Panduan Kode Program

Mata Kuliah Pemrograman WEB dan PBO

Ujian Tengah Semester

“Session dan CRUD”

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

Disusun Oleh :
Dzakiya Fikri Murtianingsih
2300313

PROGRAM STUDI SISTEM INFORMASI KELAUTAN
UNIVERSITAS PENDIDIKAN INDONESIA

2024

I. DOKUMENTASI KODE PROGRAM

Kode program yang dibuat menggunakan view engine ejs dengan bahasa pemrograman JavaScript ini membuat sistem pengelolaan kamar penginapan Stay Here yang bertujuan untuk memudahkan admin memantau kamar mana saja yang sedang terisi atau terdapat tamu, kemudian melihat kamar mana saja yang sudah mendekati tanggal check out, dan melihat jenis kamar apa saja yang masih kosong.

A. Models/

- **db.js**

```
const mysql = require('mysql2');

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

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

module.exports = db;
```

B. Public/

- **style.css**

```
body {
    background-color: #f4f4f4;
    margin: 0;
    padding: 20px;
}

h1 {
    color: #333;
    text-align: center;
}

h2{
    color: #333;
    text-align: center;
}

form {
    margin: 20px 0;
    padding: 20px;
    border-radius: 5px;
```

```css
}

input[type="email"],
input[type="password"],
input[type="number"],
input[type="date"],
select {
    width: 98%;
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #9e9e9e;
    border-radius: 5px;
}

button {
    padding: 10px 15px;
    background-color: #d94da8;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-family: 'Times New Roman', Times, serif;
    font-size: 16px;
}

button:hover {
    background-color: #a10b76;
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-top: 20px;
}

table, th, td {
    border: 1px solid #ccc;
}

th, td {
    padding: 10px;
    text-align: center;
}

a {
    color: #007bff;
    text-decoration: none;
}
```

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

.error {
    color: red;
}

::placeholder{
    font-family: 'Times New Roman', Times, serif;
}

#del{
    padding: 5px 7px;
    background-color: #d94da8;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-family: 'Times New Roman', Times, serif;
    font-size: 16px;
}

#up{
    padding: 5px 7px;
    background-color: #d94da8;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-family: 'Times New Roman', Times, serif;
    font-size: 16px;
}
```

C. Routes/

- **auth.js**

```js
const express = require('express');
const db = require('../models/db');
const router = express.Router();

// Route untuk halaman login
router.get('/login', (req, res) => {
    res.render('login', { error: null });
});

// Route untuk memproses login
router.post('/login', (req, res) => {
```

```
    const { email, password } = req.body;
    db.query('SELECT * FROM admins WHERE email = ? AND password =
?', [email, password], (err, results) => {
        if (err) throw err;
        if (results.length > 0) {
            req.session.isAuthenticated = true;
            res.redirect('/kamar');
        } else {
            res.render('login', { error: 'Email atau password
salah' });
        }
    });
});

module.exports = router;
```

- **kamar.js**

```
const express = require('express');
const db = require('../models/db');
const router = express.Router();

// Middleware untuk memeriksa autentikasi
router.use((req, res, next) => {
    if (!req.session.isAuthenticated) {
        return res.redirect('/auth/login');
    }
    next();
});

// Route untuk halaman Data Kamar
router.get('/', (req, res) => {
    db.query('SELECT * FROM kamar', (err, results) => {
        if (err) throw err;
        res.render('kamar', { kamars: results });
    });
});

// Route untuk menambah kamar
router.post('/', (req, res) => {
    const { jumlah_kamar, tipe_kamar, harga, check_in, check_out
} = req.body;
    db.query('INSERT INTO kamar (jumlah_kamar, tipe_kamar, harga,
check_in, check_out) VALUES (?, ?, ?, ?, ?)',
    [jumlah_kamar, tipe_kamar, harga, check_in, check_out], (err)
=> {
        if (err) throw err;
        res.redirect('/kamar');
```

```
    });
});

