

Laporan Praktikum
Mata Kuliah Pemrograman Web



Pertemuan 6.Praktikum6
“Session User menggunakan node js dan mysql”

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

Disusun Oleh :
Siti Nurkholizah 2308807
3A - Sistem Informasi Kelautan

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA
2024

I. PENDAHULUAN

Dalam pengembangan aplikasi web, autentikasi dan otorisasi merupakan aspek penting untuk memastikan hanya pengguna yang sah dapat mengakses fitur tertentu. Salah satu metode yang digunakan untuk melacak status login pengguna adalah session.

Session merupakan mekanisme di mana server menyimpan data sementara tentang pengguna selama interaksi dengan aplikasi. Ketika pengguna login, server membuat session unik yang menghubungkan pengguna dengan server melalui cookie. Dengan session, server dapat "mengingat" pengguna tanpa menyimpan informasi sensitif di sisi client, menjadikannya lebih aman.

II. TUJUAN

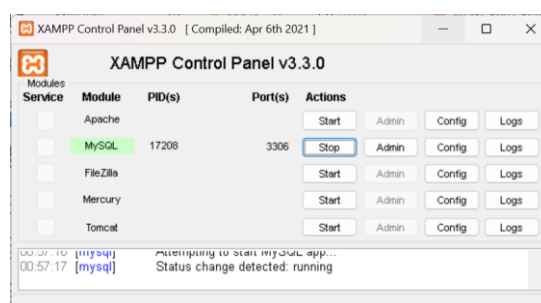
Dalam praktikum ini dapat mempelajari dan mengimplementasikan sistem autentikasi pengguna berbasis session menggunakan Node.js dan MySQL. Praktikum ini juga bertujuan untuk memahami cara mengenkripsi password menggunakan bcryptjs, serta mengelola session secara aman dengan express-session. Selain itu, penggunaan ejs sebagai template engine diharapkan dapat memberikan pengalaman dalam merender halaman web dinamis sesuai dengan status login pengguna. Dengan demikian, peserta dapat memahami konsep session, enkripsi password, serta pengelolaan autentikasi dan otorisasi dalam aplikasi web.

III. ALAT DAN BAHAN

- Laptop
- Node.js yang sudah terinstall di komputer.
- Visual Studio Code
- Browser web (Google Chrome)
- Terminal atau Command Prompt.
- File JS, EJS, dan CSS.

IV. LANGKAH KERJA

1. Sebelum koneksi database ke server aktifkan terlebih dahulu aplikasi XAMPP centang mysql sampai muncul port 3306



2. Buat database terlebih dahulu pada server xampp "user_management"

source code

```
CREATE DATABASE user_management ;
USE user_management;
CREATE TABLE users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  username VARCHAR(100) NOT NULL,
  password VARCHAR(255) NOT NULL
);
```

Lalu run hingga muncul affectrow maka db otomatis telah dibuat

3. Memanggil npm init -y dalam terminal agar node js bisa dijalankan atau node_modules muncul dalam folder explorer
4. Menginstall npm express, mysql, mysql bcryptjs body-parser express-session dan ejs dalam terminal fungsinya agar package.json dan package-lock.json muncul pada folder explorer

Fungsi

- 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.

5. Setelah itu buat folder dengan format struktur seperti ini

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

6. Input kodingan login.ejs, register.ejs, dan profile.ejs

Source code login.ejs dengan cssnya

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Login</title>
  <style>
```

```
body {
  font-family: Arial, sans-serif;
  background-color: #f0f0f0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
}

.container {
  background-color: #bdd8f8;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(49, 51, 184, 0.413);
  width: 300px;
}

h2 {
  text-align: center;
}

label {
  display: block;
  margin-bottom: 5px;
}

input {
  width: 100%;
  padding: 8px;
  margin-bottom: 15px;
  border: 1px solid #3c66c9;
  border-radius: 5px;
}

button {
  width: 100%;
  padding: 10px;
  background-color: #4f9ede;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}
```

