Laporan Praktikum

Mata Kuliah Pemrograman Web

Pertemuan UTS Tugas

ke 8 “Ujian Tengah

Semester”

Dosen Pengampu :

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

Disusun Oleh :

Muhammad Zikri Alhaq (2308504)

PROGRAM STUDI SISTEM INFORMASI KELAUTAN

UNIVERSITAS PENDIDIKAN INDONESIA

2024

Tujuan

Aplikasi Task Manager ini dirancang untuk memenuhi kebutuhan manajemen tugas personal dengan fokus pada kemudahan penggunaan, keamanan, dan efisiensi. Dengan fitur-fitur yang ada dan potensi pengembangan ke depan, aplikasi ini dapat menjadi solusi yang komprehensif untuk pengelolaan tugas sehari-hari.

Dasar Teori

Web session, atau sesi web adalah jumlah waktu yang dihabiskan oleh pengguna untuk menjelajahi website tertentu, mulai dari saat mereka memasuki halaman pertama, hingga mereka meninggalkan website tersebut.
Sesi web juga didefinisikan sebagai serangkaian tindakan yang dilakukan secara berurutan oleh pengunjung website dalam jangka waktu tertentu.
Pada umumnya, metrik ini juga mencakup kegiatan seperti: pencarian informasi, mengisi formulir untuk membaca konten, scrolling halaman website, menambahkan item ke keranjang belanja, mencari tiket transportasi secara online, dan lain sebagainya.
Setiap interaksi yang Anda lakukan terhadap suatu website dicatat sebagai sesi web ke dalam properti situs tersebut.

Deskripsi Aplikasi

Task Manager adalah aplikasi web untuk mengelola tugas/task dengan fitur autentikasi pengguna. Aplikasi ini dibangun menggunakan Node.js, Express, MySQL, dan EJS template engine.

Dokumentasi Kode Task Manager

1. Struktur File dan Kode

1.1 Server Setup (app.js)

```
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: 'your-secret-key',
    resave: false,
    saveUninitialized: true,
}));

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

// Gunakan rute auth
app.use('/auth', authRoutes);

// Root Route: Redirect ke landing page
app.get('/', (req, res) => {
```

```
    res.render('landing');
});

// Middleware untuk mengecek autentikasi
const checkAuth = (req, res, next) => {
    if (req.session.user) {
        next();
    } else {
        res.redirect('/auth/login');
    }
};

// Terapkan middleware untuk route yang membutuhkan autentikasi
app.use('/auth/dashboard', checkAuth);

// Menjalankan Server
app.listen(3000, () => {
    console.log('Server berjalan di http://localhost:3000');
});
```

1.2 Database Configuration (config/db.js)

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

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

db.connect((err) => {
    if (err) {
        console.error('Error connecting to database:', err);
        throw err;
    }
    console.log('Database connected!');
});

