UJIAN TENGAH SEMESTER

Mata Kuliah Pemrograman WEB

PERTEMUAN 8

UJIAN TENGAH SEMESTER

Dosen Pengampu:
Wildan Aprizall Arifin, S.Pd., M.Kom.

Disusun Oleh:
Nelita Maharani
2307052

PROGRAM STUDI SISTEM INFORMASI KELAUTAN

UNIVERSITAS PENDIDIKAN INDONESIA

2024

I. PENDAHULUAN

Pada zaman yang serba digital ini, pengembangan sebuah aplikasi manajemen acara berbasis relawan bernama OceanAware, yang dirancang untuk membantu organisasi dalam mengelola data relawan, jadwal acara, lokasi kegiatan, serta data pengguna. Aplikasi ini dibangun menggunakan teknologi Node.js sebagai server, Express sebagai framework backend, MySQL sebagai sistem manajemen basis data, dan EJS (Embedded JavaScript) sebagai template engine untuk menghasilkan tampilan halaman web yang dinamis. Dengan pendekatan full-stack ini, aplikasi dapat menjalankan operasi CRUD (Create, Read, Update, Delete) pada berbagai jenis data, memberikan fleksibilitas dan kemudahan bagi admin atau pengguna dalam mengelola informasi acara.

Aplikasi OceanAware memiliki beberapa fitur utama, di antaranya adalah autentikasi pengguna untuk memastikan hanya pengguna terotorisasi yang dapat mengakses data, serta sesi login yang dikelola menggunakan express-session. Sistem ini memungkinkan pengguna untuk menambah relawan baru, menetapkan jadwal acara, mengelola lokasi kegiatan, serta memperbarui atau menghapus data yang sudah ada, semuanya melalui antarmuka web yang intuitif. Setiap bagian dari aplikasi ini dirancang dengan pertimbangan keterbacaan dan konsistensi tampilan menggunakan EJS dan CSS pada file statis yang disimpan di folder public.

Aplikasi ini dibagi menjadi beberapa modul untuk memastikan efisiensi dan kemudahan dalam pengembangan serta pemeliharaan. Misalnya, koneksi ke database MySQL dipisahkan dalam file database.js, sementara app.js digunakan sebagai pusat konfigurasi server dan rute. Dengan struktur dan arsitektur ini, OceanAware tidak hanya berfungsi sebagai aplikasi manajemen data relawan yang dapat diakses dengan mudah, tetapi juga mudah dikembangkan lebih lanjut sesuai kebutuhan organisasi yang menggunakan sistem manajemen acara relawan.