```

    button:hover {
        background-color: #054375;
    }

    p {
        text-align: center;
    }

    a {
        color: #165f9bee;
        text-decoration: none;
    }

    a:hover {
        text-decoration: underline;
    }

</style>
</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="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">Login</button>
        </form>
        <p>Don't have a account? <a href="/auth/register">Register
here</a></p>
    </div>
</body>
</html>

```

Source code register.ejs dengan cssnya

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Register</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f0f0;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
    }

    .container {
      background-color: #bdd8f8;
      padding: 20px;
      border-radius: 10px;
      box-shadow: 0 0 10px rgba(49, 51, 184, 0.413);
      width: 300px;
    }

    h2 {
      text-align: center;
    }

    label {
      display: block;
      margin-bottom: 5px;
    }

    input {
      width: 100%;
      padding: 8px;
      margin-bottom: 15px;
      border: 1px solid #3c66c9;
      border-radius: 5px;
    }
```

```
button {
  width: 100%;
  padding: 10px;
  background-color: #4f9ede;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}
```

```
button:hover {
  background-color: #054375;
}
```

```
p {
  text-align: center;
}
```

```
a {
  color: #165f9bee;
  text-decoration: none;
}
```

```
a:hover {
  text-decoration: underline;
}
```

```
</style>
```

```
</head>
```

```
<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" 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>
</div>
</body>
</html>
```

Source code profile.ejs dengan cssnya

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Profile</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f0f0f0;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
    }

    .container {
      background-color: #bdd8f8;
      padding: 20px;
      border-radius: 10px;
      box-shadow: 0 0 10px rgba(49, 51, 184, 0.413);
      width: 300px;
    }

    h2 {
      text-align: center;
    }

    label {
      display: block;
      margin-bottom: 5px;
    }

    input {
      width: 100%;
      padding: 8px;
      margin-bottom: 15px;
```



```

        border: 1px solid #3c66c9;
        border-radius: 5px;
    }

    button {
        width: 100%;
        padding: 10px;
        background-color: #4f9ede;
        color: white;
        border: none;
        border-radius: 5px;
        cursor: pointer;
    }

    button:hover {
        background-color: #054375;
    }

    p {
        text-align: center;
    }

    a {
        color: #165f9bee;
        text-decoration: none;
    }

    a:hover {
        text-decoration: underline;
    }

</style>
</head>
<body>
    <div class="container">
        <h2>Welcome, <%= user.username %></h2>
        <p>Email: <%= user.email %></p>
        <a href="/auth/logout">Logout</a>
    </div>
</body>
</html>

```

7. Sebelum konfigurasi buat kode aplikasi yang dimana file server utama node.js

Source code 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();

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

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

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

// Middleware to check login status
app.use((req, res, next) => {
  if (!req.session.user && req.path !== '/auth/login' && req.path !==
'/auth/register') {
    //If the user is not logged in and trying to access any other page except
login/register
    return res.redirect('/auth/login');
  } else if (req.session.user && req.path === '/') {
    //If user is logged in and tries to access the root route, redirect to profile
    return res.redirect('/auth/profile');
  }
  next();
});

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

// Root Route: Redirect to /auth/login or /auth/profile based on session
app.get('/', (req, res) => {
```

```

    if (req.session.user) {
      return res.redirect('/auth/profile');
    } else {
      return res.redirect('/auth/login');
    }
  });

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

```

8. Selanjutnya untuk konfigurasi dengan db.js

Source code db.js

```

const mysql = require('mysql');

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

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

module.exports = db;

```

9. Setelah itu konfigurasi rute servernya dengan buat folder routes input file auth.js

Source code auth.js

```

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

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

//Proses register user
router.post('/register', (req, res) => {

```

```

const { username, email, password } = req.body;

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');
});

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

// Proses login user
router.post('/login', async (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;
        res.redirect('/auth/profile');
      } else {
        res.send('Incorrect password');
      }
    } else {
      res.send('User not found');
    }
  });
});

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

```

```

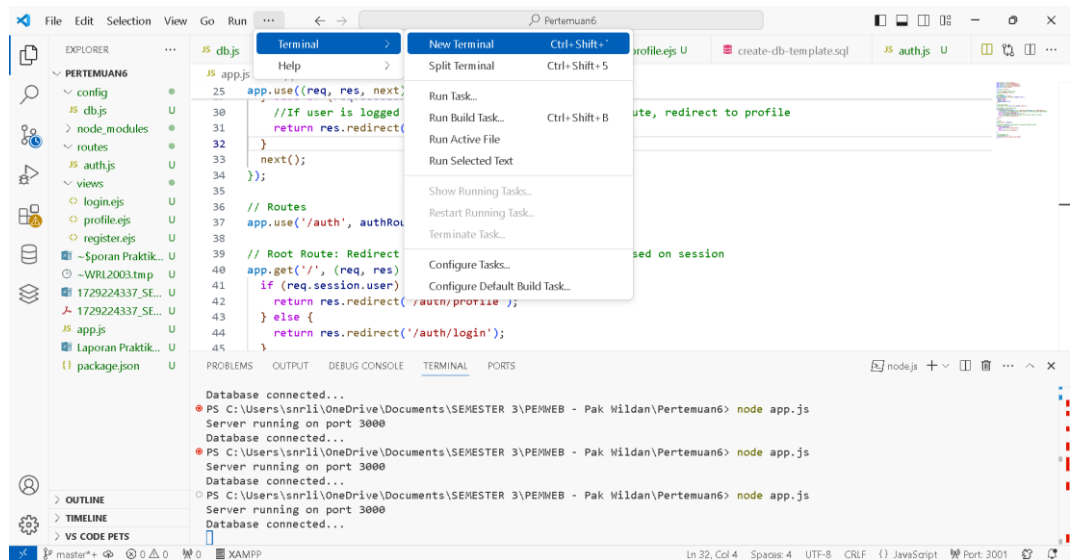
        res.redirect('/auth/login');
    }
});

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

module.exports = router;