module.exports = db;
```

1.3 Authentication Routes (routes/auth.js)

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

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

router.post('/register', (req, res) => {
    const { username, email, password } = req.body;
    if (!username || !email || !password) {
        return res.send('Mohon isi semua kolom.');
    }
    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');
    });
});

// Login Routes
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;
                return res.redirect('/auth/dashboard');
            }
            return res.send('Password salah');
        }
        return res.send('Pengguna tidak ditemukan');
    });
});

// Task Management Routes
router.get('/dashboard', (req, res) => {
    if (req.session.user) {
        const query = "SELECT * FROM tasks WHERE user_id = ? ORDER BY deadline ASC";
        db.query(query, [req.session.user.id], (err, tasks) => {
            if (err) {
                console.error('Error fetching tasks:', err);
                return res.status(500).send('Error fetching tasks');
            }
            tasks = tasks.map(task => {
                if (task.deadline) {
                    task.deadline = new Date(task.deadline).toISOString().split('T')[0];
                }
                return task;
            });
            res.render('dashboard', {
                user: req.session.user,
                tasks: tasks
            });
        });
    } else {
        res.redirect('/auth/login');
    }
});

// Add, Edit, Delete Task Routes
router.post('/add-task', (req, res) => {
    if (req.session.user) {
        const { task_name, deadline } = req.body;
        const query = "INSERT INTO tasks (user_id, task_name, deadline) VALUES (?, ?, ?)";
        db.query(query, [req.session.user.id, task_name, deadline], (err, result) => {
            if (err) {
                console.error('Error adding task:', err);
                return res.status(500).send('Error adding task');
            }
            res.redirect('/auth/dashboard');
```

```
        });
    } else {
        res.redirect('/auth/login');
    }
});

router.post('/edit-task', (req, res) => {
    if (req.session.user) {
        const { task_id, task_name, deadline } = req.body;
        const query = "UPDATE tasks SET task_name = ?, deadline = ? WHERE id = ? AND user_id =
?";
        db.query(query, [task_name, deadline, task_id, req.session.user.id], (err, result) =>
{
            if (err) {
                console.error('Error updating task:', err);
                return res.status(500).send('Error updating task');
            }
            res.redirect('/auth/dashboard');
        });
    } else {
        res.redirect('/auth/login');
    }
});

router.post('/delete-task', (req, res) => {
    if (req.session.user) {
        const { task_id } = req.body;
        const query = "DELETE FROM tasks WHERE id = ? AND user_id = ?";
        db.query(query, [task_id, req.session.user.id], (err, result) => {
            if (err) throw err;
            res.redirect('/auth/dashboard');
        });
    } else {
        res.redirect('/auth/login');
    }
});

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

module.exports = router;
```

1.4 Views

Landing Page (views/landing.ejs)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Task Manager</title>
    <link rel="stylesheet" href="/styleslanding.css">
</head>
<body>
    <div class="container">
        <nav>
            <div class="logo">Task Manager</div>
            <div class="nav-links">
                <a href="/auth/login" class="nav-btn login-btn">Login</a>
                <a href="/auth/register" class="nav-btn register-btn">Register</a>
```

```
            </div>
        </nav>

        <main>
            <div class="hero">
                <h1>Kelola Tugas Anda dengan Mudah</h1>
                <p>Platform manajemen tugas yang membantu Anda tetap terorganisir dan
produktif</p>
                <div class="cta-buttons">
                    <a href="/auth/register" class="cta-btn">Mulai Sekarang</a>
                    <a href="/auth/login" class="cta-btn secondary">Masuk ke Akun</a>
                </div>
            </div>
            <div class="features">
                <div class="feature-card">
                    <h3>Buat Tugas</h3>
                    <p>Catat semua tugas Anda dengan mudah</p>
                </div>
                <div class="feature-card">
                    <h3>Atur Deadline</h3>
                    <p>Tetap tepat waktu dengan pengingat deadline</p>
                </div>
                <div class="feature-card">
                    <h3>Pantau Progress</h3>
                    <p>Lihat perkembangan tugas Anda</p>
                </div>
            </div>
        </main>
    </div>
</body>
</html>
```

Login Page (views/login.ejs)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login</title>
    <link rel="stylesheet" href="/styleslogin.css">
</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>
```

Dashboard Page (views/dashboard.ejs)

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dashboard</title>
    <link rel="stylesheet" href="/stylesdashboard.css">
</head>
<body>
    <div class="container">
        <h2>Selamat datang, <%= user.username %></h2>
        <p>Email: <%= user.email %></p>

        <h3>Daftar Tugas</h3>
        <% if (tasks.length > 0) { %>
          <ul>
            <% tasks.forEach(task => { %>
              <li>
                <div class="task-info">
                  <span><%= task.task_name %></span>
                  <% if(task.deadline) { %>
                    <span class="task-deadline">Deadline: <%= task.deadline %></span>
                  <% } %>
                </div>
                <div class="task-actions">
                  <button class="edit-btn" onclick="showEditForm('<%= task.id %>', '<%=
task.task_name %>', '<%= task.deadline %>')">Edit</button>
                  <form action="/auth/delete-task" method="POST" style="display: inline;">
                    <input type="hidden" name="task_id" value="<%= task.id %>">
                    <button type="submit" class="delete-btn">Hapus</button>
                  </form>
                </div>
              </li>
            <% }); %>
          </ul>
        <% } else { %>
          <p>Belum ada tugas.</p>
        <% } %>

