

Laporan Praktikum
Mata Kuliah Pemograman Berorientasi Object



Tugas 6

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

Disusun Oleh :
(Nur'aini Dwi Anra)
(2301148)

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

1. Pendahuluan

Pada tugas praktikum ini, dibuat sebuah aplikasi web sederhana untuk manajemen pengguna dengan fungsi register, login, dan profil. Aplikasi ini menggunakan Node.js dengan Express sebagai framework utamanya, serta MySQL sebagai database untuk menyimpan data pengguna. Untuk keamanan, password pengguna dienkripsi menggunakan bcrypt sebelum disimpan dalam database. Aplikasi ini juga memanfaatkan session untuk mengelola status login pengguna, memastikan akses ke halaman tertentu hanya dapat dilakukan oleh pengguna yang sudah login. Dengan demikian, aplikasi ini bertujuan untuk mengimplementasikan sistem autentikasi yang aman dan sederhana.

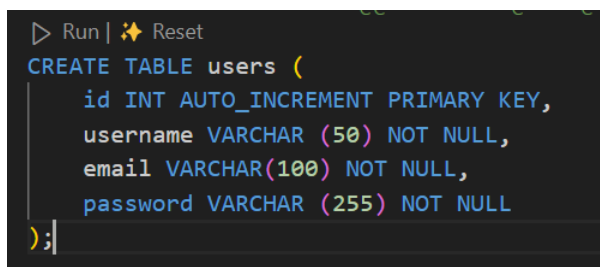
2. Struktur Program

Aplikasi ini dibangun dengan menggunakan beberapa teknologi utama:

- Node.js: Sebagai platform untuk menjalankan server aplikasi.
- Express.js: Framework yang digunakan untuk routing, middleware, dan pengelolaan request HTTP.
- MySQL: Database yang digunakan untuk menyimpan dan mengambil data pengguna.
- EJS (Embedded JavaScript): Template engine untuk merender halaman HTML secara dinamis.
- bcrypt.js: Digunakan untuk mengenkripsi kata sandi sebelum disimpan ke database.
- express-session: Untuk mengelola sesi pengguna agar tetap login selama sesi aktif.
- body-parser: Middleware untuk mengelola data yang dikirim melalui form (URL-encoded dan JSON).
- CSS: Untuk mempercantik tampilan halaman login, register, dan profil.

3. Penjelasan Code

3.1 Buat Database dan Table user

A screenshot of a code editor with a dark background. At the top, there are buttons for 'Run' and 'Reset'. Below them, the SQL code for creating a 'users' table is displayed. The code uses blue and yellow syntax highlighting. The table has four columns: 'id' (auto-incrementing primary key), 'username' (50 characters, not null), 'email' (100 characters, not null), and 'password' (255 characters, not null).

```
> Run | ✨ Reset
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR (50) NOT NULL,
  email VARCHAR(100) NOT NULL,
  password VARCHAR (255) NOT NULL
);
```

3.2 Inisiasi Node.js Project

```
PS E:\AINI\Visual Code\PEM WEB B3\Tugas 6 PEMWEB> npm init -y
```

```
PS E:\AINI\Visual Code\PEM WEB B3\Tugas 6 PEMWEB> npm install express mysql bcryptjs body-parser express-session ejs
```

3.3 Koneksi ke Database MySQL

Pada bagian ini, saya menghubungkan aplikasi ke database user_management yang berisi tabel users. Berikut adalah potongan kode untuk membuat koneksi:

```
const mysql = require('mysql');
const db=mysql.createConnection({
  host:'localhost',
  user: 'root',
  password:'',
  database:'pbotgs6'
});

db.connect ((err)=>{
  if(err) throw err;
  console.log ('Database connected...');
});

module.exports=db;
```

3.4 Koneksi visual code ke localhost:3000

```
// Menjalankan server
app.listen(3000, () => {
  console.log('Server running on port 3000, open web via http://localhost:3000');
});
```

3.5 Server Setup/Backend (app.js)

a) Templete kerangka web

```
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();

// Set EJS sebagai template engine
app.set('view engine', 'ejs');

// Set folder statis
app.use(express.static(path.join(__dirname, 'public')));
```

b) Middleware

Middleware adalah fungsi yang berjalan di antara permintaan (request) dan respons (response). Ini membantu kita mengambil data dari form yang diisi pengguna. Middleware ini menyimpan informasi sesi pengguna.

```
// Middleware
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(session({
  secret: 'secret',
  resave: false,
  saveUninitialized: true
}));
```

c) Middleware untuk Memeriksa Status Login

Fungsi ini memeriksa apakah pengguna sudah login.

