Laporan Praktikum

Mata Kuliah Pemrograman Web

Pertemuan 6

"SESSION"

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

Disusun Oleh :

Silvia Isti Lestary

2311883

PROGRAM STUDI SISTEM INFORMASI KELAUTAN

UNIVERSITAS PENDIDIKAN INDONESIA

2024

I. PENDAHULUAN

Session adalah mekanisme yang digunakan dalam aplikasi web untuk menyimpan informasi pengguna secara sementara saat mereka berinteraksi dengan aplikasi tersebut. Dalam konteks Node js, session digunakan untuk melacak dan mengingat status pengguna di antara permintaan HTTP (request) yang berbeda.

II. ALAT DAN BAHAN

1. Laptop

2. Browser atau terminal (Node.js untuk menjalankan JavaScript di luar

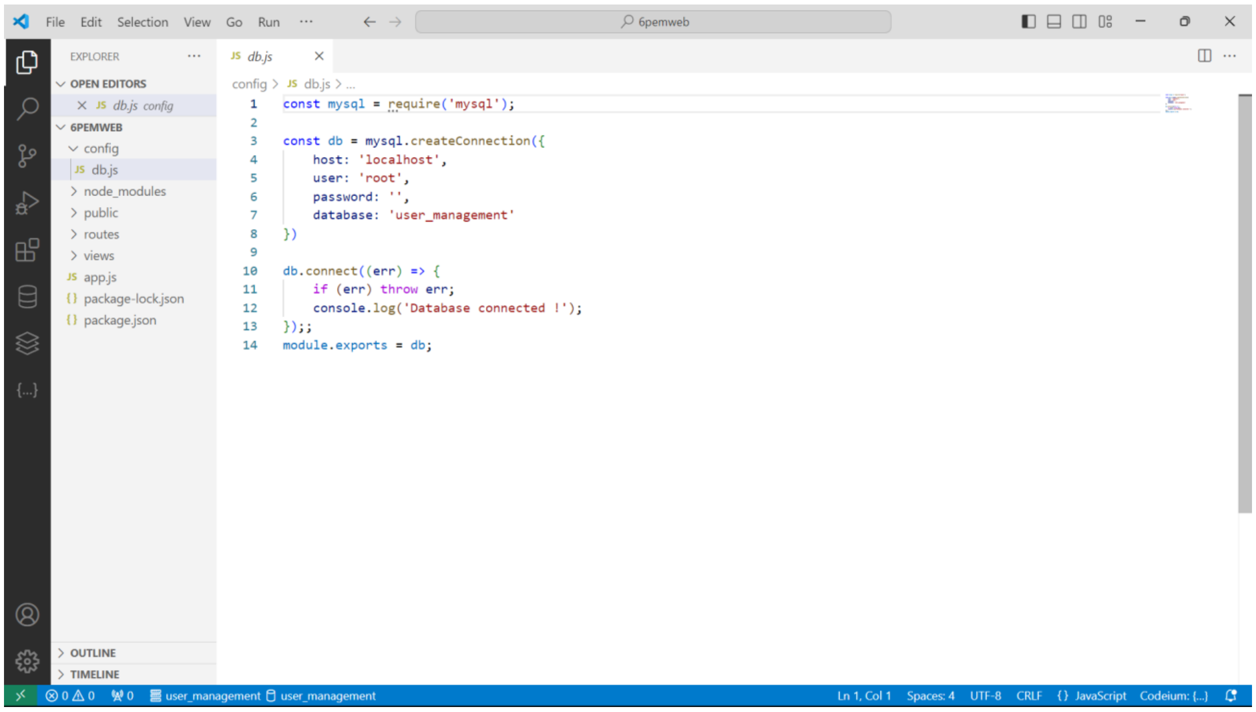
browser

3. Visual Studio Code

4. Node.js

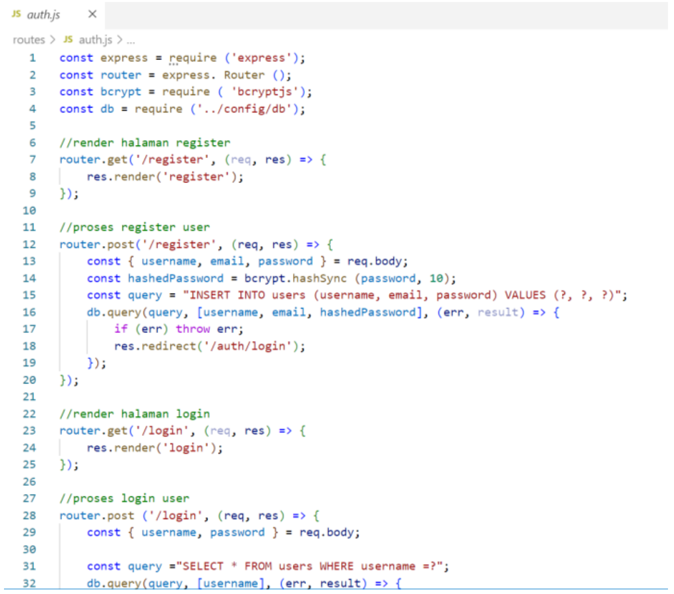
III. PENJELASAN