        <div id="editFormContainer" class="edit-form-container" style="display: none;">
          <form id="editTaskForm" action="/auth/edit-task" method="POST">
            <input type="hidden" id="edit_task_id" name="task_id">
            <input type="text" id="edit_task_name" name="task_name" required>
            <input type="date" id="edit_deadline" name="deadline">
            <button type="submit">Simpan</button>
            <button type="button" onclick="hideEditForm()">Batal</button>
          </form>
        </div>

        <form action="/auth/add-task" method="POST" class="add-task-form">
            <input type="text" name="task_name" placeholder="Tambah tugas baru" required>
            <input type="date" name="deadline">
            <button type="submit">Tambah</button>
        </form>

        <form action="/auth/logout" method="GET">
            <button type="submit">Logout</button>
        </form>
    </div>

    <script>
    function showEditForm(taskId, taskName, deadline) {
      document.getElementById('editFormContainer').style.display = 'block';
      document.getElementById('edit_task_id').value = taskId;
```

```
      document.getElementById('edit_task_name').value = taskName;
      document.getElementById('edit_deadline').value = deadline;
    }

    function hideEditForm() {
      document.getElementById('editFormContainer').style.display = 'none';
    }
    </script>
</body>
</html>
```

2. Database Schema

```
-- Create Database
CREATE DATABASE user_management;
USE user_management;

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

-- Tasks Table
CREATE TABLE tasks (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    task_name VARCHAR(255) NOT NULL,
    deadline DATE,
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

3. Package Dependencies (package.json)

```
{
  "dependencies": {
    "bcryptjs": "^2.4.3",
    "body-parser": "^1.19.0",
    "ejs": "^3.1.6",
    "express": "^4.17.1",
    "express-session": "^1.17.2",
    "mysql": "^2.18.1"
  }
}
```

4. Instalasi dan Penggunaan

```
npm install express
```

5. Cara Penggunaan

- **Fitur Utama**
- • Autentikasi Pengguna
- • Register
- • Login
- • Logout
- • Manajemen Task
- • Tambah task baru
- • Edit task
- • Hapus task
- • Atur deadline task

- Dashboard Pengguna
- Tampilan daftar task
- Status task
- Informasi deadline

Cara Penggunaan

1. Register

- Buka halaman register
- Isi username, email, dan password
- Sistem akan memvalidasi input
- Jika berhasil, akan diarahkan ke halaman login

2. Login

- Masukkan email dan password
- Jika valid, akan diarahkan ke dashboard

3. Manajemen Task

- **Tambah Task**:
- Isi nama task pada form
- Atur deadline (opsional)
- Klik "Tambah"
- **Edit Task**:
- Klik tombol "Edit" pada task
- Ubah nama atau deadline
- Klik "Simpan"
- **Hapus Task**:
- Klik tombol "Hapus" pada task
- Konfirmasi penghapusan

4. Logout

- Klik tombol "Logout" untuk keluar

Keamanan

- Password di-hash menggunakan bcryptjs
- Menggunakan session untuk autentikasi
- Validasi input di sisi client dan server
- Proteksi route menggunakan middleware