- Jika belum login dan mencoba mengakses halaman lain (selain halaman login atau register), mereka akan diarahkan ke halaman login.
- Jika sudah login dan mencoba mengakses halaman utama, mereka akan diarahkan ke halaman profil.

```
// Middleware untuk memeriksa status login
app.use((req, res, next) => {
  if (!req.session.user && req.path !== '/auth/login' && req.path !== '/auth/register') {
    return res.redirect('/auth/login');
  } else if (req.session.user && req.path === '/') {
    return res.redirect('/auth/profile');
  }
  next();
});
```

d) Routes

Ini mengatur semua permintaan yang dimulai dengan /auth untuk menggunakan rute yang kita buat di authRoutes. Jadi, semua hal yang berkaitan dengan login dan pendaftaran ada di sini.

```
// Routes
app.use('/auth', authRoutes);
```

e) Root Route

Ini menangani permintaan ke halaman utama. Jika pengguna sudah login, mereka akan diarahkan ke halaman profil. Jika belum, mereka akan diarahkan ke halaman login.

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

3.6 Template Engine (EJS)

a) Struktur Dasar HTML (Login, Register, dan Profile)

Halaman ini untuk menentukan pengkodean karakter, mengatur tampilan responsive, dan untuk menghubungkan file CSS untuk mengatur tampilan.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="/styles.css">
  <title>Register</title>
</head>
```

b) Halaman Register

Halaman ini adalah tempat untuk semua elemen pendaftaran. Terdapat kolom untuk memasukkan username, email, phone, instagram, password, dan konfirmasi password. Ada tautan untuk mengarahkan pengguna ke halaman login jika sudah punya akun.

```
<body>
  <div class="container">
    <h2>Register</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" name="email" id="email" required><br>

      <label for="phone">Phone</label>
      <input type="tel" name="phone" id="phone" required><br>

      <label for="instagram">Instagram</label>
      <input type="text" name="instagram" id="instagram" required><br>

      <label for="password">Password</label>
      <input type="password" name="password" id="password" required><br>

      <label for="confirmPassword">Confirm Password</label>
      <input type="password" name="confirmPassword" id="confirmPassword" required><br>

      <button type="submit">Register</button>
    </form>
    <p>Already have an account? <a href="/auth/login">Login here</a></p>
  </div>
```

Ketika form disubmit, script akan memeriksa apakah password dan konfirmasi password cocok. Jika tidak cocok, akan muncul alert dan proses pengiriman form dihentikan.

```
<script>
  document.querySelector('form').addEventListener('submit', function(event) {
    const password = document.getElementById('password').value;
    const confirmPassword = document.getElementById('confirmPassword').value;

    if (password !== confirmPassword) {
      event.preventDefault(); // prevent form submission
      alert('Passwords do not match!');
    }
  });
</script>
```

c) Halaman Profile

Menampilkan pesan selamat datang dengan username pengguna yang terdaftar, menampilkan email pengguna, dan menampilkan tautan untuk logout

```

<body>
  <div class="container">
    <h2>Welcome, <%= user.username %></h2>
    <p>Phone: <%= user.phone %></p>
    <p>Name: <%= user.name %></p>
    <p>Instagram: <%= user.instagram %></p>
    <p>Bio: <%= user.bio %></p>
    <a href="/auth/logout">Logout</a>
  </div>
</body>

```

d) Halaman Login

Form untuk login yang meminta username dan password. Sama seperti form registrasi, ada tombol untuk mengirimkan data dan tautan untuk mendaftar jika belum memiliki akun.

```

<div class="container">
  <h2>Login</h2>
  <form action="/auth/login" method="POST">
    <label for="username">Username</label>
    <input type="text" id="username" name="username" placeholder="Enter your username" required><br>

    <label for="password">Password</label>
    <input type="password" name="password" id="password" placeholder="Enter your password" required><br>

    <button type="submit">Login</button>
  </form>
  <p>Don't have account? <a href="/auth/register">Register here</a></p>

```

Script di halaman ini seharusnya tidak perlu memeriksa kesesuaian password karena tidak ada konfirmasi password dalam form login. Script ini dapat dihapus untuk menghindari kebingungan.

