Nama : Rian Rahman

NIM : 20240040018

Kelas : TI23E

Jawaban Tugas Pertemuan 16.

**Sistem Manajemen Tugas**

1. Pendahuluan

Sistem Manajemen Tugas adalah aplikasi yang memungkinkan pengguna untuk mengelola tugas mereka secara efektif. Sistem ini dapat digunakan oleh individu maupun tim untuk meningkatkan produktivitas dan keteraturan. Melalui REST API, sistem ini dapat berinteraksi dengan antarmuka front-end atau aplikasi lainnya.

2. Perancangan Database

2.1. Diagram ERD

Berikut adalah diagram Entity-Relationship (ERD) untuk basis data sistem ini:

Projects

Users

one-to-many

one-to-many

many-to-one

Tasks

one-to-many

Comments

2.2. Tabel Database

1. Users

- `id` (Primary Key, Integer, Auto Increment)

- `username` (String, Unique)

- `email` (String, Unique)

- `password` (String)

- `created\_at` (Timestamp)

2. Projects

- `id` (Primary Key, Integer, Auto Increment)

- `user\_id` (Foreign Key, Integer)

- `name` (String)

- `description` (Text)

- `created\_at` (Timestamp)

3. Tasks

- `id` (Primary Key, Integer, Auto Increment)

- `project\_id` (Foreign Key, Integer)

- `title` (String)

- `status` (String, e.g., "Pending", "In Progress", "Completed")

- `due\_date` (Date)

- `created\_at` (Timestamp)

4. Comments

- `id` (Primary Key, Integer, Auto Increment)

- `task\_id` (Foreign Key, Integer)

- `user\_id` (Foreign Key, Integer)

- `content` (Text)

- `created\_at` (Timestamp)

2.3. Relasi Tabel

- Users memiliki banyak Projects (one-to-many).

- Projects memiliki banyak Tasks (one-to-many).

- Tasks memiliki banyak Comments (one-to-many).

- Comments dihubungkan dengan Users (many-to-one).

3. Perancangan Endpoint

Berikut adalah daftar endpoint yang dirancang untuk API ini:

|  |  |  |
| --- | --- | --- |
| **Endpoint** | **HTTP Method** | **Deskripsi** |
| /users | GET | Mengambil semua pengguna |
| /users | POST | Menambahkan pengguna baru |
| /users/:id | GET | Mengambil detail pengguna |
| /users/:id | PUT | Memperbarui pengguna |
| /users/:id | DELETE | Menghapus pengguna |
| /projects | GET | Mengambil semua project |
| /projects | POST | Menambahkan project baru |
| /projects/:id | GET | Mengambil detail project |
| /projects/:id | PUT | Memperbarui project |
| /projects/:id | DELETE | Menghapus project |
| /tasks | GET | Mengambil semua tasks |
| /tasks | POST | Menambahkan task baru |
| /tasks/:id | GET | Mengambil detail task |
| /tasks/:id | PUT | Memperbarui task |
| /tasks/:id | DELETE | Menghapus task |
| /comments | GET | Mengambil semua komentar |
| /comments | POST | Menambahkan komentar baru |
| /comments/:id | GET | Mengambil detail komentar |
| /comments/:id | DELETE | Menghapus komentar |
| /login | POST | Login pengguna dan menghasilkan JWT |

4. Dokumentasi Backend API

Berikut adalah dokumentasi singkat untuk beberapa endpoint:

4.1. Endpoint: Menambahkan Pengguna Baru

- URL: `/users`

- Method: `POST`

- Request Body:

{

"username": "user123",

"email": "user@example.com",

"password": "securepassword"

}

- Response:

- 201 Created: Berhasil menambahkan pengguna.

- 400 Bad Request: Jika terjadi kesalahan validasi.

4.2. Endpoint: Mengambil Semua Proyek

- URL: `/projects`

- Method: `GET`

- Response:

- 200 OK: Mengembalikan daftar semua proyek dalam format JSON.

4.3. Endpoint: Login

- URL: `/login`

- Method: `POST`

- Request Body:

{

"email": "user@example.com",

"password": "securepassword"

}

- Response:

- 200 OK: Berhasil login dan mengembalikan token JWT.

- 401 Unauthorized: Jika kredensial salah.

5. Integrasi API Publik

Untuk menambah fungsionalitas, API kita bisa terintegrasi dengan API publik seperti API cuaca. Program ini dapat menggunakan data cuaca untuk pengingat tugas yang memiliki tenggat waktu berdasarkan cuaca. Misalnya, pengguna dapat menambahkan tugas luar ruangan yang kalau cuaca mendukung.

6. Middleware dan Validasi menggunakan JWT

API ini memanfaatkan JSON Web Tokens (JWT) untuk autentikasi pengguna. Middleware dijalankan pada setiap permintaan untuk memastikan akses ke endpoint dilindungi. Kode middleware:

// Middleware untuk memeriksa token JWT

const authenticateJWT = (req, res, next) => {

    const token = req.header('Authorization');

    if (!token) {

        return res.status(401).send('Access Denied');

    }

    try {

        const verified = jwt.verify(token, secretKey);

        req.user = verified;

        next();

    } catch (err) {

        res.status(400).send('Invalid Token');

    }

};

// Endpoint untuk login dan menghasilkan JWT

app.post('/login', (req, res) => {

    const { username, password } = req.body;

    const query = 'SELECT \* FROM Users WHERE username = ? AND password = ?';

    db.query(query, [username, password], (err, results) => {

        if (err) return res.status(500).send('Login failed');

        if (results.length === 0) return res.status(400).send('Invalid credentials');

        const token = jwt.sign({ id: results[0].id }, secretKey, { expiresIn: '1h' });

        res.status(200).json({ token });

    });

});

7. Pengujian Setiap Endpoint

Pengujian setiap endpoint dilakukan menggunakan Postman. Setiap endpoint dicoba dengan skenario berikut:

1. Berhasil dengan data valid.
2. Menangani kesalahan dengan data tidak lengkap.
3. Memastikan middleware JWT berfungsi.
4. Menguji respons untuk izin pengguna yang tidak sah.

Berikut adalah contoh hasil tes menggunakan Postman:

[Postman Test]