II. ALAT DAN BAHAN

1. Komputer/Laptop
2. Visual Studio Code
3. XAMPP
4. MySQL
5. Chrome

III. LANGKAH KERJA

1. App.js

```
const express = require('express');
const session = require('express-session');
const bodyParser = require('body-parser');
const db = require('./database');

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

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

app.get('/login', (req, res) => {
    res.render('login', { title: 'Login' });
});

app.post('/login', (req, res) => {
    const { username, password } = req.body;
    db.query('SELECT * FROM users WHERE username = ? AND password = ?', [username, password], (err, results) => {
        if (results.length > 0) {
            req.session.user = results[0];
            res.redirect('/users');
        } else {
            res.send('Username atau password salah!');
        }
    });
});

app.get('/users', (req, res) => {
    if (!req.session.user) {
        return res.redirect('/login');
    }
```

```
37.
38.     db.query('SELECT * FROM users', (err, users) => {
39.         if (err) throw err;
40.         res.render('users', { title: 'Manage Users', users });
41.     });
42.});
43.

44.app.post('/addUser', (req, res) => {
45.     const { username, password } = req.body;
46.     db.query('INSERT INTO users (username, password) VALUES (?, ?)',
   [username, password], (err) => {
47.         if (err) throw err;
48.         res.redirect('/users');
49.     });
50.});
51.

52.app.get('/editUser/:id', (req, res) => {
53.     const userId = req.params.id;
54.     db.query('SELECT * FROM users WHERE id = ?', [userId], (err,
   results) => {
55.         if (err) throw err;
56.         res.render('editUser', { title: 'Edit User', user: results[0]
   });
57.     });
58.});
59.

60.app.post('/editUser/:id', (req, res) => {
61.     const userId = req.params.id;
62.     const { username, password } = req.body;
63.     db.query('UPDATE users SET username = ?, password = ? WHERE id = ?',
   [username, password, userId], (err) => {
64.         if (err) throw err;
65.         res.redirect('/users');
66.     });
67.});
68.

69.app.post('/deleteUser/:id', (req, res) => {
70.     const userId = req.params.id;
71.     db.query('DELETE FROM users WHERE id = ?', [userId], (err) => {
72.         if (err) throw err;
```

```
        res.redirect('/users');
    });
});

app.get('/volunteers', (req, res) => {
    db.query('SELECT * FROM volunteers', (err, volunteers) => {
        if (err) throw err;
        res.render('volunteers', { title: 'Manage Volunteers', volunteers });
    });
});

app.post('/addVolunteer', (req, res) => {
    const { name, contact } = req.body;
    db.query('INSERT INTO volunteers (name, contact) VALUES (?, ?)', [name, contact], (err) => {
        if (err) throw err;
        res.redirect('/volunteers');
    });
});

app.get('/editVolunteer/:id', (req, res) => {
    const volunteerId = req.params.id;
    db.query('SELECT * FROM volunteers WHERE id = ?', [volunteerId], (err, results) => {
        if (err) throw err;
        res.render('editVolunteer', { title: 'Edit Volunteer', volunteer: results[0] });
    });
});

                                app.post('/editVolunteer/:id', (req, res) => {
                                    const volunteerId = req.params.id;
                                    const { name, contact } = req.body;
                                    db.query('UPDATE volunteers SET name = ?, contact = ? WHERE id = ?', [name, contact, volunteerId], (err) => {
                                        if (err) throw err;
                                        res.redirect('/volunteers');
```

```
                                    });
                                });

                                app.post('/deleteVolunteer/:id', (req, res) => {
                                    const volunteerId = req.params.id;
                                    db.query('DELETE FROM volunteers WHERE id = ?', [volunteerId], (err) => {
                                        if (err) throw err;
                                        res.redirect('/volunteers');
                                    });
                                });

                                app.get('/eventDates', (req, res) => {
                                    db.query('SELECT * FROM event_dates', (err, eventDates) => {
                                        if (err) throw err;
                                        res.render('eventDates', { title: 'Manage Event Dates', eventDates });
                                    });
                                });

                                app.post('/addEventDate', (req, res) => {
                                    const { date } = req.body;
                                    db.query('INSERT INTO event_dates (date) VALUES (?)', [date], (err) => {
                                        if (err) throw err;
                                        res.redirect('/eventDates');
                                    });
                                });

                                app.get('/editEventDate/:id', (req, res) => {
                                    const eventDateId = req.params.id;
                                    db.query('SELECT * FROM event_dates WHERE id = ?', [eventDateId], (err, results) => {
                                        if (err) throw err;
                                        res.render('editEventDate', { title: 'Edit Event Date', eventDate: results[0] });
```

```
137.                                     });
138.                                 });
139.
140.                                 app.post('/editEventDate/:id', (req,
    res) => {
141.                                     const eventDateId = req.params.id;
142.                                     const { date } = req.body;
143.                                     db.query('UPDATE event_dates SET
    date = ? WHERE id = ?', [date, eventDateId], (err) => {
144.                                         if (err) throw err;
145.                                         res.redirect('/eventDates');
146.                                     });
147.                                 });
148.
149.                                 app.post('/deleteEventDate/:id', (req,
    res) => {
150.                                     const eventDateId = req.params.id;
151.                                     db.query('DELETE FROM event_dates
    WHERE id = ?', [eventDateId], (err) => {
152.                                         if (err) throw err;
153.                                         res.redirect('/eventDates');
154.                                     });
155.                                 });
156.
157.                                 app.get('/locations', (req, res) => {
158.                                     db.query('SELECT * FROM locations',
    (err, locations) => {
159.                                         if (err) throw err;
160.                                         res.render('locations', {
    title: 'Manage Locations', locations });
161.                                     });
162.                                 });
163.
164.                                 app.post('/addLocation', (req, res) =>
    {
165.                                     const { address } = req.body;
166.                                     db.query('INSERT INTO locations
    (address) VALUES (?)', [address], (err) => {
167.                                         if (err) throw err;
168.                                         res.redirect('/locations');
```

```
                                    });
                                });

                                app.get('/editLocation/:id', (req, res)
    => {
                                    const locationId = req.params.id;
                                    db.query('SELECT * FROM locations
    WHERE id = ?', [locationId], (err, results) => {
                                        if (err) throw err;
                                        res.render('editLocation', {
    title: 'Edit Location', location: results[0] });
                                    });
                                });

                                app.post('/editLocation/:id', (req,
    res) => {
                                    const locationId = req.params.id;
                                    const { address } = req.body;
                                    db.query('UPDATE locations SET
    address = ? WHERE id = ?', [address, locationId], (err) => {
                                        if (err) throw err;
                                        res.redirect('/locations');
                                    });
                                });

                                app.post('/deleteLocation/:id', (req,
    res) => {
                                    const locationId = req.params.id;
                                    db.query('DELETE FROM locations
    WHERE id = ?', [locationId], (err) => {
                                        if (err) throw err;
                                        res.redirect('/locations');
                                    });
                                });

                                const PORT = process.env.PORT || 3000;
                                app.listen(PORT, () => {
                                    console.log(`Server running on
    http://localhost:${PORT}`);
                                });
```

```
201.
```