```

<script>
  document.querySelector('form').addEventListener('submit', function(event) {
    const password = document.getElementById('password').value;
    const confirmPassword = document.getElementById('confirmPassword').value;

    if (password !== confirmPassword) {
      event.preventDefault(); // prevent form submission
      alert('Passwords do not match!');
    }
  });
</script>

```

3.7 Fungsional Dasar (auth.js)

Bagian dari sistem otentikasi menggunakan auth.js, termasuk proses pendaftaran, login, dan pengelolaan sesi pengguna:

a) Halaman pendaftaran (Register)

```

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

```

b) Proses pendaftaran pengguna

Menangani pengiriman form pendaftaran. Mengambil data dari form pendaftaran. Mengacak password sebelum, menyimpannya ke database untuk keamanan. Menjalankan query untuk menyimpan data pengguna baru ke dalam tabel users. Jika berhasil, pengguna akan diarahkan ke halaman login.

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

  const hashedPassword = bcrypt.hashSync(password, 10);

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

c) Proses login pengguna

Menangani pengiriman form login. Memeriksa apakah username ada di database.

- Jika username ditemukan, memeriksa apakah password yang dimasukkan cocok dengan yang ada di database.
- Jika cocok, informasi pengguna disimpan di sesi (session), dan pengguna diarahkan ke halaman profil. Jika tidak, pesan kesalahan ditampilkan.

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

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

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

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

d) Proses logout

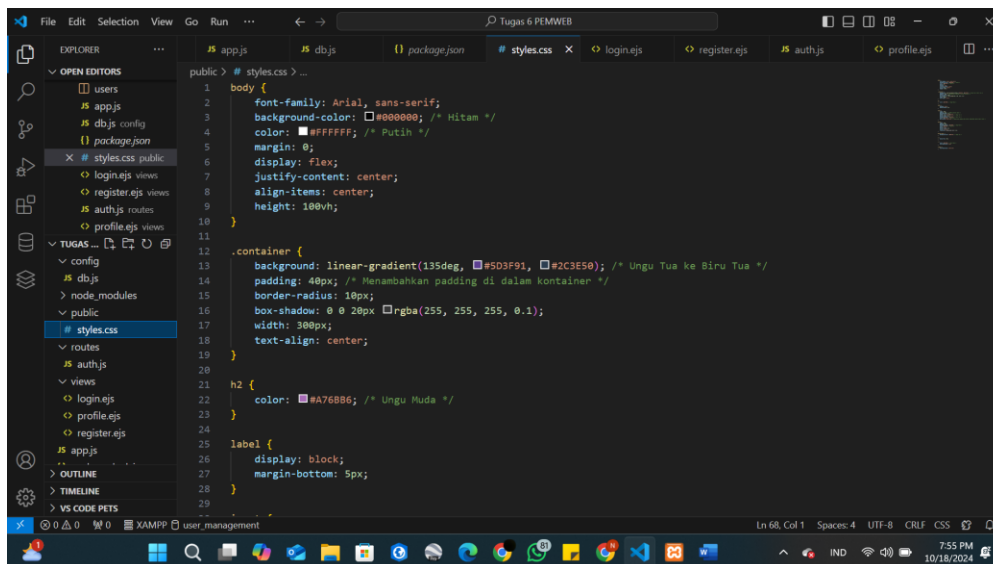
```
// Proses logout
router.get('/logout', (req, res) => {
  req.session.destroy();
  res.redirect('/auth/login');
});
```

3.8 Interaksi dan Validasi

