Laporan Praktikum
Mata Kuliah Pemograman Web Berorientasi Objek

Pertemuan 5 Tugas ke 4

“SESSION”

Dosen Pengampu :

Willdan Aprizal Arifin, S.Pd., M.Kom

Disusun Oleh :

Muhammad Zikri Alhaq (2308504)

PROGRAM STUDI SISTEM INFORMASI KELAUTAN

UNIVERSITAS PENDIDIKAN INDONESIA

2024

Pendahuluan

Praktikum ini bertujuan untuk membangun aplikasi web sederhana menggunakan Node.js dan Express yang memungkinkan pengguna untuk melakukan pendaftaran, login, dan mengelola profil mereka. Aplikasi ini mengintegrasikan database MySQL untuk menyimpan data pengguna dan menggunakan EJS sebagai template engine untuk menghasilkan halaman HTML secara dinamis.

Aplikasi ini berfokus pada pengelolaan pengguna yang melibatkan autentikasi dan pengelolaan sesi pengguna. Dengan aplikasi ini, pengguna dapat mendaftar untuk akun baru, masuk ke akun mereka, dan melihat informasi profil mereka.

Tujuan

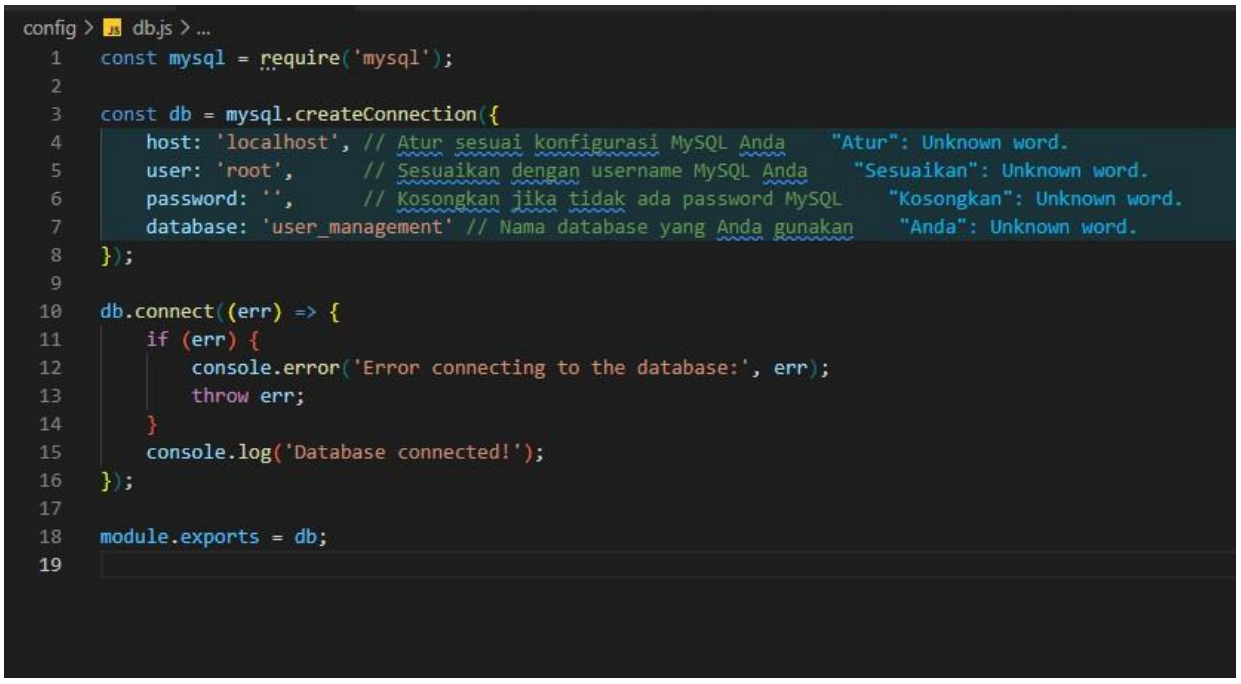
1. Membangun aplikasi web yang dapat melakukan pendaftaran dan login pengguna.
2. Menyimpan data pengguna di database MySQL.
3. Mengelola sesi pengguna menggunakan express-session untuk mengingat status login.
4. Menerapkan validasi input untuk memastikan data yang dimasukkan pengguna adalah valid.

Alat dan Bahan

1. **Node.js**: Runtime JavaScript untuk menjalankan aplikasi server.
2. **Express.js**: Framework web untuk Node.js yang memudahkan routing dan pengelolaan middleware.
3. **MySQL**: Database untuk menyimpan data pengguna, memungkinkan operasi CRUD (Create, Read, Update, Delete).
4. **EJS**: Template engine untuk menghasilkan halaman HTML secara dinamis berdasarkan data yang diberikan.
5. **Bcrypt.js**: Library untuk hashing password guna meningkatkan keamanan penyimpanan password.
6. **HTML/CSS**: Untuk membuat tampilan antarmuka pengguna yang responsif dan menarik.

Penjelasan Program

db.js (Folder config)



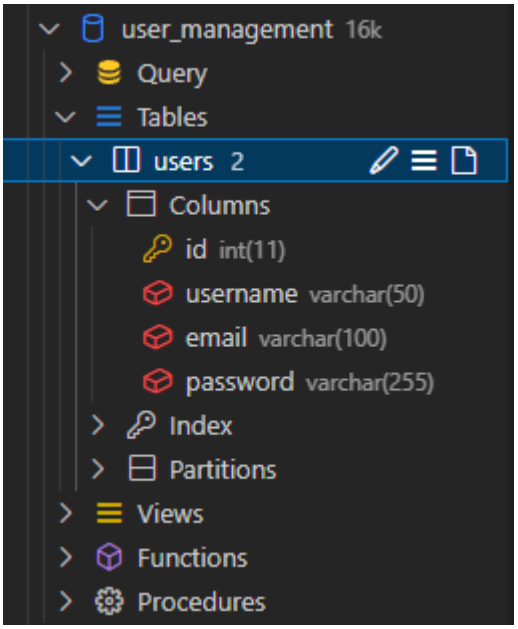

```
const mysql = require('mysql');

const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'user_management'
});

db.connect((err) => {
    if (err) {
        console.error('Error connecting to the database:', err);
        throw err;
    }
    console.log('Database connected!');
});

module.exports = db;
```

Penjelasan:

1. **require('mysql')**: Mengimpor modul MySQL yang memungkinkan koneksi ke database.
2. **mysql.createConnection()**: Membuat koneksi ke database MySQL. Parameter yang digunakan meliputi:

    A. host: Lokasi database, biasanya localhost untuk pengembangan lokal.
    B. user: Username untuk mengakses MySQL, di sini adalah root.
    C. password: Password untuk pengguna MySQL, di sini kosong karena tidak ada password yang ditentukan.
    D. database: Nama database yang digunakan, dalam hal ini adalah user_management.