// Route untuk menghapus kamar
router.post('/delete/:id', (req, res) => {
    db.query('DELETE FROM kamar WHERE id = ?', [req.params.id],
(err) => {
        if (err) throw err;
        res.redirect('/kamar');
    });
});

// Route untuk mengedit kamar
router.get('/edit/:id', (req, res) => {
    db.query('SELECT * FROM kamar WHERE id = ?', [req.params.id],
(err, results) => {
        if (err) throw err;
        res.render('editKamar', { kamar: results[0] });
    });
});

router.post('/edit/:id', (req, res) => {
    const { jumlah_kamar, tipe_kamar, harga, check_in, check_out
} = req.body;
    db.query('UPDATE kamar SET jumlah_kamar = ?, tipe_kamar = ?,
harga = ?, check_in = ?, check_out = ? WHERE id = ?',
    [jumlah_kamar, tipe_kamar, harga, check_in, check_out,
req.params.id], (err) => {
        if (err) throw err;
        res.redirect('/kamar');
    });
});

module.exports = router;
```

D. Views/

- **editKamar.ejs**

```
<!DOCTYPE html>
<html>
<head>
    <title>Edit Kamar</title>
    <link rel="stylesheet" href="/style.css">
</head>
<body>
    <h1>Edit Kamar</h1>
    <form action="/kamar/edit/<%= kamar.id %>" method="POST">
```

```
        <input type="number" name="jumlah_kamar" value="<%=
kamar.jumlah_kamar %>" required>
        <select name="tipe_kamar" required>
            <option value="regular" <%= kamar.tipe_kamar ===
'regular' ? 'selected' : '' %>>Regular</option>
            <option value="classic" <%= kamar.tipe_kamar ===
'classic' ? 'selected' : '' %>>Classic</option>
            <option value="luxury" <%= kamar.tipe_kamar ===
'luxury' ? 'selected' : '' %>>Luxury</option>
        </select>
        <input type="number" name="harga" value="<%= kamar.harga
%>" required>
        <input type="text" name="check_in" value="<%=
kamar.check_in %>" required>
        <input type="text" name="check_out" value="<%=
kamar.check_out %>" required>
        <button type="submit">Update Kamar</button>
    </form>
</body>
</html>
```

- **index.ejs**

```
<!DOCTYPE html>
<html>
<head>
    <title>Halaman Utama</title>
    <!-- <link rel="stylesheet" href="/style.css"> -->
     <style>
        body{
            text-align: center;
            margin-top: 220px;
            background-color: #f4f4f4;
        }

        #log{
            padding: 10px 15px;
            background-color: #d94da8;
            color: #fff;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            font-family: 'Times New Roman', Times, serif;
            font-size: 16px;
        }
     </style>
</head>
<body>
    <h1>Stay Here</h1>
```

```
    <a href="/auth/login" id="log">Login Admin</a>
</body>
</html>
```

- **kamar.ejs**

```
<!DOCTYPE html>
<html>
<head>
    <title>Data Kamar</title>
    <link rel="stylesheet" href="/style.css">
</head>
<body>
    <h1>Data Kamar</h1>
    <form action="/kamar" method="POST">
        <input type="number" name="jumlah_kamar"
placeholder="Nomor Kamar" required>
        <select name="tipe_kamar" required>
            <option value="" disabled selected>Pilih Tipe
Kamar</option>
            <option value="regular">Regular</option>
            <option value="classic">Classic</option>
            <option value="luxury">Luxury</option>
        </select>
        <input type="number" name="harga" placeholder="Harga"
required>
        <h4>Check In</h4>
        <input type="date" name="check_in" placeholder="Waktu
Check In" required>
        <h4>Check Out</h4>
        <input type="date" name="check_out" placeholder="Waktu
Check Out" required> <br> <br>
        <button type="submit">Tambah Kamar</button>
    </form>

    <h2>Daftar Kamar</h2>
    <table>
        <thead>
            <tr>
                <th>Nomor Kamar</th>
                <th>Tipe Kamar</th>
                <th>Harga</th>
                <th>Check In</th>
                <th>Check Out</th>
                <th>Aksi</th>
            </tr>
        </thead>
        <tbody>
```