Kode program ini merupakan aplikasi manajemen berbasis web untuk OceanAware yang memungkinkan pengelolaan pengguna, relawan, tanggal acara, dan lokasi kegiatan menggunakan teknologi Node.js, Express, MySQL, dan EJS. Aplikasi dimulai dengan mengimpor modul-modul yang diperlukan: `express` untuk membangun server, `express-session` untuk mengelola sesi pengguna, `body-parser` untuk menguraikan data formulir, serta `database.js` untuk koneksi ke database MySQL. Dengan `express`, aplikasi membuat beberapa rute yang memungkinkan pengguna melakukan operasi CRUD (Create, Read, Update, Delete) pada data pengguna, relawan, tanggal acara, dan lokasi kegiatan. Aplikasi menggunakan sesi untuk autentikasi, memastikan hanya pengguna yang sudah login yang dapat mengakses dan mengelola data. Rute login (`/login`) menerima `username` dan `password` yang kemudian divalidasi terhadap data di tabel `users` pada database. Jika valid, sesi pengguna diatur; jika tidak, aplikasi memberikan pesan kesalahan. Setelah login, pengguna dapat mengakses halaman-halaman CRUD, termasuk `/users` untuk data pengguna, `/volunteers` untuk data relawan, `/eventDates` untuk tanggal acara, dan `/locations` untuk lokasi kegiatan. Setiap halaman ini menggunakan metode HTTP `GET` untuk mengambil dan menampilkan data dari database, `POST` untuk menambahkan data baru, dan metode lainnya seperti `POST` dengan parameter untuk memperbarui dan menghapus entri berdasarkan `id`. Tampilan dibuat menggunakan EJS sebagai template engine, memungkinkan data ditampilkan secara dinamis di setiap halaman. Dalam setiap rute, data dari database dikirim ke file `.ejs` yang sesuai, misalnya `users.ejs` untuk data pengguna atau `volunteers.ejs` untuk data relawan. File ini dirender dengan data yang diperoleh dari database, memudahkan pengguna untuk melihat, mengedit, menambah, atau menghapus data langsung dari halaman. Akhirnya, aplikasi mendengarkan di port tertentu, memungkinkan akses melalui `http://localhost:3000`. Kode ini menyediakan sistem manajemen yang terstruktur dan responsif untuk mengelola data OceanAware secara efisien.

2. Database.js

```
3. const mysql = require('mysql');
4.
5. const db = mysql.createConnection({
6.     host: 'localhost',
7.     user: 'SIK',
8.     password: '',
9.     database: 'oceanaware'
10.});
11.
12.db.connect((err) => {
13.    if (err) throw err;
14.    console.log('Connected to database.');
15.});
16.
17.module.exports = db;
18.
```

Kode di atas bertujuan untuk mengatur koneksi ke database MySQL yang digunakan dalam aplikasi OceanAware. Pertama, modul `mysql` diimpor untuk memungkinkan interaksi dengan database. Kemudian, objek koneksi dibuat dengan menggunakan `mysql.createConnection`, yang mengharuskan beberapa parameter penting: `host`, `user`, `password`, dan `database`. Dalam hal ini, `host` diset ke `'localhost'`, yang berarti database dijalankan di mesin lokal, `user` diisi dengan nama pengguna `'SIK'`, `password` dibiarkan kosong (yang mungkin perlu disesuaikan berdasarkan pengaturan pengguna di MySQL), dan `database` diatur ke `'oceanaware'`, yang merupakan nama database yang akan digunakan oleh aplikasi. Selanjutnya, kode menginisialisasi koneksi dengan memanggil metode `db.connect`, yang akan mencoba menghubungkan aplikasi ke database. Jika koneksi gagal, fungsi callback menerima parameter `err` yang akan menampilkan kesalahan jika ada. Jika berhasil, pesan "Connected to database." dicetak di konsol, yang menandakan bahwa aplikasi berhasil terhubung ke database. Akhirnya, objek koneksi `db` diekspor menggunakan `module.exports`, sehingga dapat digunakan dalam file lain dalam aplikasi, seperti file utama `app.js`, untuk melakukan query dan mengelola data di database OceanAware. Dengan cara ini, kode ini menyediakan fungsionalitas dasar untuk berkomunikasi dengan database MySQL.