3. **db.connect()**: Menghubungkan aplikasi dengan database. Jika terjadi kesalahan, akan ditampilkan pesan kesalahan, dan aplikasi akan berhenti. Jika berhasil, akan muncul pesan bahwa koneksi berhasil.



4. **module.exports = db**: Mengekspor koneksi database agar bisa digunakan di file lain.

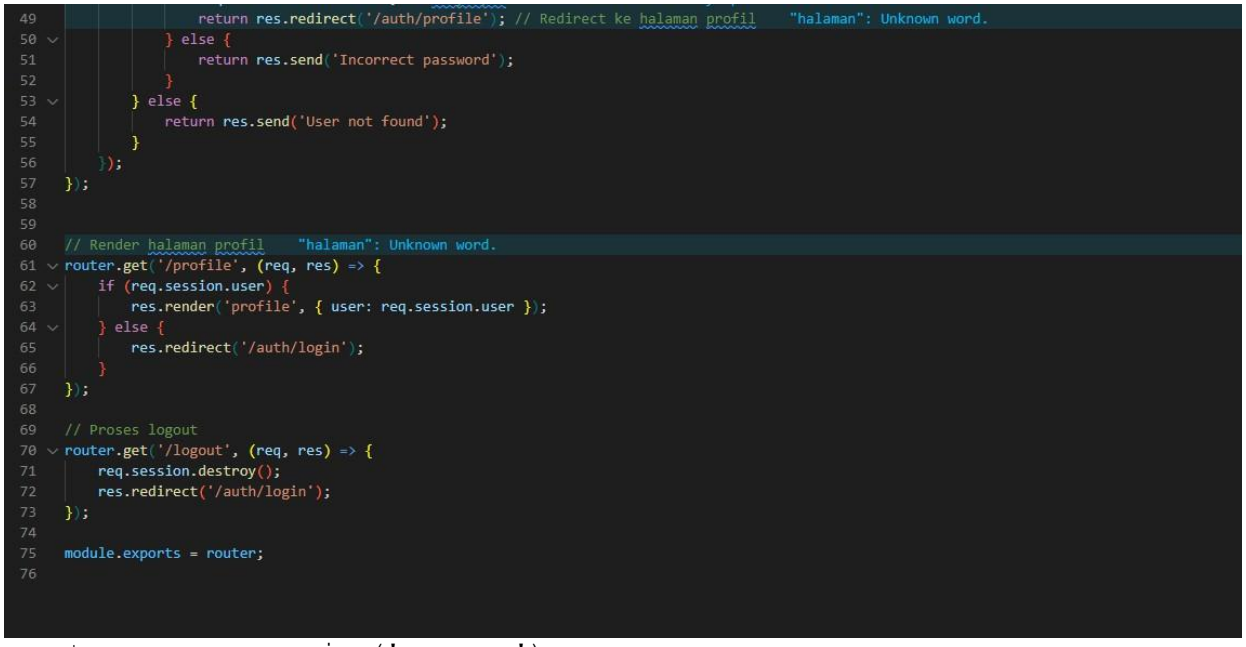
Penjelasan:

1. **Reset CSS**: Menghilangkan margin dan padding default pada semua elemen untuk memastikan tampilan konsisten di berbagai browser.
2. **Styling Body**: Mengatur font, warna latar belakang, dan menggunakan flexbox untuk memusatkan konten secara vertikal dan horizontal di tengah halaman.
3. **Container**: Menciptakan area untuk form dengan latar belakang putih, padding, border-radius, dan bayangan untuk efek visual yang menarik.
4. **Input Fields**: Mengatur lebar, padding, dan transisi border untuk meningkatkan pengalaman pengguna saat memasukkan data.
5. **Buttons**: Mengatur gaya untuk tombol, termasuk warna latar belakang, warna teks, dan efek hover untuk memberikan umpan balik kepada pengguna.
6. **Media Queries**: Menyesuaikan ukuran elemen untuk tampilan yang lebih baik di perangkat dengan layar kecil.

auth.js (Folder routes)

routes > auth.js > ...

```
const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
const db = require('../config/db');

// Render halaman register    "halaman": Unknown word.
router.get('/register', (req, res) => {
    res.render('register');
});

// Proses register user
router.post('/register', (req, res) => {
    const { username, email, password } = req.body;

    // Validasi server-side untuk memastikan input tidak kosong    "Validasi": Unknown word.
    if (!username || !email || !password) {
        return res.send('Please fill in all fields.');
    }

    // Hash password sebelum menyimpan ke database    "sebelum": Unknown word.
    const hashedPassword = bcrypt.hashSync(password, 10);

    const query = "INSERT INTO users (username, email, password) VALUES (?, ?, ?)";
    db.query(query, [username, email, hashedPassword], (err, result) => {
        if (err) throw err;
        res.redirect('/auth/login'); // Redirect ke halaman login setelah register sukses    "halaman": Unknown word.
    });
});

// Render halaman login    "halaman": Unknown word.
router.get('/login', (req, res) => {
    res.render('login');
});

// Proses login user
router.post('/login', (req, res) => {
    const { username, password } = req.body;

    const query = "SELECT * FROM users WHERE username = ?";
    db.query(query, [username], (err, result) => {
        if (err) throw err;

        if (result.length > 0) {
            const user = result[0];

            // Periksa password    "Periksa": Unknown word.
            if (bcrypt.compareSync(password, user.password)) {
                req.session.user = user; // Menyimpan user di session    "Menyimpan": Unknown word.
```



```
                return res.redirect('/auth/profile'); // Redirect ke halaman profil    "halaman": Unknown word.
            } else {
                return res.send('Incorrect password');
            }
        } else {
            return res.send('User not found');
        }
    });
});


// Render halaman profil    "halaman": Unknown word.
router.get('/profile', (req, res) => {
    if (req.session.user) {
        res.render('profile', { user: req.session.user });
    } else {
        res.redirect('/auth/login');
    }
});

// Proses logout
router.get('/logout', (req, res) => {
    req.session.destroy();
    res.redirect('/auth/login');
});

module.exports = router;
```

```
const express = require('express');
const router = express.Router();
const bcrypt = require('bcryptjs');
const db = require('../config/db');

// Route handlers for registration, login, profile, and logout
```

