Laporan Praktikum
PEMEMOGRAMAN BERORIENTASI OBJEK

Pertemuan 7. Praktikum 6

“Session”

Dosen Pengampu :

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

Disusun Oleh :

Rere Citra Ramadhan

(2306182)

PROGRAM STUDI SISTEM INFORMASI KELAUTAN

UNIVERSITAS PENDIDIKAN INDONESIA

2024

```
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();

```

- **express**: Framework web untuk Node.js.
- **body-parser**: Middleware untuk menguraikan body dari permintaan HTTP (POST).
- **express-session**: Middleware untuk mengelola sesi pengguna.
- **authRoutes**: Mengimpor rute autentikasi dari file auth.js.
- **path**: Modul bawaan Node.js untuk menangani path file.
- Membuat instance dari aplikasi Express.

```

app.set('view engine', 'ejs');
app.set('views', path.join(__dirname, 'views'));

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

app.use(session({
    secret: 'secret',
    resave: false,
    saveUninitialized: true
}));

```

- Mengatur EJS sebagai engine untuk rendering tampilan.
- Menentukan folder views sebagai lokasi file EJS.
- Menggunakan body-parser untuk menguraikan JSON dan data URL-encoded dari permintaan HTTP.
- Mengkonfigurasi sesi pengguna dengan beberapa opsi:
    - secret: Kunci untuk menandatangani sesi.
    - resave: Tidak menyimpan kembali sesi jika tidak ada perubahan.
    - saveUninitialized: Menyimpan sesi baru yang belum diubah.

```

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

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();
});

```

- Memeriksa apakah pengguna telah login (memiliki sesi pengguna).
- Jika pengguna belum login dan mencoba mengakses rute selain halaman login atau pendaftaran, mereka akan dialihkan ke halaman login.
- Jika pengguna sudah login dan mengakses halaman utama (/), mereka akan dialihkan ke halaman profil.

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

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

app.get('/page', (req, res) => {
    res.render('page', { title: 'Page', content: 'This is the page.' });
});

app.get('/', (req, res) => {
    res.render('home', { title: 'Home Page', message: 'Welcome to home!' });

});

app.listen(3004, () => {
    console.log('Server running on port 3004');
});
```

- Menggunakan rute autentikasi yang diimpor dari auth.js.
- Menangani rute utama (/) dan mengalihkan pengguna sesuai status sesi mereka.
- Menangani rute /page untuk merender tampilan page.ejs dengan judul dan konten yang sesuai.
- Menjalankan server di port 3004 dan mencetak pesan ke konsol saat server aktif.

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


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

- **express**: Framework web untuk Node.js.
- **router**: Membuat router baru untuk mendefinisikan rute autentikasi.
- **rypt**: Library untuk hashing dan membandingkan password.
- **db**: Mengimpor konfigurasi database dari file db.js, yang biasanya mengandung koneksi ke database.
- Menangani permintaan GET untuk /register. Jika pengguna mengakses rute ini, akan merender tampilan register.ejs.

```

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');
    });
});

```

- Menangani permintaan POST untuk /register.
- Mengambil username, email, dan password dari body permintaan.
- Meng-hash password menggunakan bcrypt.hashSync dengan 10 putaran untuk keamanan.
- Menyimpan data pengguna ke dalam database menggunakan query SQL

INSERT.

- Setelah berhasil, pengguna dialihkan ke halaman login (/auth/login).

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


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

- Menangani permintaan GET untuk /login. Jika pengguna mengakses rute ini, akan merender tampilan login.ejs.
- Menangani permintaan POST untuk /login.
- Mengambil username dan password dari body permintaan.
- Mencari pengguna di database menggunakan query SQL SELECT.
- Jika pengguna ditemukan, membandingkan password yang diberikan dengan yang tersimpan di database menggunakan bcrypt.compareSync.
- Jika password cocok, menyimpan data pengguna ke dalam sesi (req.session.user) dan mengalihkan pengguna ke halaman profil (/auth/profile). Jika tidak, menampilkan pesan kesalahan yang sesuai.

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


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

module.exports = router;
```

- Menangani permintaan GET untuk /profile.
- Memeriksa apakah pengguna telah login (yaitu, apakah req.session.user ada).
- Jika pengguna login, merender tampilan profile.ejs dengan data pengguna. Jika tidak, pengguna dialihkan ke halaman login.
- Menangani permintaan GET untuk /logout.
- Menghancurkan sesi pengguna saat ini (req.session.destroy()), sehingga pengguna dianggap keluar.

- Setelah keluar, pengguna dialihkan ke halaman login.
- Mengekspor router agar bisa digunakan di file utama aplikasi (seperti app.js), sehingga rute-rute autentikasi bisa diakses.