3. Main.ejs

```
4. <h1>WELCOME TO OCEANAWARE</h1>
5.     <nav>
6.         <ul>
7.             <li><a href="/">Home</a></li>
8.             <li><a href="/users">Users</a></li>
9.             <li><a href="/volunteers">Volunteers</a></li>
10.            <li><a href="/eventDates">Event Dates</a></li>
11.            <li><a href="/locations">Locations</a></li>
12.            <li><a href="/login">Login</a></li>
13.        </ul>
14.    </nav>
15.    <div class="container">
16.
17.    </div>
18.    <footer>
19.        &copy; 2024 OceanAware. All rights reserved.
20.    </footer>
21.</body>
22.</html>
23.
```

Kode di atas merupakan potongan HTML yang mendefinisikan struktur dasar dari halaman web untuk aplikasi OceanAware. Di bagian atas, terdapat elemen `<h1>` yang menampilkan teks "WELCOME TO OCEANAWARE", memberikan sambutan kepada pengguna saat mereka mengunjungi halaman utama aplikasi.

24. Index.ejs

```
  <h1>Selamat Datang di OceanAware</h1>
    <p>Website untuk manajemen relawan dan kegiatan.</p>

    <nav>
        <ul>
            <li><a href="/">Home</a></li>
            <li><a href="/users">Users</a></li>
            <li><a href="/volunteers">Volunteers</a></li>
            <li><a href="/programs">Programs</a></li>
            <li><a href="/login">Login</a></li>
        </ul>
    </nav>


    <div class="container">
        <h2>Login</h2>
        <form action="/login" method="POST">
            <div class="form-group">
                <label for="username">Username:</label>
                <input type="text" id="username" name="username"
   required>
            </div>
            <div class="form-group">
                <label for="password">Password:</label>
                <input type="password" id="password" name="password"
   required>
            </div>
            <button type="submit" class="btn btn-primary">Login</button>
        </form>
    </div>


    <div class="container">
        <h2>Manage Users</h2>
        <form action="/addUser" method="POST">
            <div class="form-group">
                <label for="username">Username:</label>
                <input type="text" id="username" name="username"
   required>
            </div>
            <div class="form-group">
                <label for="password">Password:</label>
                <input type="password" id="password" name="password"
   required>
```

```
            </div>
            <button type="submit" class="btn btn-success">Add User</button>
        </form>
    </div>

    <!-- Manage Volunteers Section -->
    <div class="container">
        <h2>Manage Volunteers</h2>
        <form action="/addVolunteer" method="POST">
            <input type="text" name="name" placeholder="Name" required>
            <input type="text" name="contact" placeholder="Contact" required>
            <button type="submit" class="btn btn-success">Add Volunteer</button>
        </form>
    </div>

    <!-- Manage Programs Section -->
    <div class="container">
        <h2>Manage Programs</h2>
        <form action="/addProgram" method="POST">
            <input type="text" name="programName" placeholder="Program Name" required>
            <input type="text" name="description" placeholder="Description" required>
            <button type="submit" class="btn btn-success">Add Program</button>
        </form>
        <table>
            <tr>
                <th>ID</th>
                <th>Program Name</th>
                <th>Description</th>
                <th>Action</th>
            </tr>
            <tr>
                <td>program.id</td>
                <td>program.programName</td>
                <td>program.description</td>
                <td>
                        <a href="/editProgram/<%= program.id %>" class="btn btn-warning">Edit</a>
```

```
                        <form action="/deleteProgram/<%= program.id
   %>" method="POST" style="display:inline;">
                            <button type="submit" class="btn btn-
   danger">Delete</button>
                        </form>
                    </td>
                </tr>
            </table>
        </div>
    </body>
    </html>

```

Program di atas adalah potongan kode HTML yang merupakan bagian dari aplikasi OceanAware, dirancang untuk manajemen relawan, pengguna, dan program acara. Kode ini menyusun antarmuka pengguna yang intuitif, memungkinkan pengguna untuk melakukan beberapa tindakan seperti login, menambah pengguna baru, serta mengelola relawan dan program. Pada bagian atas, terdapat elemen `<h1>` yang menyambut pengguna dengan teks "Selamat Datang di OceanAware" diikuti dengan paragraf deskriptif tentang tujuan situs. Navigasi utama yang terletak dalam elemen `<nav>` berisi tautan ke berbagai bagian situs, termasuk halaman utama, halaman pengguna, relawan, program, dan halaman login.