1. Membuat config, code dibawah digunakan untuk membuat server utama.

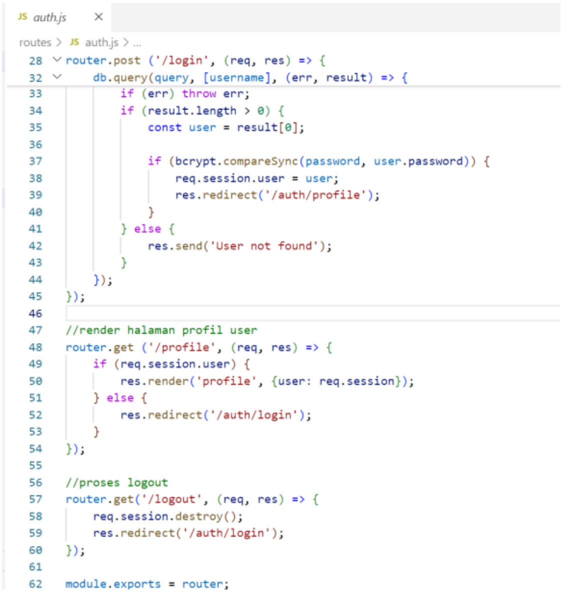


```
const mysql = require('mysql');

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;
```

2. Membuat auth, yaitu sebagai arah webste yang dibuat.

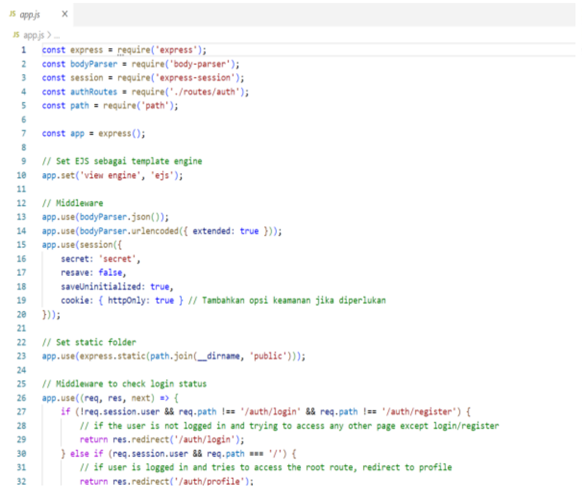


```javascript
const express = require ('express');
const router = express. Router ();
const bcrypt = require ( 'bcryptjs');
const db = require ('../config/db');

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

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

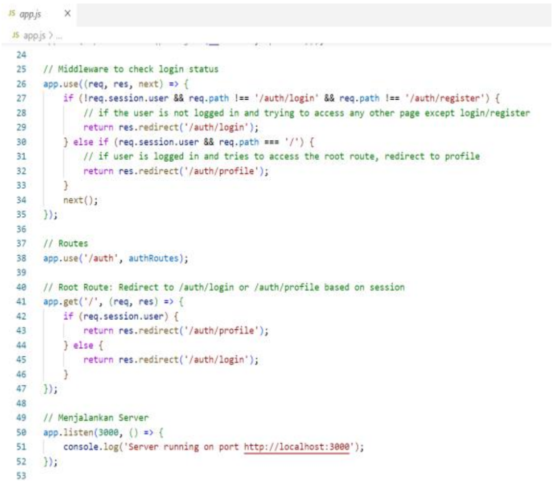


```javascript
router.post ('/login', (req, res) => {
    db.query(query, [username], (err, result) => {
        if (err) throw err;
        if (result.length > 0) {
            const user = result[0];