Penjelasan:

1. **`express.Router()`**: Membuat router baru untuk mengelompokkan rute terkait autentikasi.
2. **Bcrypt**: Mengimpor bcrypt untuk hashing password sebelum disimpan di database.
3. **Database Connection**: Mengimpor koneksi database yang telah didefinisikan di `db.js` untuk melakukan operasi CRUD pada tabel pengguna.

Route Handlers

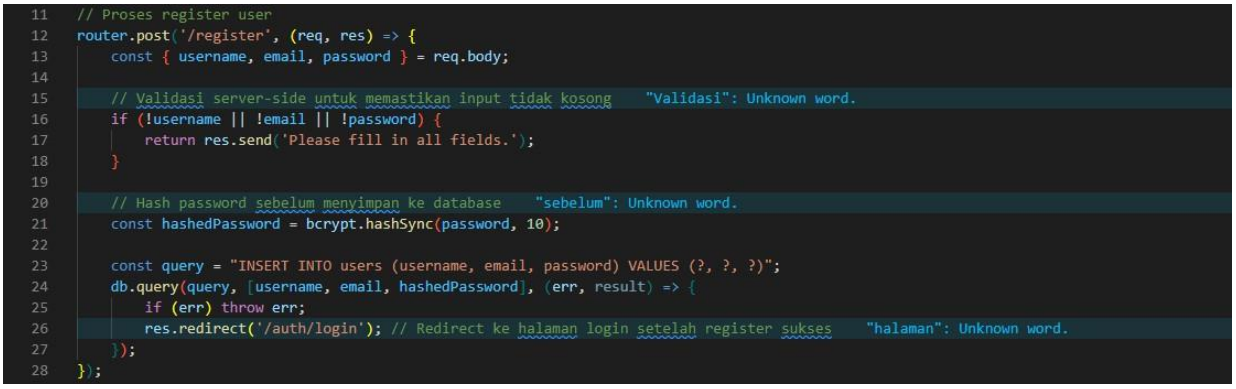
A. **Render halaman register**

```
// Render halaman register    "halaman": Unknown word.
router.get('/register', (req, res) => {
    res.render('register');
});

```

```
router.get('/register', (req, res) => {
    res.render('register');
});
```

Penjelasan: Rute ini merender halaman pendaftaran saat pengguna mengunjungi `/auth/register`.

B. **Proses register user**

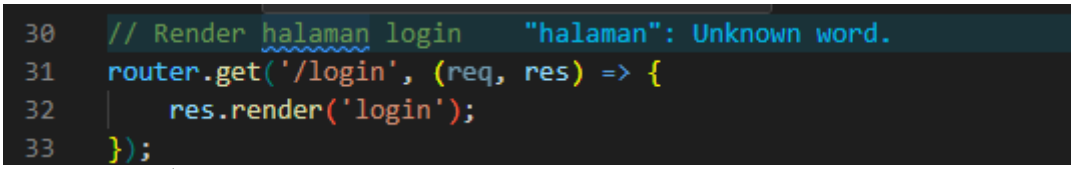


```
router.post('/register', (req, res) => {
    const { username, email, password } = req.body;

    if (!username || !email || !password) {
        return res.send('Please fill in all fields.');
    }

    const hashedPassword = bcrypt.hashSync(password, 10);
    const query = "INSERT INTO users (username, email, password) VALUES
(?, ?, ?)";
    db.query(query, [username, email, hashedPassword], (err, result) =>
{
        if (err) throw err;
        res.redirect('/auth/login');
    });
});
```

Penjelasan:

1. Mengambil data dari form pendaftaran (`username`, `email`, dan `password`).
2. Memvalidasi bahwa semua kolom diisi. Jika tidak, menampilkan pesan kesalahan.
3. Menghash password menggunakan bcrypt sebelum menyimpan ke database.
4. Menyimpan data pengguna ke tabel `users` dalam database.
5. Mengalihkan pengguna ke halaman login setelah berhasil mendaftar.

C. Render halaman login

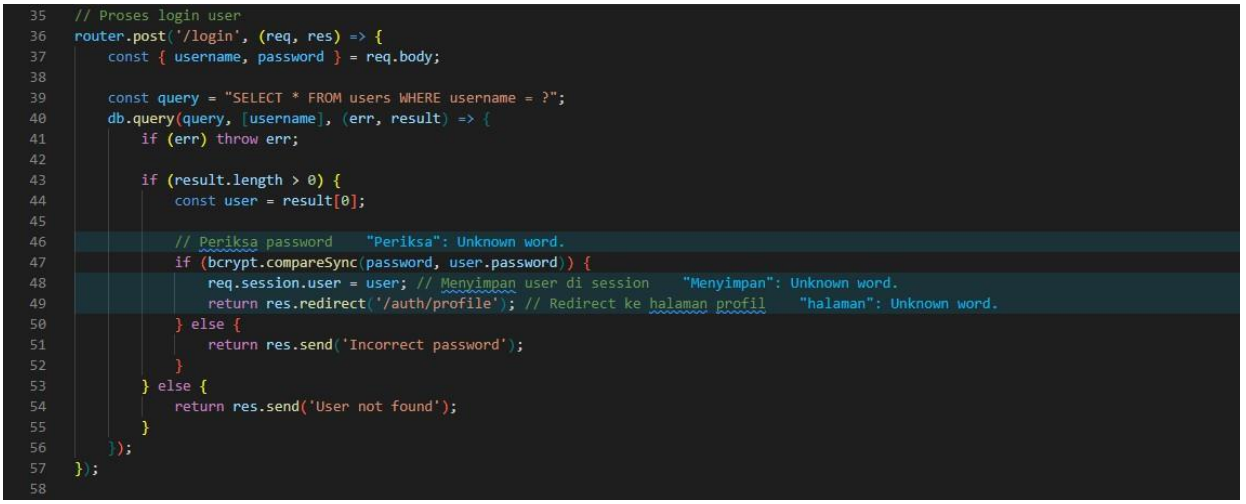


```
router.get('/login', (req, res) => {
    res.render('login');
});
```