4. Login.ejs

```
<div class="container">
    <h2>Login</h2>
    <form action="/login" method="POST">
        <div class="form-group">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" required>
        </div>
        <div class="form-group">
            <label for="password">Password:</label>
            <input type="password" id="password" name="password"
   required>
        </div>
        <button type="submit" class="btn btn-primary">Login</button>
    </form>
</div>

```

Kode di atas merupakan bagian dari formulir HTML yang dirancang untuk fungsi login dalam aplikasi OceanAware. Dimulai dengan elemen `<div>` bertipe "container" yang berfungsi sebagai wadah untuk konten formulir, kode ini mengorganisir tampilan visual

secara efisien. Di dalamnya terdapat elemen `<h2>` yang menampilkan judul "Login", menandakan kepada pengguna bahwa bagian ini khusus untuk proses masuk ke aplikasi. Formulir yang didefinisikan oleh elemen `<form>` mengarahkan data ke rute `/login` di server dengan metode `POST`, yang menjaga keamanan data sensitif seperti kata sandi. Terdapat dua grup formulir: yang pertama untuk menangkap `username`, lengkap dengan label dan elemen input tipe teks yang bersifat wajib diisi, dan yang kedua untuk `password`, menggunakan input tipe password yang menyembunyikan karakter saat diketik untuk perlindungan tambahan. Di bagian bawah formulir, terdapat tombol `<button>` yang berfungsi sebagai pengirim formulir dengan teks "Login", dan menggunakan kelas "btn btn-primary" untuk memberikan gaya yang menarik. Secara keseluruhan, kode ini menciptakan antarmuka pengguna yang intuitif dan terstruktur, memungkinkan pengguna untuk dengan mudah memasukkan kredensial mereka untuk mengakses aplikasi.

5. Volunteer.ejs

```
<h2>Manage Volunteers</h2>
<table>
    <tr>
        <th>ID</th>
        <th>Name</th>
        <th>Contact</th>
        <th>Action</th>
    </tr>
    <tr>
        <td>= volunteer.id</td>
        <td>= volunteer.name </td>
        <td>= volunteer.contact </td>
        <td>
            <a href="/editVolunteer/<%= volunteer.id %>">Edit</a>
            <form action="/deleteVolunteer/<%= volunteer.id %>" method="POST" style="display:inline;">
                <button type="submit">Delete</button>
            </form>
        </td>
    </tr>

</table>
<form action="/addVolunteer" method="POST">
    <input type="text" name="name" placeholder="Name" required>
    <input type="text" name="contact" placeholder="Contact" required>
    <button type="submit">Add Volunteer</button>
</form>

```

Kode di atas adalah bagian dari antarmuka pengguna untuk manajemen relawan dalam aplikasi OceanAware. Diawali dengan elemen `<h2>` yang menampilkan judul "Manage Volunteers", bagian ini memberikan konteks tentang fungsi halaman. Selanjutnya, terdapat tabel yang diwakili oleh elemen `<table>`, yang berfungsi untuk menampilkan daftar relawan yang terdaftar. Tabel dimulai dengan baris judul (`<tr>`) yang berisi empat kolom: ID, Name, Contact, dan Action. Kolom "Action" menyediakan opsi bagi pengguna untuk mengedit atau menghapus data relawan. Dalam baris berikutnya (`<tr>`), placeholder dengan sintaks `<%= volunteer.id %>`, `<%= volunteer.name %>`, dan `<%= volunteer.contact %>` menunjukkan di mana informasi relawan dari database akan diisi menggunakan template engine EJS, memungkinkan penyajian data dinamis. Di dalam kolom "Action", terdapat tautan untuk mengedit relawan yang diarahkan ke rute `/editVolunteer/<%= volunteer.id %>`, yang akan memunculkan halaman untuk mengubah informasi relawan tersebut. Selain itu, terdapat formulir (`<form>`) untuk menghapus relawan, yang akan mengirimkan permintaan `POST` ke rute `/deleteVolunteer/<%= volunteer.id %>`. Tombol `Delete` berada di dalam formulir yang disembunyikan secara visual dengan atribut `style="display:inline;"`, memastikan bahwa tindakan penghapusan dapat dilakukan tanpa membuat baris baru. Di bawah tabel, terdapat formulir tambahan untuk menambah relawan baru, yang meminta input nama dan kontak. Setelah diisi, data akan dikirim ke rute `/addVolunteer` dengan metode `POST`, memungkinkan pengguna untuk memperluas daftar relawan dengan mudah. Dengan cara ini, kode ini menyediakan fungsi CRUD (Create, Read, Update, Delete) yang esensial untuk mengelola relawan dalam aplikasi dengan tampilan yang terstruktur dan fungsional.