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

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

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

module.exports = router;
```

3. Membuat server dari node js dengan beberapa module yang sudah disiapkan.

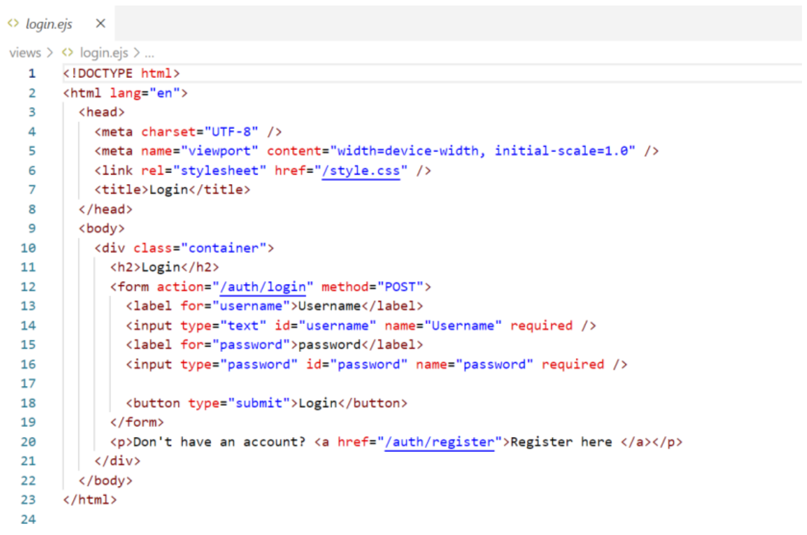


```javascript
const express = require('express');
const bodyParser = require('body-parser');
const session = require('express-session');
const authRoutes = require('./routes/auth');
const path = require('path');

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,
    cookie: { httpOnly: true } // Tambahkan opsi keamanan jika diperlukan
}));

// 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');
```

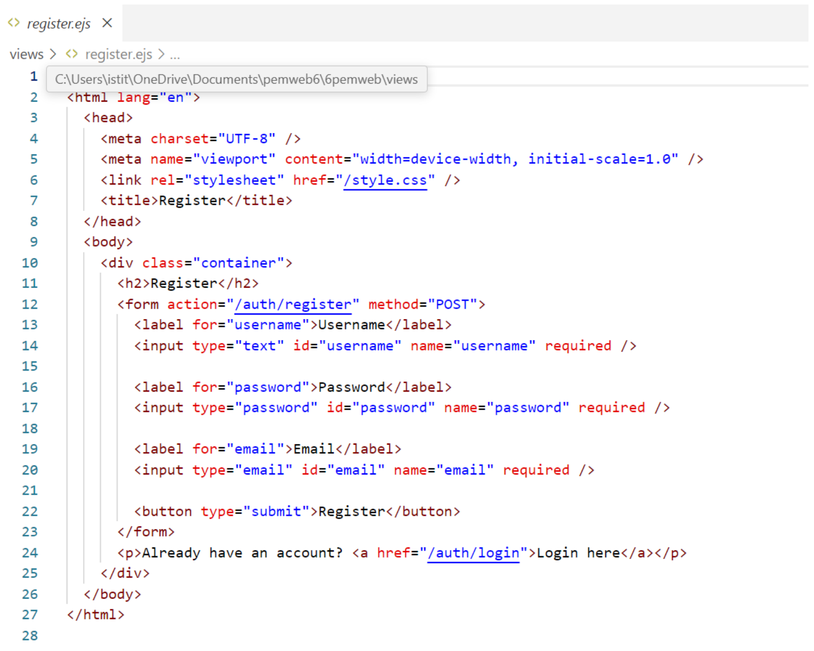


```javascript

// 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 http://localhost:3000');
});

```

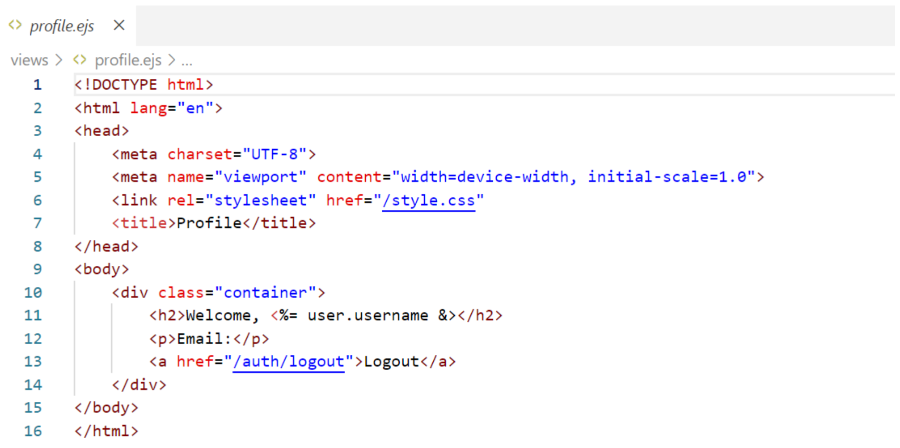
4. Membuat kode tampilan yaitu untuk halaman login, register, dan profile