Penjelasan:

Rute ini merender halaman login saat pengguna mengunjungi `/auth/login`.

D. Proses login user



```
// Proses login user
router.post('/login', (req, res) => {
    const { username, password } = req.body;

    const query = "SELECT * FROM users WHERE username = ?";
    db.query(query, [username], (err, result) => {
        if (err) throw err;

        if (result.length > 0) {
            const user = result[0];

            // Periksa password    "Periksa": Unknown word.
            if (bcrypt.compareSync(password, user.password)) {
                req.session.user = user; // Menyimpan user di session    "Menyimpan": Unknown word.
                return res.redirect('/auth/profile'); // Redirect ke halaman profil    "halaman": Unknown word.
            } else {
                return res.send('Incorrect password');
            }
        } else {
            return res.send('User not found');
        }
    });
});

```

```
router.post('/login', (req, res) => {
    const { username, password } = req.body;

    const query = "SELECT * FROM users WHERE username = ?";
    db.query(query, [username], (err, result) => {
        if (err) throw err;

        if (result.length > 0) {
            const user = result[0];

            if (bcrypt.compareSync(password, user.password)) {
                req.session.user = user;
                return res.redirect('/auth/profile');
            } else {
                return res.send('Incorrect password');
            }
        } else {
            return res.send('User not found');
        }
    });
});
```

Penjelasan:

- Mengambil data dari form login (`username` dan `password`).
- Melakukan query untuk mencari pengguna berdasarkan username.
- Jika pengguna ditemukan, memeriksa password yang dimasukkan dengan password yang tersimpan (yang telah di-hash).
- Jika password benar, menyimpan data pengguna ke dalam session dan mengalihkan ke halaman profil.
- Jika password salah atau pengguna tidak ditemukan, menampilkan pesan kesalahan.

E. **Render halaman profil**

```
// Render halaman profil    "halaman": Unknown word.
router.get('/profile', (req, res) => {
    if (req.session.user) {
        res.render('profile', { user: req.session.user });
    } else {
        res.redirect('/auth/login');
    }
});

```

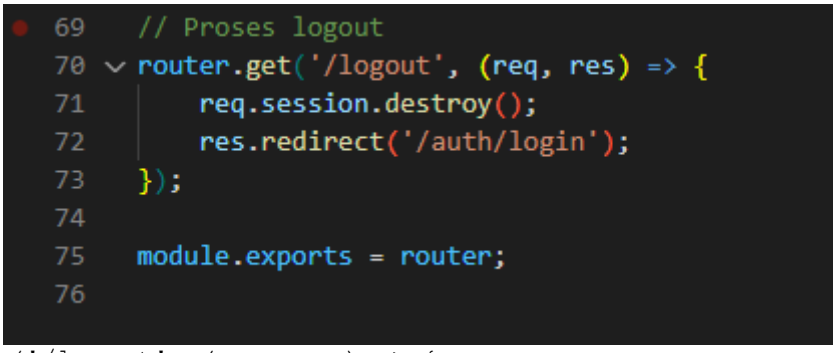
```
router.get('/profile', (req, res) => {
    if (req.session.user) {
        res.render('profile', { user: req.session.user });
```

```
    } else {
        res.redirect('/auth/login');
    }
});
```

Penjelasan:

1. Memeriksa apakah ada pengguna yang login (tersimpan dalam session).
2. Jika ada, merender halaman profil dan mengirimkan data pengguna ke template.
3. Jika tidak, mengarahkan pengguna kembali ke halaman login.

F. Logout user




```
router.get('/logout', (req, res) => {
    req.session.destroy((err) => {
        if (err) {
            return res.redirect('/auth/profile');
        }
        res.redirect('/auth/login');
    });
});
```

Penjelasan:

- Menghancurkan sesi pengguna saat logout.
- Mengalihkan pengguna kembali ke halaman login setelah berhasil logout

login.ejs (Folder Views)