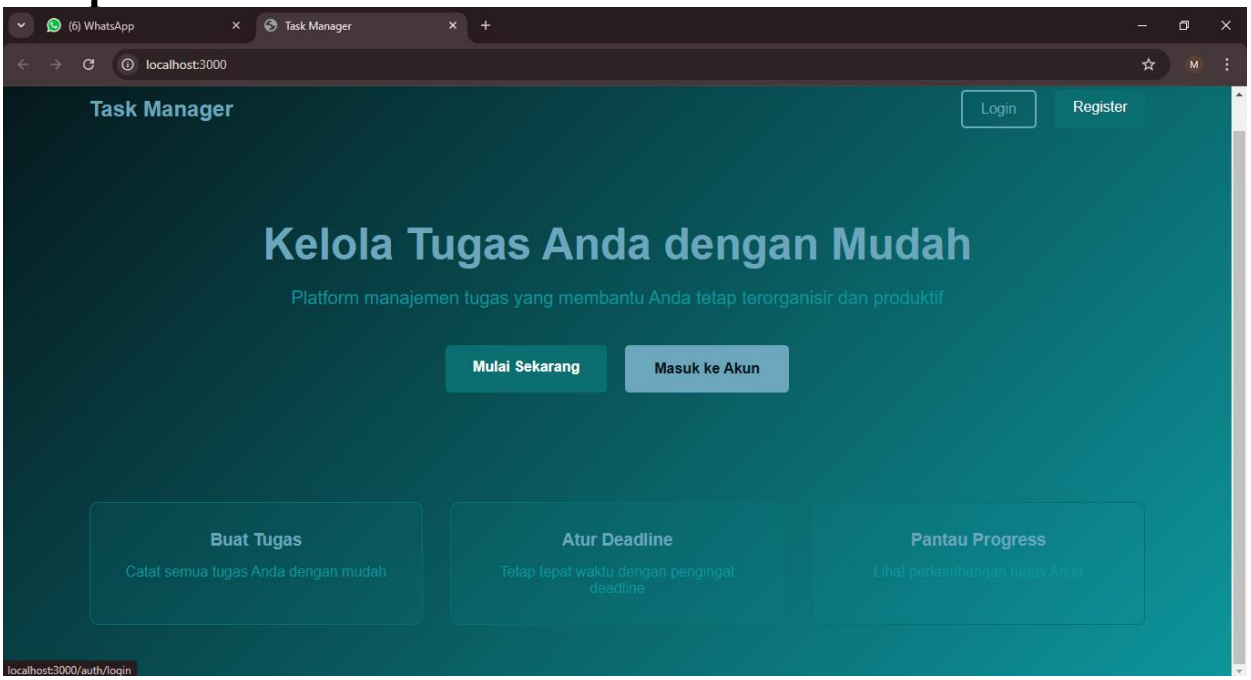
Struktur Tampilan

- **Landing Page** (`landing.ejs`)
- Halaman utama
- Menu login/register
- **Register** (`register.ejs`)
- Form pendaftaran
- Validasi input
- **Login** (`login.ejs`)
- Form login
- Pesan error
- **Dashboard** (`dashboard.ejs`)
- Daftar task
- Form tambah/edit task
- Menu logout