```html
<!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="/style.css" />
    <title>Login</title>
  </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>

```

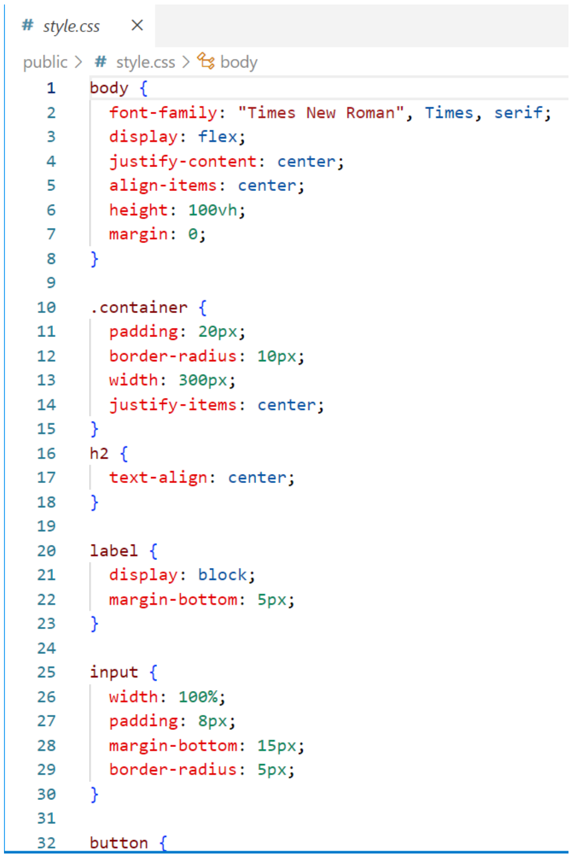

register.ejs
views > register.ejs > ...
C:\Users\istit\OneDrive\Documents\pemweb6\6pemweb\views

```html
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <link rel="stylesheet" href="/style.css" />
    <title>Register</title>
  </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="password">Password</label>
        <input type="password" id="password" name="password" required />

        <label for="email">Email</label>
        <input type="email" id="email" name="email" required />

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

```

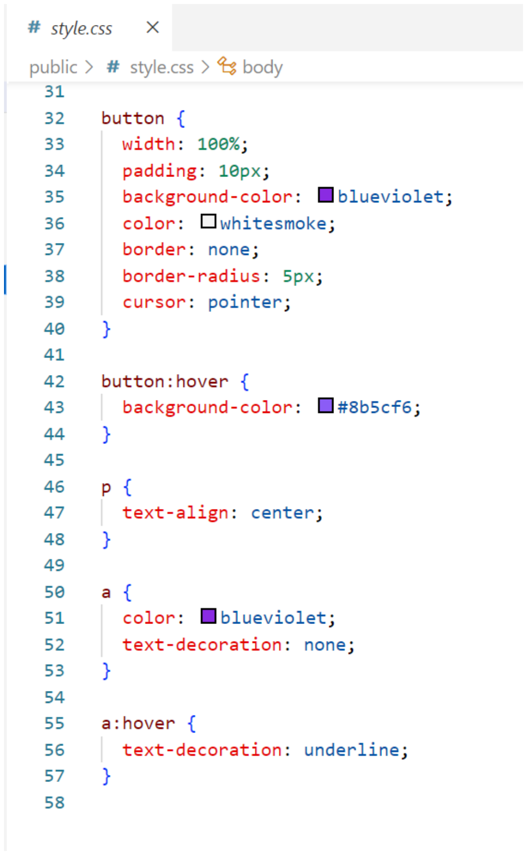


```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/style.css"
    <title>Profile</title>
</head>
<body>
    <div class="container">
        <h2>Welcome, <%= user.username &></h2>
        <p>Email:</p>
        <a href="/auth/logout">Logout</a>
    </div>
</body>
</html>
```

5. Membuat CSS untuk menentukan bagaimana dokumen dan website akan disajikan.



```css
body {
  font-family: "Times New Roman", Times, serif;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
}

.container {
  padding: 20px;
  border-radius: 10px;
  width: 300px;
  justify-items: center;
}
h2 {
  text-align: center;
}

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

input {
  width: 100%;
  padding: 8px;
  margin-bottom: 15px;
  border-radius: 5px;
}

button {
```



```css

button {
  width: 100%;
  padding: 10px;
  background-color: blueviolet;
  color: whitesmoke;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
  background-color: #8b5cf6;
}

p {
  text-align: center;
}

a {
  color: blueviolet;
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}

```

IV. KESIMPULAN

Materi session memberikan pengetahuan mengenai pengelolaan informasi user yang digunakan sementara. Dengan pengelolaan session yang efektif, keamanan data terjaga, akses tidak sah dapat dicegah, dan penggunaan sumber daya sistem dioptimalkan. Konsep ini sangat penting dalam aplikasi yang memerlukan autentikasi, seperti platform e-commerce dan perbankan online, di mana session memastikan komunikasi yang lancar dan aman antara pengguna dan server.