Untuk meningkatkan pengalaman pengguna, kami menambahkan beberapa interaksi dan validasi, seperti konfirmasi sebelum menghapus pengguna dan pesan alert yang muncul setelah menambahkan atau mengupdate data pengguna.

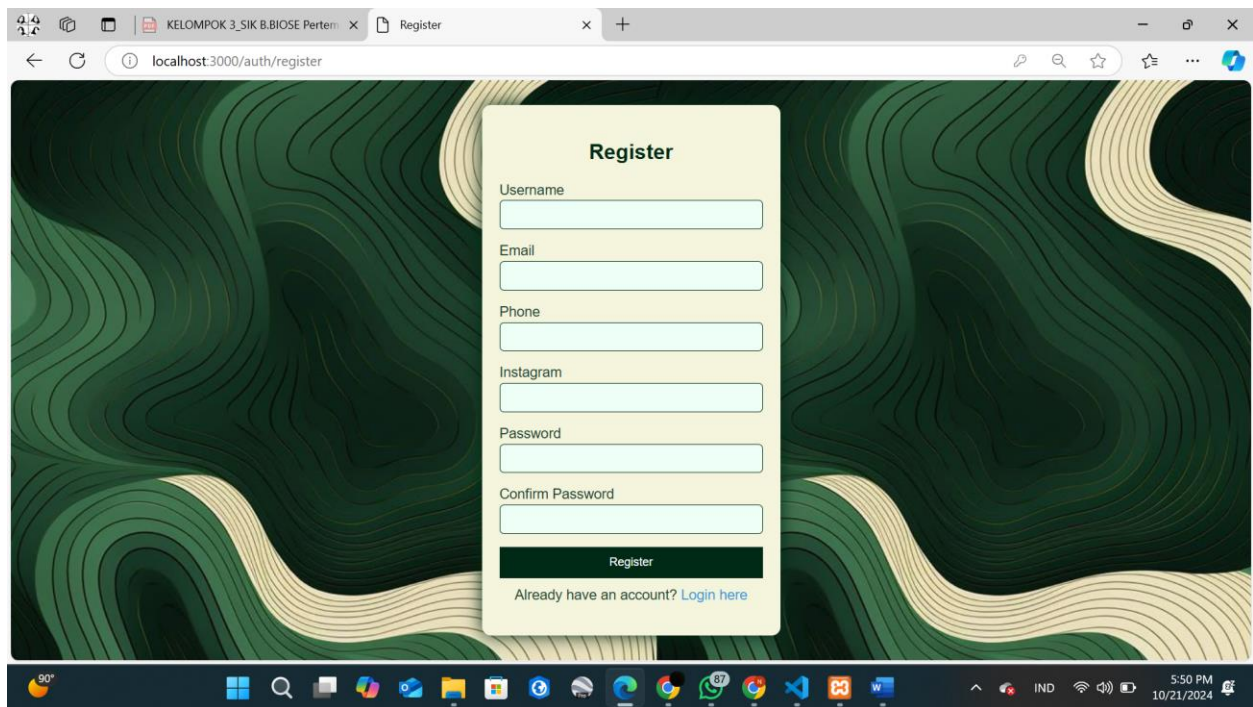
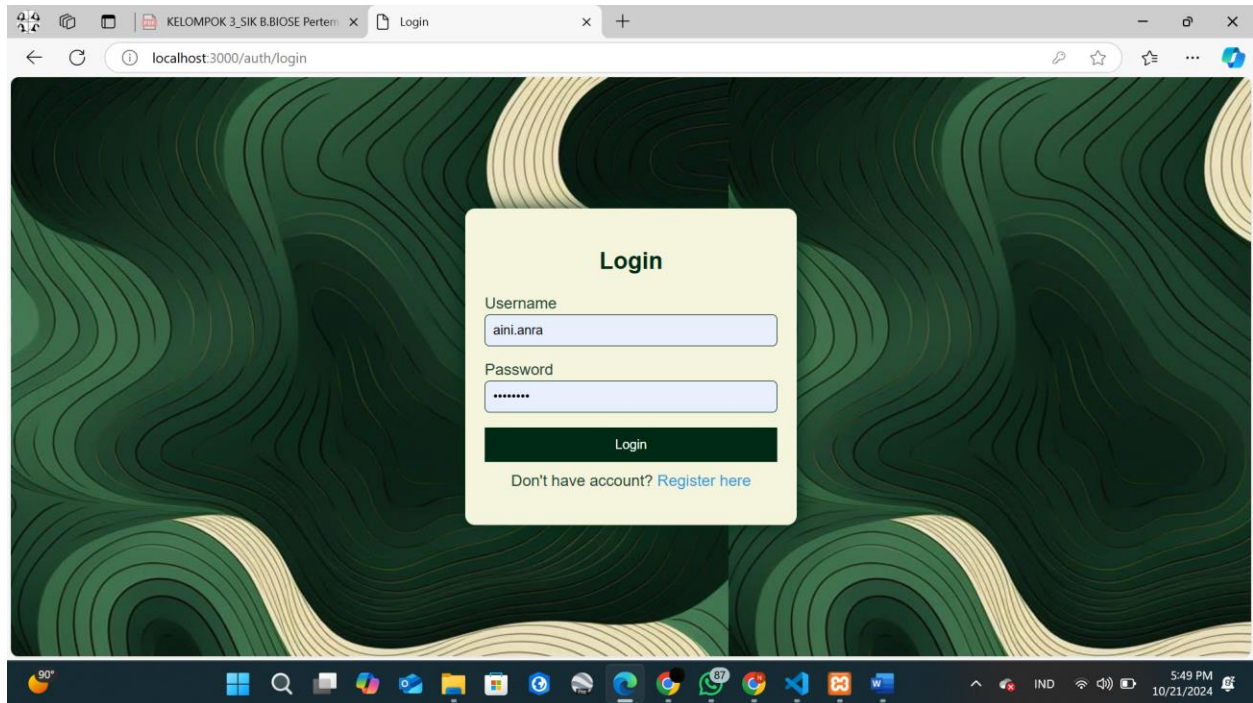
```
<script>
function showAlert(message) {
  alert(message);
}
```

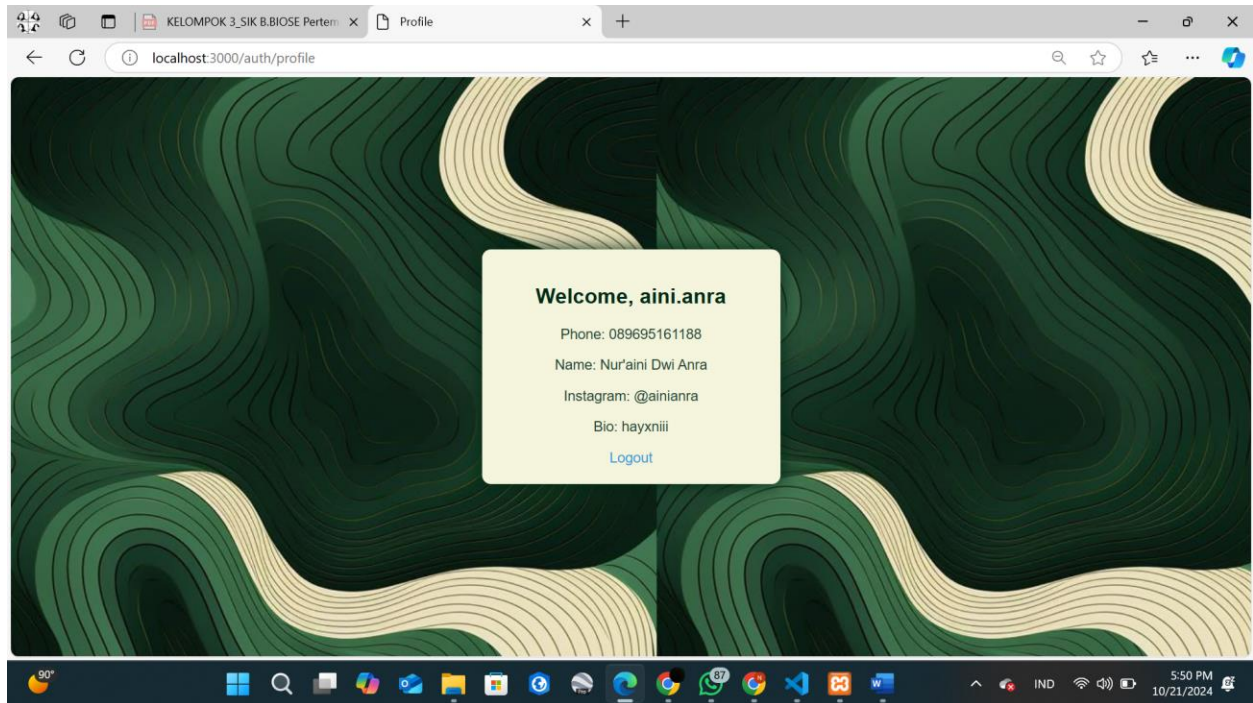
3.9 CSS



4. Hasil

Berikut adalah tampilan dari aplikasi CRUD yang telah di tambahkan CSS setelah dijalankan di localhost:





5. Kesimpulan

Melalui praktikum ini, berhasil mengimplementasikan sistem otentikasi yang terdiri dari proses pendaftaran, login, dan logout menggunakan Express.js dan EJS sebagai template engine. Password pengguna diacak menggunakan bcrypt sebelum disimpan ke database, sehingga meningkatkan keamanan data. Selain itu, kami juga memanfaatkan sesi (session) untuk mengelola status login pengguna. Sistem ini sudah berjalan dengan baik, mampu memverifikasi kredensial pengguna, serta mengarahkan pengguna ke halaman yang sesuai berdasarkan status login mereka.