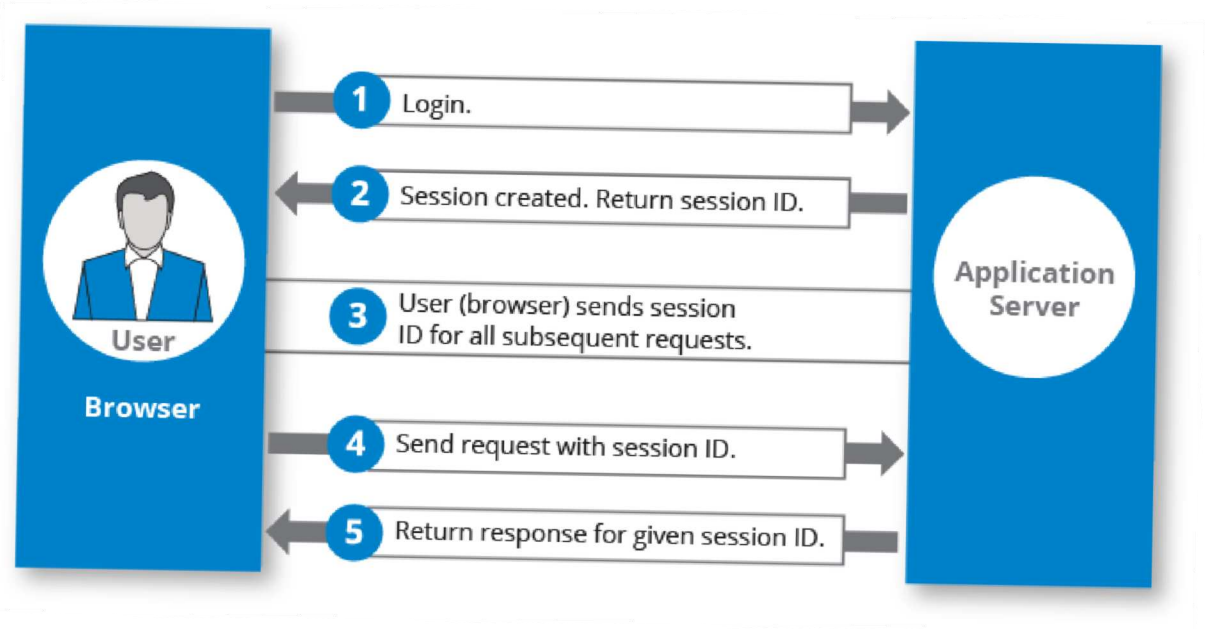
SESSION

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.



Berikut adalah beberapa poin penting tentang session:

1. **Penyimpanan Informasi Sementara**: Ketika pengguna login, informasi seperti ID pengguna, username, atau token disimpan dalam session, sehingga server dapat mengingat pengguna tersebut selama sesi mereka berlangsung.
2. **Terikat pada Pengguna**: Setiap pengguna yang terhubung ke aplikasi memiliki session unik. Ketika pengguna melakukan request baru (misalnya berpindah halaman), session yang terkait dengannya akan diidentifikasi dan diakses.
3. **Berbasis Cookie**: Session biasanya disimpan di server, dan untuk melacak session tersebut, browser pengguna akan menyimpan cookie yang berisi ID session. Server akan memeriksa cookie ini untuk mengidentifikasi session pengguna.
4. **Contoh Penggunaan**:
    - Ketika pengguna login, aplikasi menyimpan informasi login dalam session.
    - Pada request berikutnya, aplikasi dapat memeriksa session untuk memastikan bahwa pengguna sudah login.
    - Jika pengguna logout, session akan dihapus, sehingga pengguna harus login kembali.