```
            <% kamars.forEach(kamar => { %>
                <tr>
                    <td><%= kamar.jumlah_kamar %></td>
                    <td><%= kamar.tipe_kamar %></td>
                    <td><%= kamar.harga %></td>
                    <td><%= kamar.check_in %></td>
                    <td><%= kamar.check_out %></td>
                    <td>
                        <form action="/kamar/delete/<%= kamar.id
%>" method="POST" style="display:inline;">
                            <button type="submit"
id="del">Hapus</button>
                        </form>
                        <a href="/kamar/edit/<%= kamar.id %>"
id="up">Edit</a>
                        <!-- <a href="/kamar/delete/<%= kamar.id
%>">Hapus</a> -->
                    </td>
                </tr>
            <% }); %>
        </tbody>
    </table>
</body>
</html>
```

- **login.ejs**

```
<!DOCTYPE html>
<html>
<head>
    <title>Login Admin</title>
    <link rel="stylesheet" href="/style.css">
</head>
<body>
    <h1>Hai, Min-stay!!</h1>
    <form action="/auth/login" method="POST">
        <input type="email" name="email" placeholder="Email"
required>
        <input type="password" name="password"
placeholder="Password" required>
        <button type="submit">Login</button>
    </form>
    <% if (error) { %>
        <p class="error"><%= error %></p>
    <% } %>
</body>
</html>
```

E. app.js

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

const app = express();
const PORT = 3300;

app.set('view engine', 'ejs');
app.use(bodyParser.urlencoded({ extended: true }));
app.use(session({
    secret: 'secret-key',
    resave: false,
    saveUninitialized: true
}));
app.use(express.static('public'));

app.use('/auth', authRoutes);
app.use('/kamar', kamarRoutes);

app.get('/', (req, res) => {
    res.render('index');
});

app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

II. PANDUAN KODE

1. **Struktur proyek**
   - **Models**
     Menyimpan file untuk menghubungkan aplikasi dengan database.
   - **Routes**
     Menyimpan file untuk mengatur rute aplikasi (routes) yang menangani logika dan permintaan HTTP.
   - **Views**
     Menyimpan template EJS untuk tampilan antarmuka pengguna.
   - **Public**
     Menyimpan file CSS dan file statis lainnya.
   - **App.js**
     File utama yang mengatur server dan middleware.
2. **Alur proses**
   a. **Memulai sistem**
      1. **Pengaturan awal**
         - Impor paket yang sudah diinstall sebelumnya ke dalam **app.js** seperti Express, EJS, dan body-parser.
         - Mengatur session dengan express-session untuk manajemen autentikasi.
         - Menentukan port untuk server.

         Kodenya:

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

const app = express();
const PORT = 3300;
```

      2. **Mengatur view engine**
         - Mengatur EJS sebagai template engine untuk rendering HTML.

         Kodenya:

```
app.set('view engine', 'ejs');
```

      3. **Middleware**
         - Menggunakan body-parser untuk mem-parsing data dari form.
         - Menggunakan session untuk menyimpan status autentikasi pengguna.
         - Menyajikan file statis dari direktori public.

         Kodenya:

```
app.set('view engine', 'ejs');
app.use(bodyParser.urlencoded({ extended: true }));
app.use(session({
    secret: 'secret-key',
    resave: false,
    saveUninitialized: true
}));
app.use(express.static('public'));
```

b. **Rute sistem**

1. **Rute untuk autentikasi**
    - Menggunakan rute dari **auth.js** untuk mengatur proses login.

    Kodenya:

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

2. **Rute untuk data kamar**
    - Menggunakan rute dari **kamar.js** untuk mengatur operasi terkait kamar.

    Kodenya:

```
app.use('/kamar', kamarRoutes);
```

3. **Halaman utama**
    - Rute default yang merender halaman utama **index.ejs**.

    Kodenya:

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

4. **Menjalankan server**
    - Menjalankan server pada port yang ditentukan.