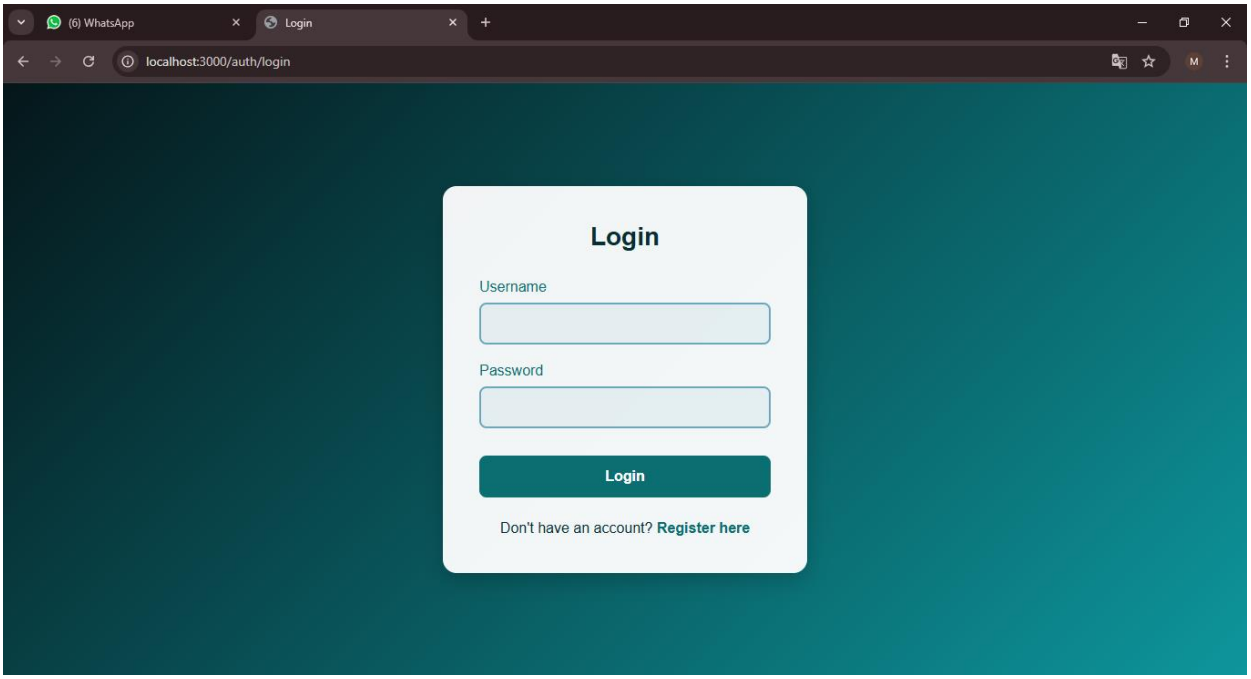
Struktur Folder

```
project/
├── config/
│   └── db.js
├── public/
│   ├── styleslanding.css
│   ├── styleslogin.css
│   ├── stylesregister.css
│   └── stylesdashboard.css
├── routes/
│   └── auth.js
├── views/
│   ├── landing.ejs
│   ├── login.ejs
│   ├── register.ejs
│   └── dashboard.ejs
├── app.js
└── package.json
```