6. Users.ejs

```
<h2>Manage Users</h2>
<form action="/addUser" method="POST">
    <div class="form-group">
        <label for="username">Username:</label>
        <input type="text" class="form-control" id="username" name="username" required>
    </div>
    <div class="form-group">
        <label for="password">Password:</label>
        <input type="password" class="form-control" id="password" name="password" required>
    </div>
    <button type="submit" class="btn btn-success">Add User</button>
</form>
<table class="table mt-3">
    <thead>
        <tr>
```

```
            <th>ID</th>
            <th>Username</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>

            <tr>
                <td> user.id </td>
                <td> user.username </td>
                <td>
                    <a href="/editUser/<%= user.id %>" class="btn btn-warning">Edit</a>
                    <form action="/deleteUser/<%= user.id %>" method="POST" style="display:inline;">
                        <button type="submit" class="btn btn-danger">Delete</button>
                    </form>
                </td>
            </tr>

    </tbody>
</table>

```

Kode di atas adalah bagian dari antarmuka pengguna untuk mengelola pengguna dalam aplikasi OceanAware. Dimulai dengan elemen `<h2>` yang menandakan bagian ini berfungsi untuk "Manage Users", memberikan pengguna konteks tentang apa yang bisa dilakukan di halaman ini. Selanjutnya, terdapat formulir yang menggunakan elemen `<form>` dengan atribut `action` yang mengarah ke `/addUser` dan metode `POST`, yang memungkinkan pengguna untuk menambahkan pengguna baru ke dalam sistem. Di dalam formulir, terdapat dua grup input: satu untuk `username` dan satu lagi untuk `password`, keduanya diwajibkan untuk diisi (dengan atribut `required`). Setiap input memiliki label yang jelas dan kelas CSS "form-control" untuk memberikan gaya yang konsisten dan responsif. Tombol submit di bagian bawah formulir bertuliskan "Add User" dan memiliki kelas "btn btn-success", yang memberikan tampilan yang menarik dan menonjol. Di bawah formulir, terdapat tabel yang berfungsi untuk menampilkan daftar pengguna yang telah terdaftar. Tabel ini diorganisir dengan elemen `<table>` dan menggunakan elemen `<thead>` untuk mendefinisikan kepala tabel dengan tiga kolom: ID, Username, dan Actions. Di dalam `<tbody>`, terdapat placeholder yang diwakili oleh `user.id` dan `user.username`, di mana informasi pengguna akan diisi menggunakan template engine EJS, memungkinkan data untuk ditampilkan secara dinamis. Di dalam kolom "Actions", terdapat dua pilihan: tautan untuk mengedit

pengguna yang mengarah ke rute `/editUser/<%= user.id %>` dengan tombol "Edit" yang diberi gaya kelas "btn btn-warning", serta formulir untuk menghapus pengguna dengan metode `POST` yang mengarah ke rute `/deleteUser/<%= user.id %>`, yang disertai dengan tombol "Delete" dengan gaya "btn btn-danger". Dengan demikian, kode ini menyajikan antarmuka yang terstruktur dan fungsional untuk melakukan operasi CRUD pada pengguna dengan cara yang intuitif dan menarik.