    Kodenya:

```
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

c. **Proses rute autentikasi (auth.js)**

1. **Halaman login**
    - Mengatur rute untuk menampilkan form login

    Kodenya:

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

2. **Proses login**
    - Mengambil email dan password dari form, lalu memverifikasi data ke database.
    - Jika berhasil, menyimpan status autentikasi dalam session dan mengalihkan ke halaman kamar.
    - Jika gagal, menampilkan pesan kesalahan.

    Kodenya:

```
router.post('/login', (req, res) => {
    const { email, password } = req.body;
    db.query('SELECT * FROM admins WHERE email = ? AND password = ?', [email, password], (err, results) => {
        if (err) throw err;
        if (results.length > 0) {
            req.session.isAuthenticated = true;
            res.redirect('/kamar');
        } else {
            res.render('login', { error: 'Email atau password salah' });
        }
    });
});
```

d. Rute data kamar (kamar.js)

1. **Middleware untuk autentikasi**
    - Memeriksa apakah pengguna sudah terautentikasi sebelum mengakses rute kamar.

    Kodenya:

```
router.use((req, res, next) => {
    if (!req.session.isAuthenticated) {
        return res.redirect('/auth/login');
    }
    next();
});
```

2. **Menampilkan data kamar**
    - Mengambil data kamar dari database dan merender halaman dengan daftar kamar.

    Kodenya:

```
router.get('/', (req, res) => {
    db.query('SELECT * FROM kamar', (err, results) => {
        if (err) throw err;
        res.render('kamar', { kamars: results });
    });
});
```

3. **Menambah data kamar**
    - Mengambil data dari form dan menyimpannya ke database.

    Kodenya:

```
router.post('/', (req, res) => {
    const { jumlah_kamar, tipe_kamar, harga, check_in, check_out } = req.body;
    db.query('INSERT INTO kamar (jumlah_kamar, tipe_kamar, harga, check_in, check_out) VALUES (?, ?, ?, ?, ?)
    ',
    [jumlah_kamar, tipe_kamar, harga, check_in, check_out], (err) => {
        if (err) throw err;
        res.redirect('/kamar');
    });
});
```

4. **Menghapus data kamar**
    - Menghapus data kamar berdasarkan ID yang diterima dari URL.

    Kodenya:

```
router.post('/delete/:id', (req, res) => {
    db.query('DELETE FROM kamar WHERE id = ?', [req.params.id], (err) => {
        if (err) throw err;
        res.redirect('/kamar');
    });
});
```

5. **Mengedit kamar**
    - Mengambil data kamar berdasarkan ID, menampilkannya dalam form untuk diedit.
    - Setelah form disubmit, memperbarui data di database.

    Kodenya:

```
router.get('/edit/:id', (req, res) => {
    db.query('SELECT * FROM kamar WHERE id = ?', [req.params.id], (err, results) => {
        if (err) throw err;
        res.render('editKamar', { kamar: results[0] });
    });
});

router.post('/edit/:id', (req, res) => {
    const { jumlah_kamar, tipe_kamar, harga, check_in, check_out } = req.body;
    db.query('UPDATE kamar SET jumlah_kamar = ?, tipe_kamar = ?, harga = ?, check_in = ?, check_out = ? WHERE
    id = ?',
    [jumlah_kamar, tipe_kamar, harga, check_in, check_out, req.params.id], (err) => {
        if (err) throw err;
        res.redirect('/kamar');
    });
});
```

e. **View ejs**

1. **Halaman Utama (index.ejs):**
    - Menyediakan tautan untuk login.
2. **Halaman Login (login.ejs):**
    - Form untuk input email dan password. Menampilkan pesan kesalahan jika login gagal.
3. **Halaman Data Kamar (kamar.ejs):**
    - Form untuk menambah kamar dengan berbagai input. Menampilkan tabel yang berisi semua kamar yang sudah ditambahkan. Setiap baris memiliki opsi untuk mengedit atau menghapus kamar.
4. **Halaman Edit Kamar (editKamar.ejs):**
    - Form yang terisi dengan data kamar yang ingin diedit. Setelah diubah, data akan diperbarui di database.