```

10. Untuk menjalankan server buka terminal dan jalankan perintah **node app.js**

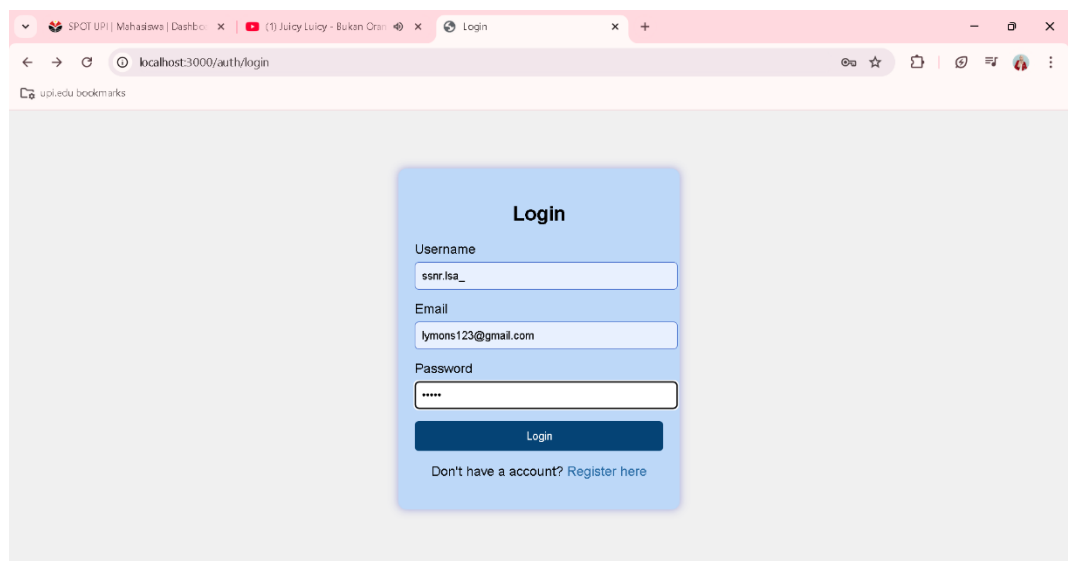


Kemudian server akan berjalan di port 3000 dan dapat mengakses halaman web di browser melalui URL : <http://localhost:3000>.

V. HASIL DAN PEMBAHASAN

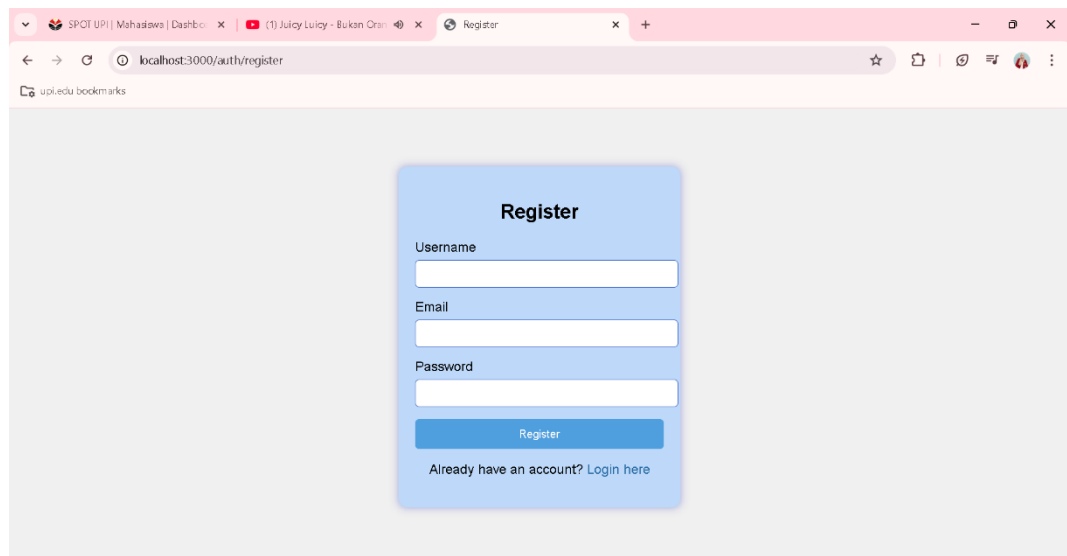
Setelah server dijalankan, halaman login.ejs akan tampil di browser ketika mengakses `http://localhost:3000`. Proses ini menunjukkan bagaimana server Node.js membaca file dari sistem dan mengirimkannya sebagai tanggapan ke browser. Dalam praktik ini, saya berhasil membuat server sederhana dan memahami bagaimana data diambil dari file sistem serta ditampilkan sebagai halaman web.