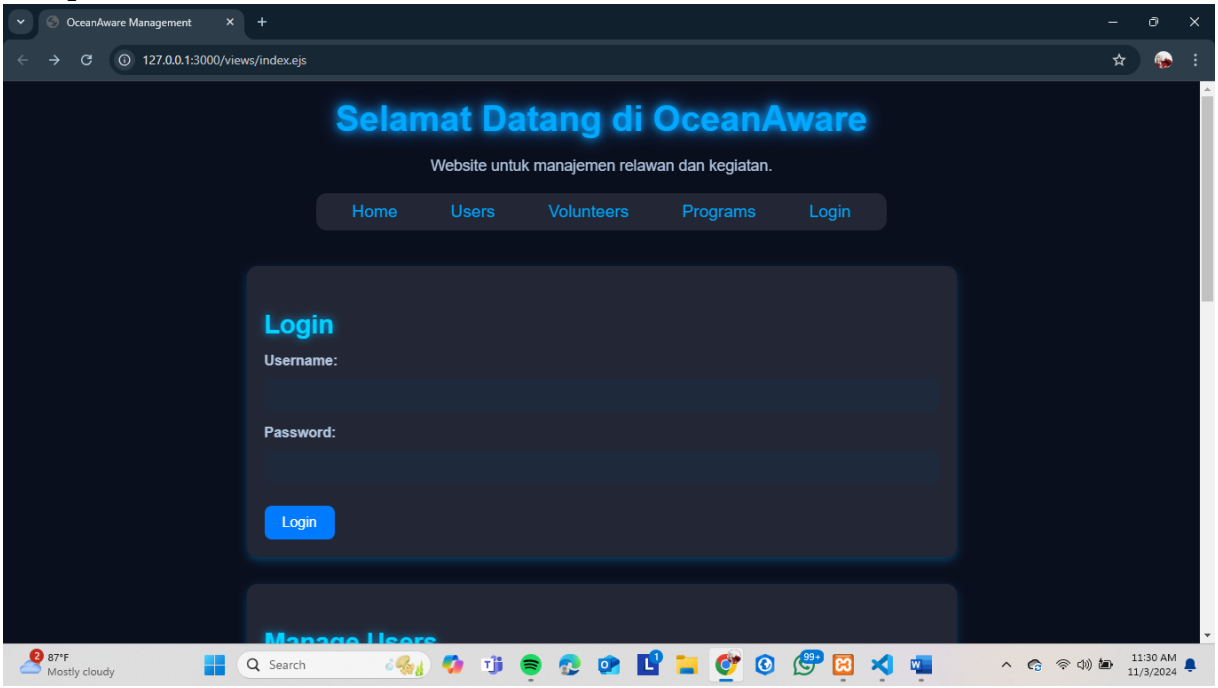
7. Database Mysql

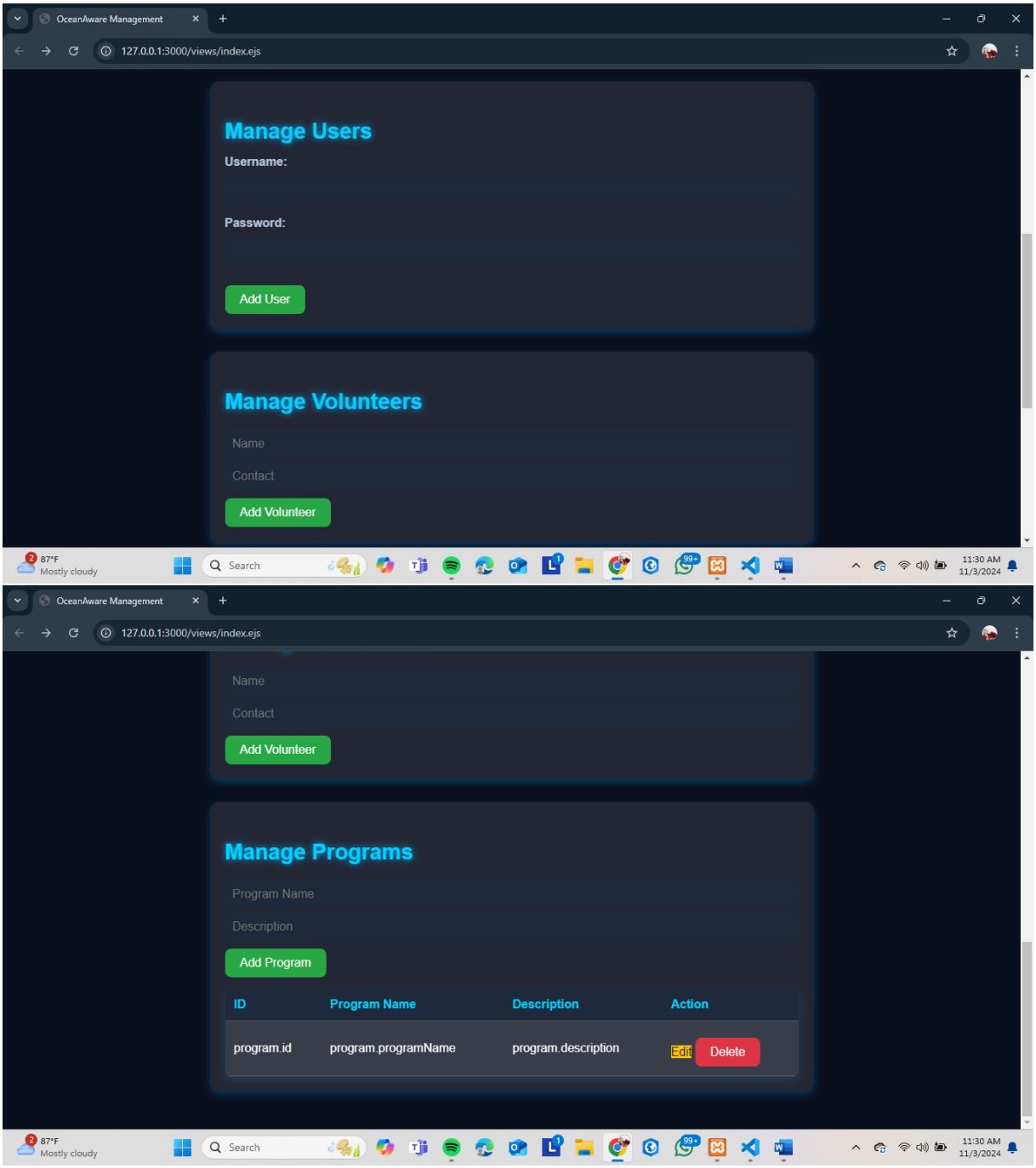
```
-- Active: 1727361777082@@127.0.0.1@3306@oceanaware

CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(255) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL
);

CREATE TABLE volunteers (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    contact VARCHAR(255) NOT NULL
);

CREATE TABLE event_dates (
    id INT AUTO_INCREMENT PRIMARY KEY,
    date DATE NOT NULL
);

CREATE TABLE locations (
    id INT AUTO_INCREMENT PRIMARY KEY,
    address VARCHAR(255) NOT NULL
);