Contoh Penggunaan dalam Node.js:

```
const session = require('express-session');

app.use(session({
    secret: 'mySecretKey', // Kunci rahasia untuk mengenkripsi session
    resave: false, // Session tidak akan disimpan ulang jika tidak diubah
    saveUninitialized: true // Session yang baru akan disimpan walau tidak
diisi
}));
```

- `secret`: Kunci rahasia yang digunakan untuk menandatangani cookie session.
- `resave`: Menentukan apakah session harus disimpan ulang meskipun tidak ada perubahan.
- `saveUninitialized`: Menyimpan session yang baru, bahkan jika tidak ada data yang diisi.

Contoh Penggunaan dalam Aplikasi Login:

- Saat pengguna berhasil login, session bisa disimpan dengan cara:

```
req.session.user = { id: user.id, username: user.username };
```

- Ketika pengguna mengakses halaman lain, aplikasi bisa memeriksa apakah session pengguna ada:

```
if (req.session.user) {
    // Jika pengguna sudah login, tampilkan halaman profil
} else {
    // Jika belum login, arahkan ke halaman login
}
```

PRAKTIK SESSION

1. Persiapan

- **Software yang dibutuhkan:**
  - Node.js (pastikan sudah terinstal)
  - MySQL (bisa menggunakan XAMPP atau standalone)
  - Postman atau browser untuk testing API
- **Langkah Awal:**
  - Buat database bernama `user_management` dan tabel `users` dengan struktur berikut:

```
CREATE DATABASE user_management;

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

2. Inisialisasi Node.js Project

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.

3. Struktur Folder

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

4. Konfigurasi Koneksi Database (config/db.js)

Buat file `db.js` untuk mengatur koneksi ke database.

```
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;
```

5. Server Setup (app.js)

Buat file `app.js` untuk menginisialisasi server Express dan mengatur middleware.

```
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');
});
```

6. Membuat Tampilan Views

Halaman Register (views/register.ejs)

```
<!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>
<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>
```

Halaman Login (views/login.ejs)

```
<!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>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>
```

Halaman Profil (views/profile.ejs)

```
<!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>Profile</title>
</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. Rute Login, Logout, dan Register (routes/auth.js)

Buat file `auth.js` di dalam folder `routes` untuk mengelola autentikasi pengguna.

```
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) => {
        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;
```

8. CSS (public/styles.css)

Berikut adalah contoh gaya sederhana untuk tampilan form:

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

.container {
    background-color: #fff;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    width: 300px;
}

h2 {
    text-align: center;
}

label {
```

```
    display: block;
    margin-bottom: 5px;
}

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

button {
    width: 100%;
    padding: 10px;
    background-color: #5cb85c;
    color: white;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}

button:hover {
    background-color: #4cae4c;
}

p {
    text-align: center;
}

a {
    color: #5cb85c;
    text-decoration: none;
}

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

9. Testing

- Menjalankan server: `node app.js`
- Buka browser dan akses: http://localhost:3000/

TUGAS

1. **Tugas Hari Ini:**

   Ikuti latihan praktikum di atas, kumpulkan dalam bentuk link github dan laporan praktikum berupa file pdf berisikan penjelasan dan pemahaman yang di dapat setelah pembelajaran hari ini. Dikumpulkan di hari yang sama di spot, pukul 23.50.

2. **Tugas Kuis:**

Buatlah sebuah web dengan tema yang tematik untuk dan memiliki fitur sebagai berikut:

a. Memiliki landing page (halaman utama)
b. Registrasi akun, login, logout (Session)
c. Memiliki halaman profile dengan informasi yang lengkap
d. Tambahkan Update Profile dan Hapus Akun untuk melengkapi fitur profile
e. Buat 1 menu yang mengimplementasikan CRUD sesuai dengan kebutuhan web. (misalnya CRUD berita, CRUD agenda, dll)
f. Buat tampilan yang menarik

Tugas dikerjakan secara berkelompok 2-3 orang (tidak lebih). Dan file dikumpulkan adalah :

a. Link github
b. Dokumentasi kode dan panduan penggunaan
c. Video Presentasi web yang di buat 20-30 menit.

Tugas dikumpulkan hari kamis, 24 Oktober 2024, 23:50