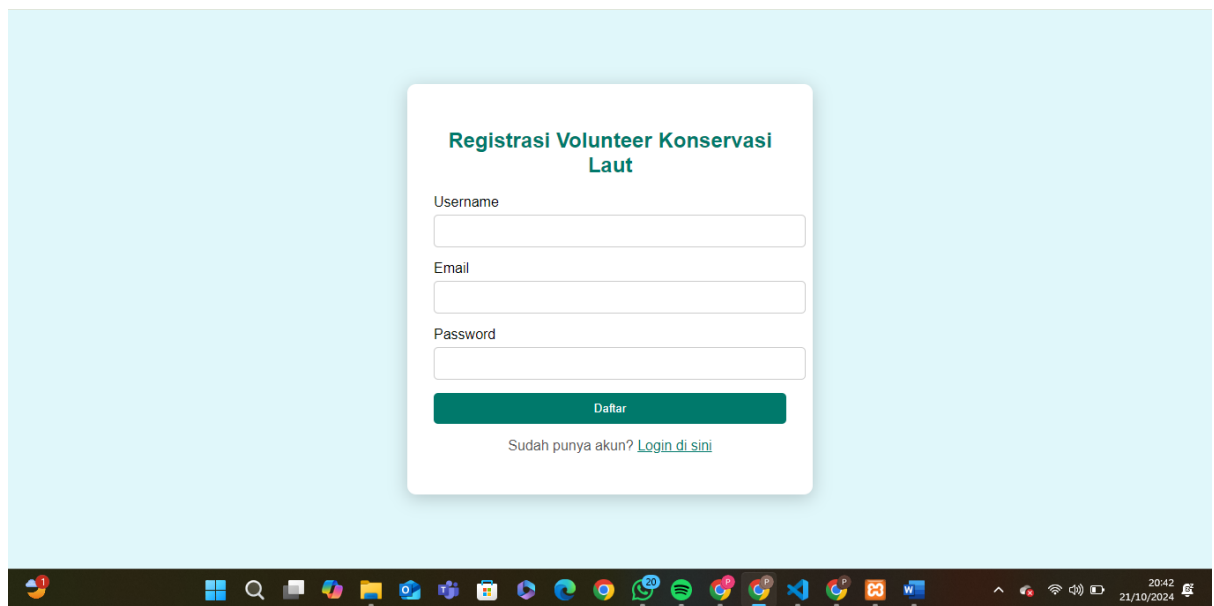
```
# style.css
1 body {
2   font-family: Arial, sans-serif;
3   background-color: #e0f7fa;
4   display: flex;
5   justify-content: center;
6   align-items: center;
7   height: 100vh;
8   margin: 0;
9 }
10
11 .container {
12   background-color: #ffffff;
13   padding: 30px;
14   border-radius: 10px;
15   box-shadow: 0 4px 20px rgba(0, 0, 0, 0.2);
16   width: 400px;
17 }
18
19 h2 {
20   text-align: center;
21   color: #00796b;
22 }
23
24 label {
25   display: block;
26   margin-bottom: 5px;
27 }
28
29 input {
30   width: 100%;
31   padding: 10px;
32   margin-bottom: 15px;
```

The status bar at the bottom indicates 'Ln 9, Col 2', 'Spaces: 4', 'UTF-8', 'CRLF', and 'CSS'. The system tray at the bottom shows the date and time as '21/10/2024' and '19:57'.



```
# style.css
body
29 input {
30   padding: 10px;
31   margin-bottom: 15px;
32   border: 1px solid #ccc;
33   border-radius: 5px;
34 }
35
36
37 button {
38   width: 100%;
39   padding: 10px;
40   background-color: #00796b;
41   color: white;
42   border: none;
43   border-radius: 5px;
44   cursor: pointer;
45 }
46
47 button:hover {
48   background-color: #004d40;
49 }
50
51 p {
52   text-align: center;
53   color: #555;
54 }
55
56 a {
57   color: #00796b;
58   text-decoration: underline;
59   cursor: pointer;
60 }
61
```

➤ **TAMPILAN HALAMAN:**



Setelah dijalankan di server localhost:3000, aplikasi web ini akan menampilkan halaman login untuk pengguna yang ingin bergabung sebagai volunteer konservasi laut. Jika pengguna mencoba untuk login tanpa memiliki akun, sistem akan menampilkan pesan "User tidak ditemukan" melalui pop-up alert, yang mengindikasikan bahwa mereka harus melakukan registrasi terlebih dahulu. Setelah pengguna berhasil melakukan registrasi, mereka akan diminta untuk login kembali. Jika login berhasil, pengguna akan diarahkan ke halaman profil, di mana mereka dapat melihat informasi akun mereka, seperti username dan email.

Di halaman profil, pengguna akan melihat opsi untuk logout. Ketika mereka mengklik tombol logout, sesi mereka akan dihapus, dan mereka akan diarahkan kembali ke halaman

login. Jika pengguna melakukan kesalahan saat login, seperti memasukkan password yang salah, pop-up alert juga akan muncul untuk memberi tahu mereka tentang kesalahan tersebut. Dengan fitur pop-up alert ini, pengguna akan mendapatkan informasi yang jelas dan langsung mengenai status autentikasi mereka, memudahkan navigasi dan penggunaan aplikasi.

IV. KESIMPULAN

Web untuk registrasi dan autentikasi pengguna dalam konteks volunteer konservasi laut berhasil dibangun menggunakan Node.js dengan framework Express, mencakup beberapa fitur penting, seperti halaman registrasi, login, dan profil pengguna, yang dihubungkan dengan database untuk menyimpan informasi pengguna. Proses registrasi dan login dirancang untuk memastikan keamanan data pengguna dengan menggunakan hashing password melalui bcrypt. Sistem juga memberikan umpan balik yang jelas kepada pengguna melalui pop-up alert, yang muncul saat terjadi kesalahan, seperti ketika pengguna mencoba login tanpa akun atau salah memasukkan password.

Setelah berhasil login, pengguna akan diarahkan ke halaman profil yang menampilkan informasi akun mereka, serta menyediakan opsi untuk logout. Ketika logout dilakukan, sesi pengguna akan dihapus, dan mereka akan kembali ke halaman login. Dengan demikian, web tersebut tidak hanya berfungsi sebagai platform pendaftaran volunteer, tetapi juga memberikan pengalaman pengguna yang interaktif dan informatif melalui fitur pop-up alert dan navigasi yang jelas.