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet" href="/styleslogin.css">
</head>
<body>
    <div class="container">
        <h2>Login</h2>
        <form action="/auth/login" method="POST">
            <label for="username">Username</label>
            <input type="text" id="username" name="username" required>
            <label for="password">Password</label>
            <input type="password" id="password" name="password"
required>
            <button type="submit">Login</button>
        </form>
        <p>Don't have an account? <a href="/auth/register">Register
here</a></p>
    </div>
</body>
</html>
```

Penjelasan:

1. **EJS Template**: Memungkinkan untuk menyisipkan data dinamis (meskipun dalam halaman login ini tidak ada data yang disisipkan).
2. **Form Login**: Menggunakan metode POST untuk mengirimkan username dan password ke rute `/auth/login`.
3. **Tautan Pendaftaran**: Mengarahkan pengguna ke halaman pendaftaran jika mereka belum memiliki akun.

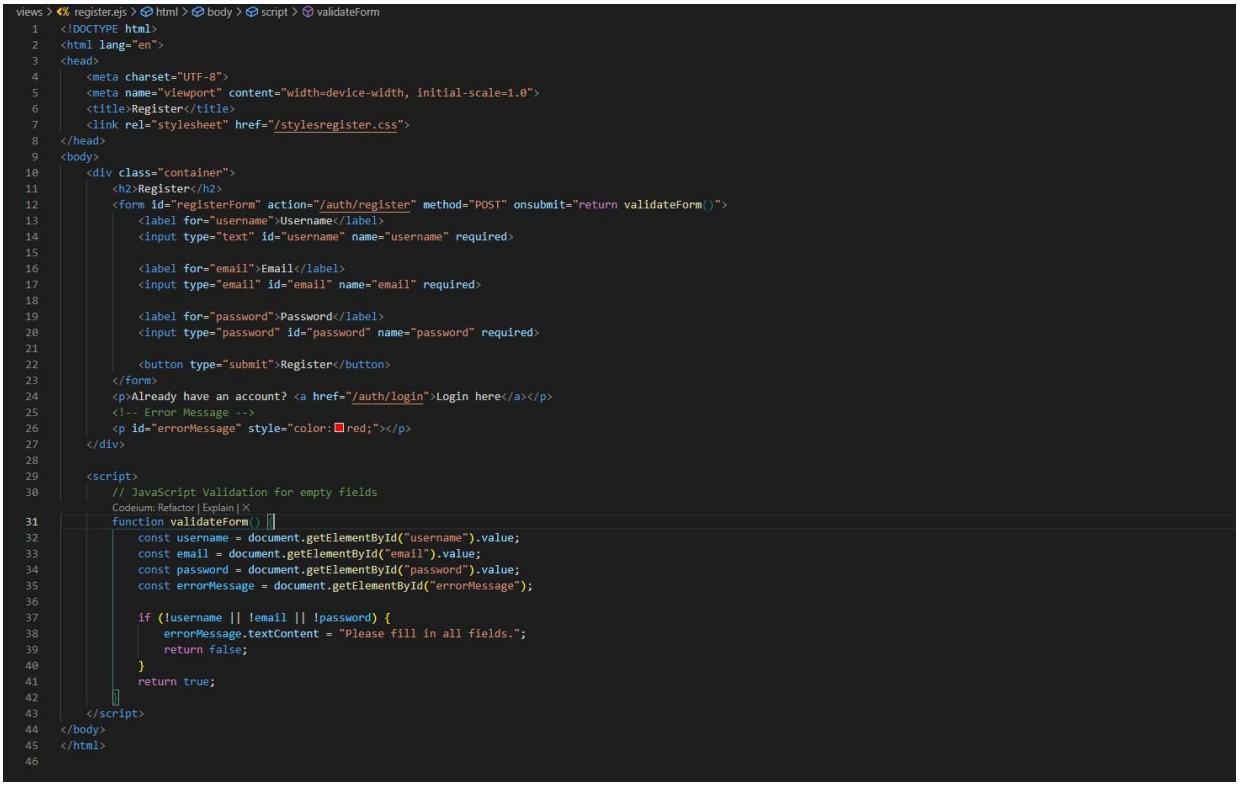
profile.ejs (Folder Views)


```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Profile</title>
    <link rel="stylesheet" href="/stylesprofile.css">
</head>
<body>
    <div class="container">
        <h2>Welcome, <%= user.username %></h2>
        <p>Email: <%= user.email %></p>
         <form action="/auth/logout" method="GET">
             <button type="submit">Logout</button>
        </form>
    </div>
</body>
</html>
```

Penjelasan:

- **Data Dinamis**: Menggunakan EJS untuk menampilkan nama pengguna yang sedang login (`<%= user.username %>`).
- **Informasi Pengguna**: Menampilkan email pengguna di halaman profil.
- **Tombol Logout**: Mengirimkan permintaan GET untuk logout, mengarahkan pengguna kembali ke halaman login.

register.ejs (Folder Views)



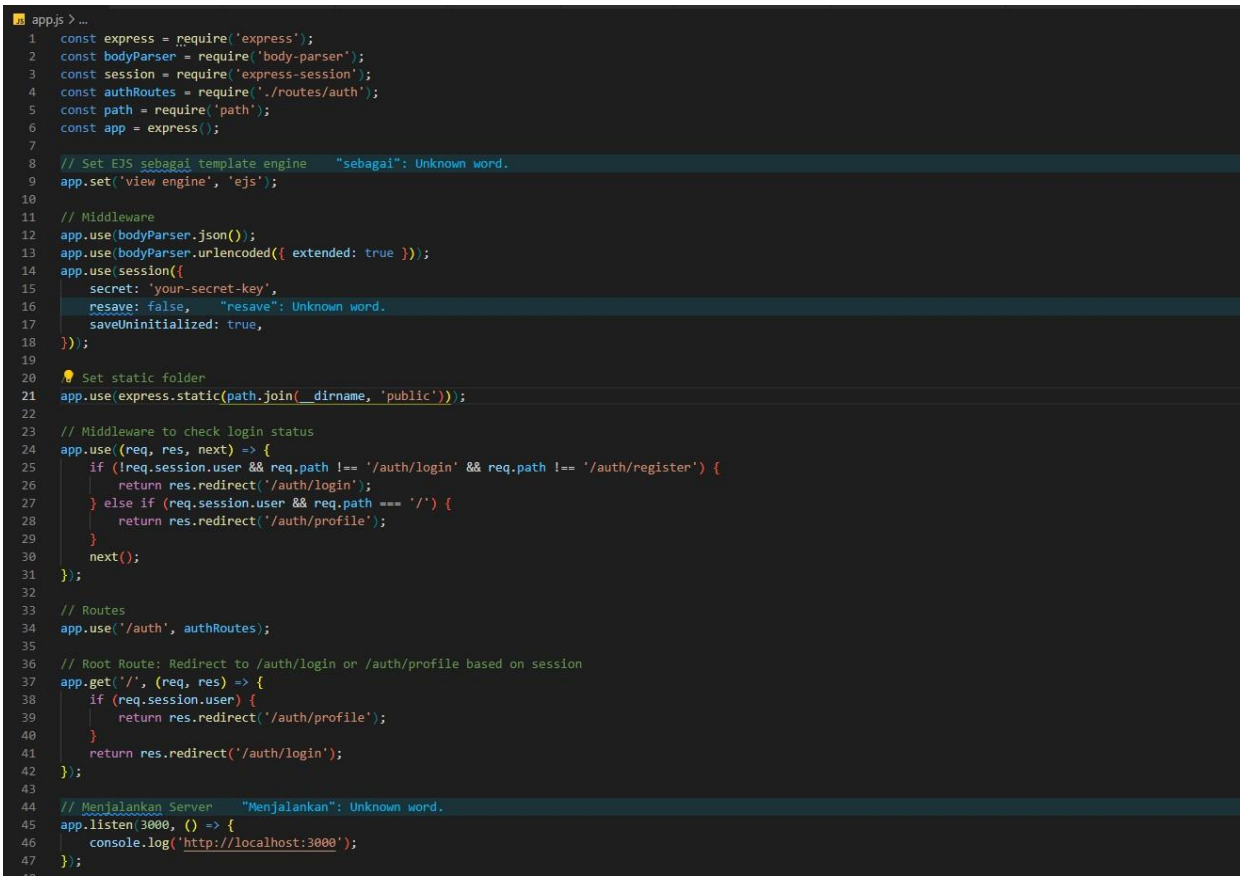
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Register</title>
    <link rel="stylesheet" href="/stylesregister.css">
</head>
<body>
    <div class="container">
        <h2>Register</h2>
        <form id="registerForm" action="/auth/register" method="POST"
onsubmit="return validateForm()">
            <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">Register</button>
        </form>
        <p>Already have an account? <a href="/auth/login">Login
here</a></p>
        <p id="errorMessage" style="color:red;"></p>
    </div>
    <script>
        function validateForm() {
            // JavaScript validation code
        }
    </script>
</body>
</html>
```