```

Kode di atas adalah skrip SQL yang digunakan untuk membuat empat tabel utama dalam basis data OceanAware, masing-masing dengan fungsi spesifik yang mendukung fitur-fitur aplikasi. Pertama, tabel `users` dirancang untuk menyimpan informasi pengguna, dengan kolom `id` sebagai kunci utama yang otomatis bertambah untuk setiap entri baru. Kolom `username` dan `password` keduanya didefinisikan sebagai tipe `VARCHAR(255)`, di mana `username` harus unik dan tidak boleh kosong (NOT NULL), memastikan bahwa tidak ada dua pengguna yang memiliki nama pengguna yang sama. Selanjutnya, tabel `volunteers` berfungsi untuk menyimpan data relawan, dengan kolom `id` yang juga otomatis bertambah, dan dua kolom tambahan: `name` dan `contact`, yang keduanya wajib diisi. Tabel ini memungkinkan aplikasi untuk

melacak informasi kontak relawan yang berpartisipasi dalam berbagai kegiatan. Tabel `event_dates` dikhususkan untuk mencatat tanggal pelaksanaan acara, dengan kolom `id` sebagai kunci utama dan kolom `date` yang menyimpan tanggal acara tersebut, dijamin tidak boleh kosong. Terakhir, tabel `locations` menyimpan alamat tempat kegiatan berlangsung, di mana kolom `id` berfungsi sebagai kunci utama yang otomatis bertambah dan kolom `address` yang mencatat alamat dengan batasan bahwa isian tidak boleh kosong. Keempat tabel ini saling terkait dan menyediakan struktur data yang diperlukan untuk mendukung fitur-fitur manajemen relawan dan kegiatan dalam aplikasi OceanAware, sehingga memudahkan pengelolaan informasi yang penting bagi operasional sistem.

8. Tampilan WEB





IV. KESIMPULAN

Kesimpulan dari keseluruhan proyek OceanAware ini adalah bahwa aplikasi web yang dibangun menggunakan teknologi seperti Express.js, EJS, dan MySQL bertujuan untuk menyediakan platform yang efisien dan terorganisir untuk manajemen relawan dan kegiatan lingkungan. Dengan struktur yang jelas dan antarmuka pengguna yang intuitif, aplikasi ini memungkinkan pengguna untuk melakukan berbagai operasi dasar seperti menambah,

mengedit, dan menghapus data terkait pengguna, relawan, tanggal acara, dan lokasi kegiatan. Database yang dirancang dengan baik mendukung penyimpanan data yang diperlukan, memastikan integritas dan konsistensi informasi. Melalui implementasi CRUD (Create, Read, Update, Delete), aplikasi ini menawarkan fungsionalitas yang mendasar namun kuat untuk memenuhi kebutuhan manajemen kegiatan dan relawan. Selain itu, penggunaan session untuk manajemen login memastikan bahwa hanya pengguna yang terautentikasi yang dapat mengakses fitur tertentu, meningkatkan keamanan aplikasi. Secara keseluruhan, OceanAware merupakan solusi yang efektif untuk organisasi yang ingin mengelola kegiatan relawan dan pengelolaan lingkungan dengan lebih baik.

Link Github:

https://github.com/maharaninelita/UTS.git

Link Youtube:

https://youtu.be/FwDZl2SM0sg?si=iO2QaL_8HHHKf-2H