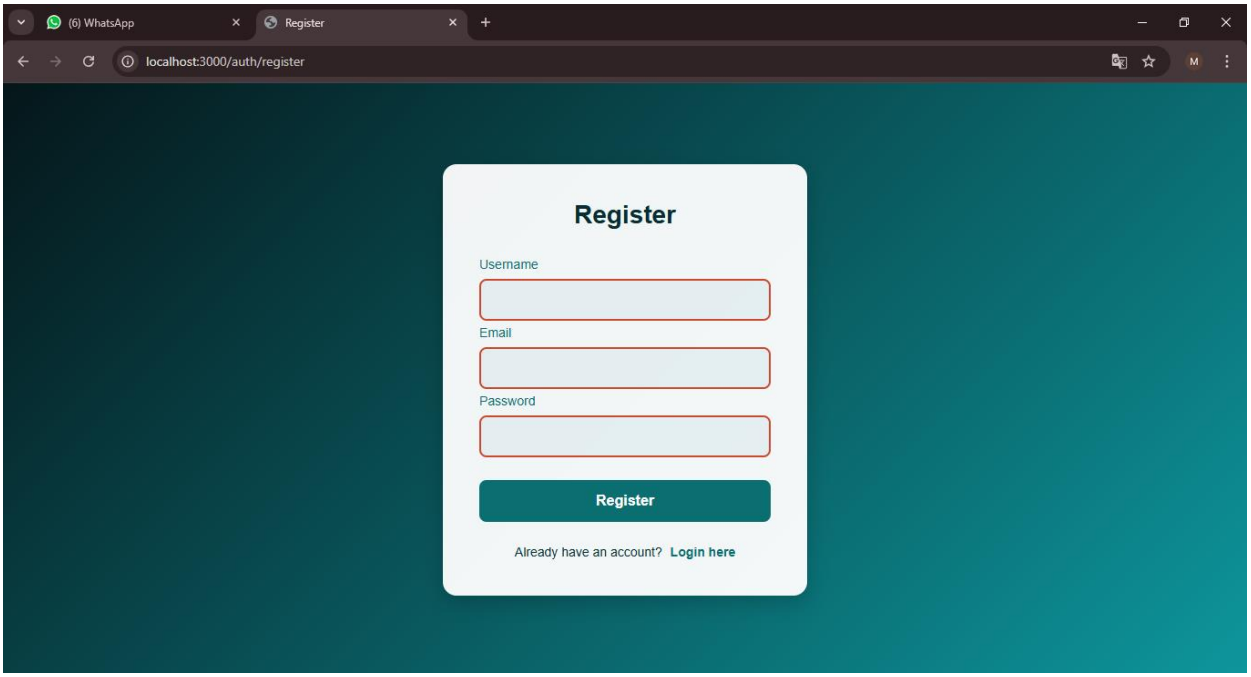
Tampilan Website



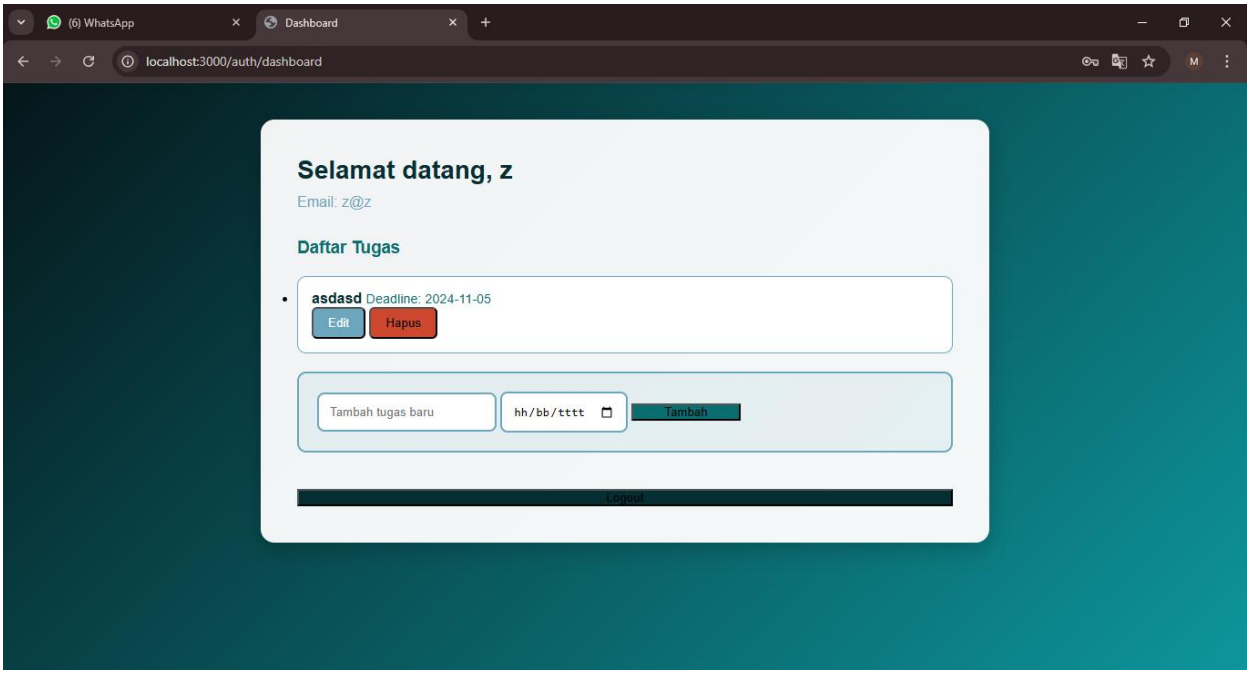
Gambar 1. Halaman Home Page.



Gambar 2. Halaman Login.



Gambar 3. Halaman Register



Gambar 4. Halaman Dashboard

Kesimpulan

Proyek ini mendemonstrasikan implementasi dasar sistem otentikasi dan manajemen pengguna, yang dapat menjadi fondasi untuk aplikasi web yang lebih kompleks. Dengan struktur yang modular dan penggunaan teknologi modern, proyek ini menyediakan kerangka kerja yang solid untuk pengembangan lebih lanjut, seperti penambahan fitur-fitur baru atau peningkatan keamanan.