Penjelasan:

- **Form Pendaftaran**: Mengirimkan data pengguna ke rute `/auth/register`.

- **Validasi JavaScript**: Dapat digunakan untuk melakukan validasi tambahan sebelum form dikirimkan (misalnya, memastikan username tidak mengandung karakter terlarang).
- **Pesan Kesalahan**: Ditempatkan di dalam elemen `<p>` yang dapat digunakan untuk menampilkan kesalahan jika validasi tidak terpenuhi.

app.js



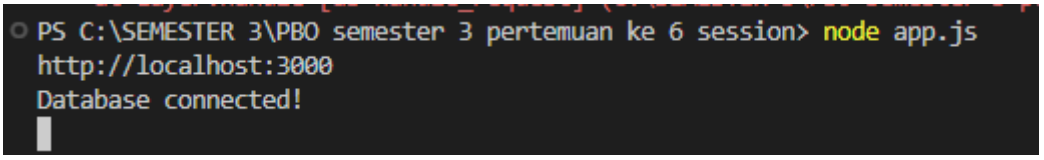

```
const express = require('express');
const bodyParser = require('body-parser');
const session = require('express-session');
const authRoutes = require('./routes/auth');
const path = require('path');
const app = express();

// Middleware setups
app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));
app.use(bodyParser.urlencoded({ extended: true }));
app.use(express.static('public'));
app.use(session({
    secret: 'secretkey',
    resave: false,
    saveUninitialized: true
}));

// Route setups
app.use('/auth', authRoutes);

// Starting the server
app.listen(3000, () => {
    console.log('Server running on http://localhost:3000');
});
```

Penjelasan:

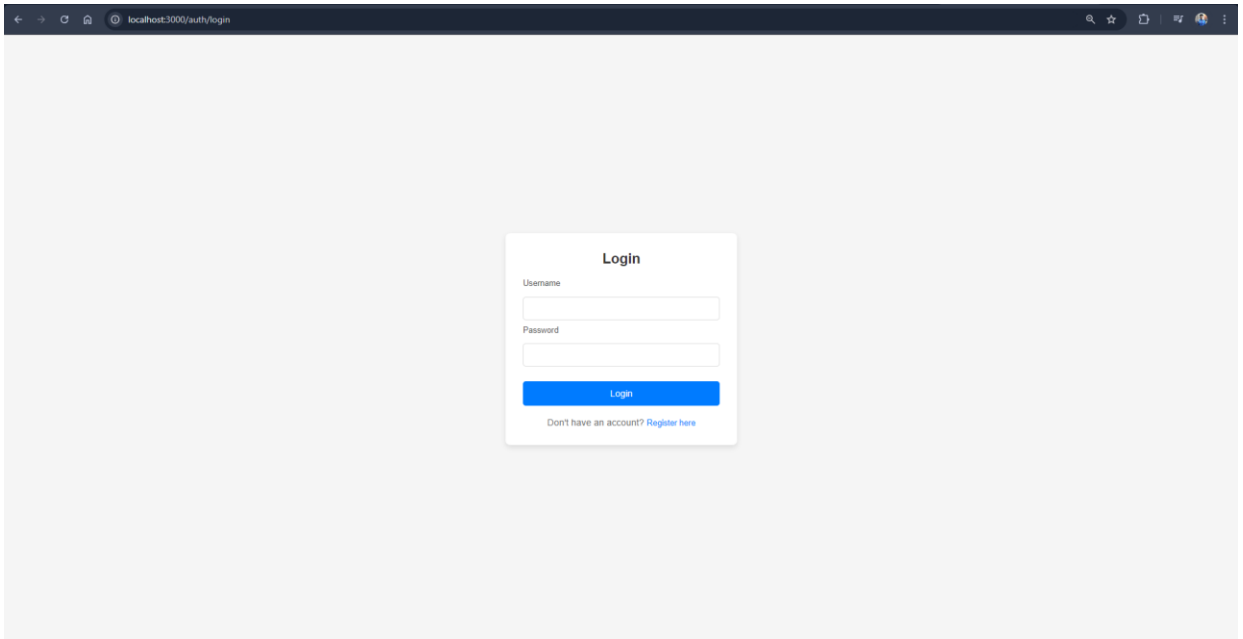
- **Express Setup**: Mengimpor dan membuat instance aplikasi Express.
- **View Engine**: Mengatur EJS sebagai template engine untuk merender tampilan.
- **Middleware**:
  - `body-parser`: Memungkinkan pengolahan data yang dikirim melalui form.
  - `express.static`: Mengatur folder `public` untuk menyajikan file statis seperti CSS.
  - `express-session`: Menyimpan data sesi untuk pengguna, mengelola status login.
- **Routing**: Menggunakan router yang telah didefinisikan untuk rute autentikasi.
- **Menjalankan Server**: Mendengarkan pada port 3000 dan mencetak pesan di konsol jika server berhasil dijalankan.