Tampilan login.ejs



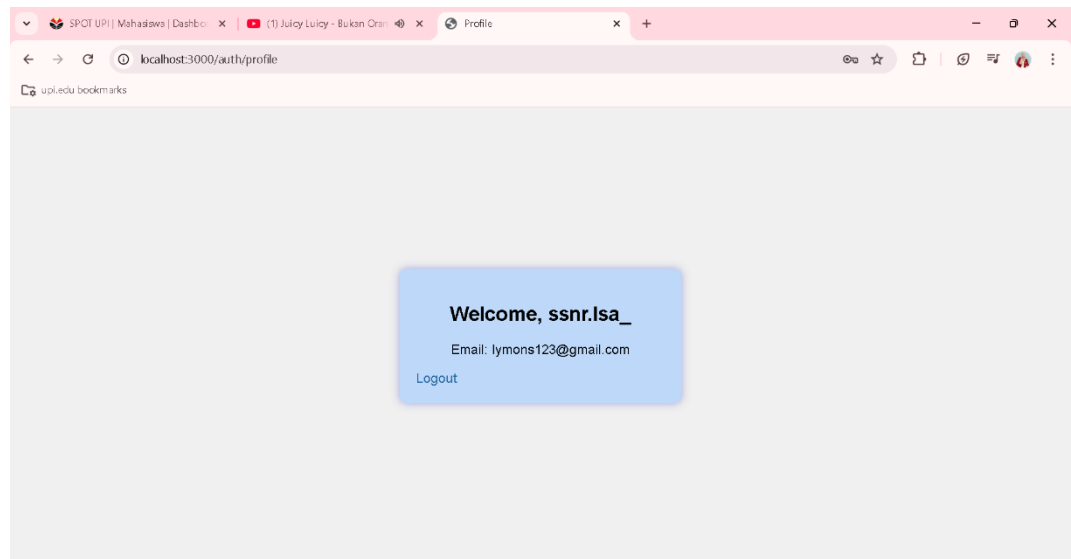
The screenshot shows a web browser window with the address bar displaying `localhost:3000/auth/login`. The page features a light blue login form with the title "Login". The form contains three input fields: "Username" with the value "ssnr.lsa_", "Email" with the value "lymons123@gmail.com", and "Password" with masked characters "*****". Below the inputs is a dark blue "Login" button. At the bottom of the form, there is a link that says "Don't have a account? Register here".

Tampilan register.ejs



The screenshot shows a web browser window with the address bar displaying `localhost:3000/auth/register`. The page features a light blue register form with the title "Register". The form contains three input fields: "Username", "Email", and "Password", all of which are currently empty. Below the inputs is a blue "Register" button. At the bottom of the form, there is a link that says "Already have an account? Login here".

Tampilan profile.ejs ketika setelah berhasil login



VI. KESIMPULAN

Dalam praktikum ini, bahwa autentikasi berbasis session dapat diimplementasikan secara efisien menggunakan **Node.js**, **MySQL**, serta dukungan **bcryptjs** untuk enkripsi password dan **express-session** untuk manajemen session. Session menjaga status login pengguna dengan aman selama interaksi dengan aplikasi, memungkinkan akses ke halaman yang dibatasi tanpa memasukkan ulang kredensial. Penggunaan **bcryptjs** melindungi password dengan enkripsi, sedangkan **ejs** memungkinkan perenderan halaman web dinamis sesuai status login. Praktikum ini menunjukkan pemahaman mengenai autentikasi, session, dan enkripsi data dalam aplikasi web yang aman.