Menjalankan Aplikasi:

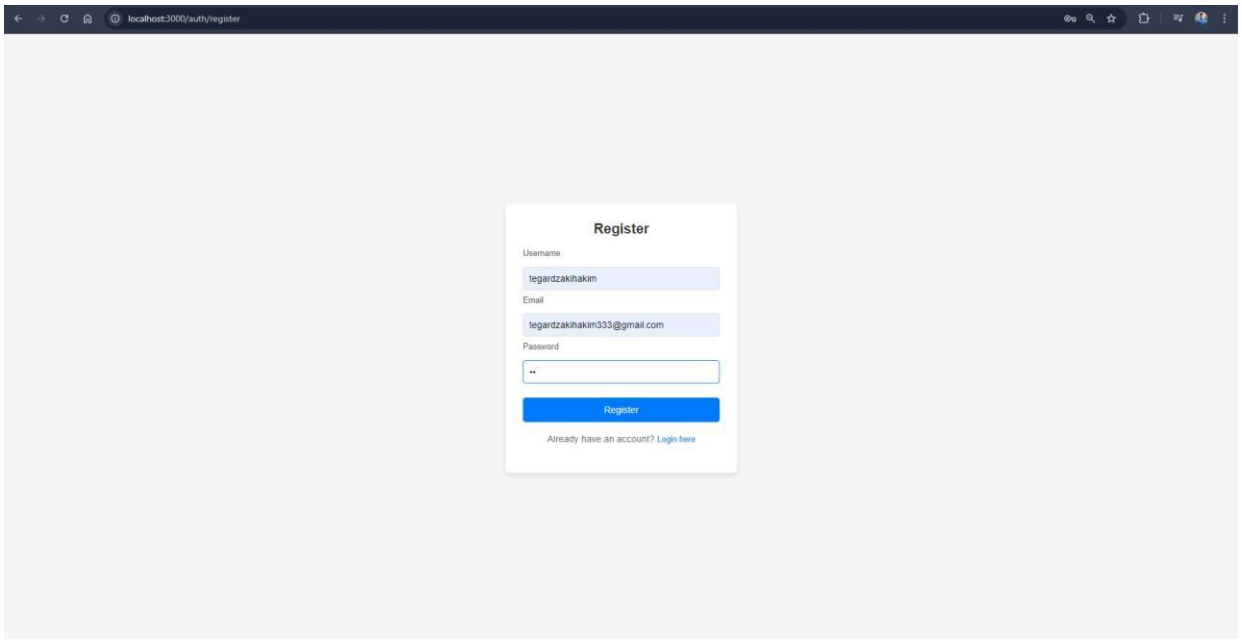
Setelah selesai dengan kode, saya menjalankan server menggunakan perintah node app.js. Kemudian,



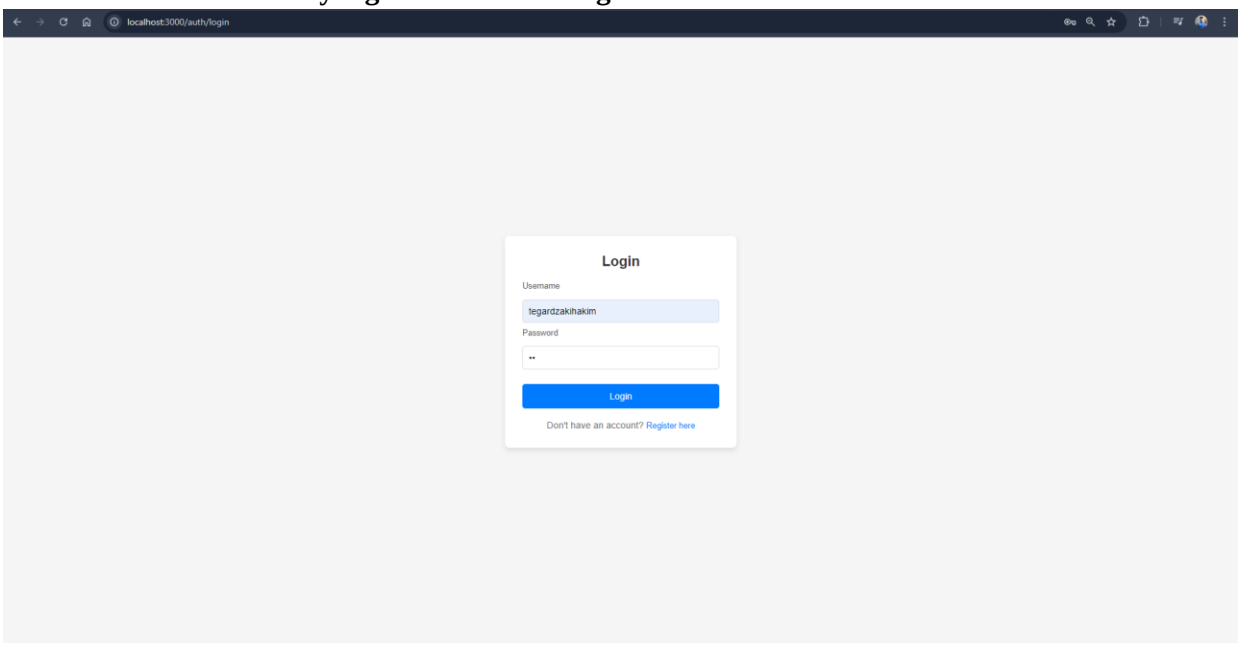
saya membuka browser dan mengakses aplikasi di http://localhost:3000.
Tampilan setelah browser dibuka:



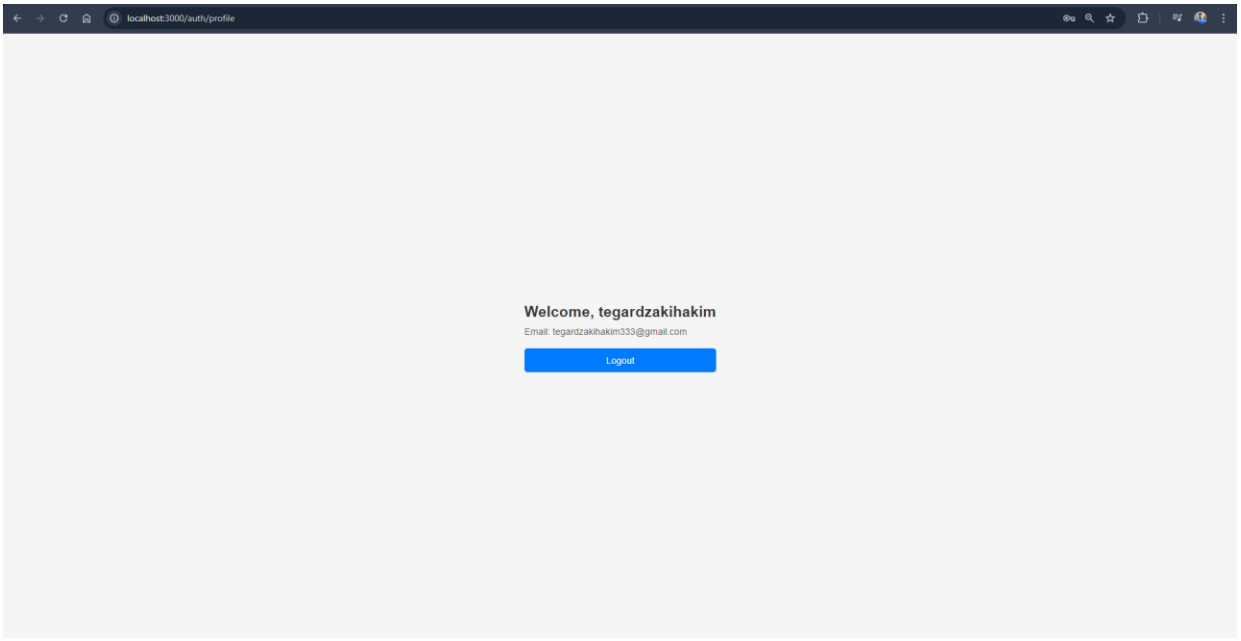
Proses Register:



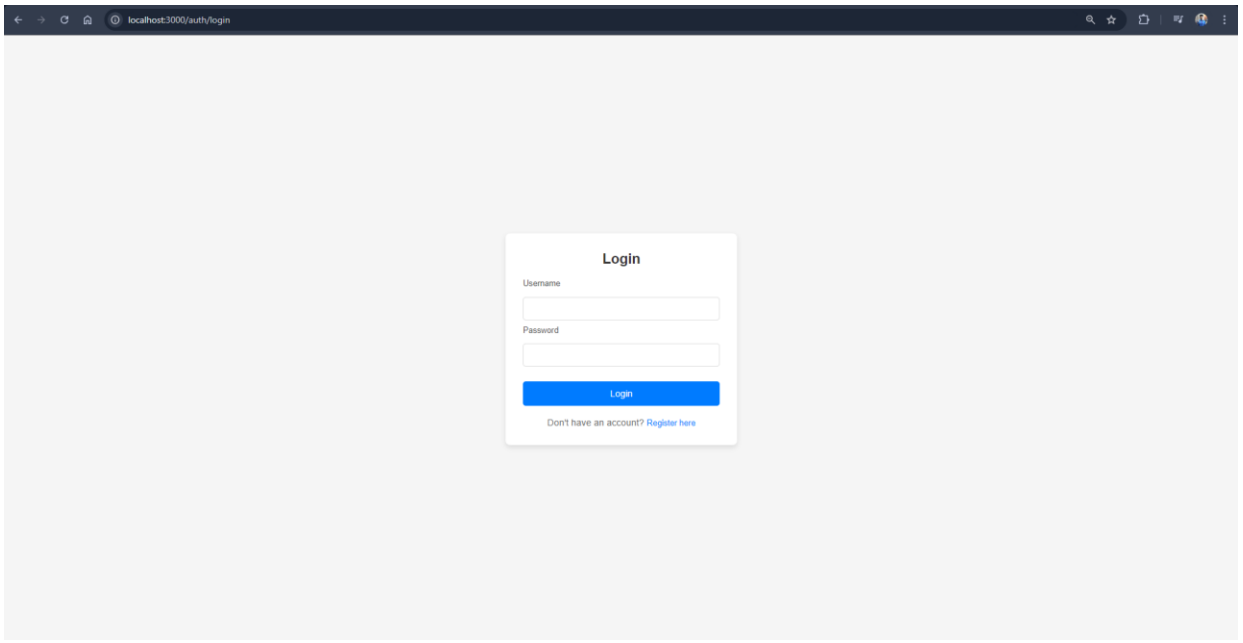
Proses memasukan akun yang telah dibuat di login



Hasil Ketika berhasil memasukan akun



Gambar saat sudah mengklik tombol logout



Kesimpulan

Praktikum ini berhasil menciptakan aplikasi web yang sederhana namun fungsional untuk manajemen pengguna. Aplikasi ini memungkinkan pengguna untuk mendaftar, login, dan melihat profil mereka. Dengan mengintegrasikan Node.js, Express, MySQL, dan EJS, aplikasi ini menunjukkan dasar-dasar pengembangan aplikasi web berbasis server dengan menggunakan teknologi JavaScript. Keberhasilan proyek ini memberikan pemahaman yang lebih dalam tentang pengelolaan sesi, validasi input, dan interaksi dengan database.