LAPORAN PRAKTIKUM

SISTEM MANAGEMENT KAMPUS

Disusun untuk Memenuhi Tugas Mata Kuliah Pemrograman Berbasis Platform

Dosen Pengampu:

Alun Sujjada, S.Kom, M.T

Disusun Oleh:

Ujang Herlan (20220040028)

Sany Noor Fauzianty (20220040264)

Ghardi Akbar (20220040050)

FAKULTAS TEKNIK, KOMPUTER DAN DESAIN

JURUSAN TEKNIK INFORMATIKA

UNIVERSITAS NUSA PUTRA

TAHUN 2024

A. DESKRIPSI UMUM

Sistem Manajemen Kampus adalah platform digital yang dirancang untuk menyederhanakan dan meningkatkan efisiensi operasional serta administratif di lingkungan perguruan tinggi atau kampus. Sistem ini mencakup berbagai aspek, mulai dari administrasi akademik, manajemen data mahasiswa, hingga pengelolaan sumber daya kampus secara keseluruhan. Dibangun menggunakan Node.js dengan Express sebagai framework utama. Menggunakan MySQL sebagai database utama, dan terdapat kontrol akses menggunakan JWT (JSON Web Token).

B. RANCANGAN DATABASE DAN API ENDPOINT

a. Rancangan Database

a) Tabel Mahasiswa:

Nama kolom	Tipe	Ukuran	Keterangan
Mahasiswa_id	int	11	● ID Mahasiswa ● PRIMARY KEY ● auto_increment
nama	varchar	25	Nama Mahasiswa
nim	varchar	20	Nomor Induk Mahasiswa
jurusan	varchar	50	Jurusan Mahasiswa
semester	int	11	Semester Mahasiswa
created_at	timestamp		Waktu Pembuatan Data Mahasiswa
updated_at	timestamp		● Waktu Perubahan Terakhir Data Mahasiswa ● on update current_timestamp( )

b) Tabel Mata Kuliah:

Nama kolom	Tipe	Ukuran	Keterangan
mata_kuliah_id	int	11	● ID Mata Kuliah ● PRIMARY KEY ● auto_increment
kode_matkul	varchar	20	Kode Mata Kuliah
nama_matkul	varchar	25	Nama Mata Kuliah

sks	int	11	Satuan Kredit Semester
created_at	timestamp		Waktu Pembuatan Data Matakuliah
updated_at	timestamp		• Waktu Terakhir Perubahan Data Matakuliah • on update current_timestamp( )

c) Tabel Penilaian:

Nama kolom	Tipe	Ukuran	Keterangan
penilaian_id	int	11	• ID Penilaian • PRIMARY KEY • auto_increment
mahasiswa_id	int	11	ID Mahasiswa
mata_kuliah_id	int	11	ID Matakuliah
nilai	int	11	Nilai Mahasiswa
created_at	timestamp		Waktu Pembuatan Data Penilaian
updated_at	timestamp		• Waktu Terakhir Perubahan Data Penilaian • On update curren_tinestamp( )

b. Rancangan API Endpoint

Metode	Endpoint	Keterangan
Mahasiswa		
GET	/mahasiswa	Mendapatkan daftar mahasiswa
POST	/mahasiswa	Mendaftarkan mahasiswa baru
GET	/mahasiswa/{mahasiswa_id}	Mendapatkan detail mahasiswa
PUT	/mahasiswa/{mahasiswa_id}	Mengedit data mahasiswa
DELETE	/mahasiswa/{mahasiswa_id}	Menghapus data mahasiswa
Matakuliah		
GET	/mata-kuliah	Mendapatkan daftar mata kuliah
POST	/mata-kuliah	Menambahkan mata kuliah baru

GET	/mata-kuliah/{mata_kuliah_id}	Mendapatkan detail mata kuliah
PUT	/mata-kuliah/{mata_kuliah_id}	Mengedit data mata kuliah
DELETE	/mata-kuliah/{mata_kuliah_id}	Menghapus mata kuliah
Penilaian		
GET	/penilaian	Mendapatkan daftar penilaian
POST	/penilaian	Menambahkan data penilaian
GET	/penilaian/{penilaian_id}	Mendapatkan detail penilaian
PUT	/penilaian/{penilaian_id}	Mengedit data penilaian
DELETE	/penilaian/{penilaian_id}	Menghapus data penilaian
GET	/penilaian/{mahasiswa_id} /penilaian	Mendapatkan daftar penilaian mahasiswa

C. DOKUMENTASI BACKEND API

a) Tabel Endpoint untuk Mahasiswa

Method	**Endpoint**	**Parameter**	**Hasil yang diharapkan**
GET	/api/mahasiswa		Mengembalikan daftar semua mahasiswa
POST	/api/mahasiswa	name (string), nim (string), jurusan (string), semester (integer)	Mengembalikan objek mahasiswa yang baru dibuat.
GET	/api/mahasiswa/{mahasiswa_id}	mahasiswa_id (integer)	Mengembalikan informasi tentang mahasiswa dengan ID tertentu atau {error: 'Mahasiswa not found'} jika tidak ditemukan.
PUT	/api/mahasiswa/{mahasiswa_id}	mahasiswa_id (integer), name	Mengembalikan {success: true, message:

		(string), nim (string), jurusan (string), semester (integer)	'Mahasiswa updated successfully'} jika berhasil.
DELETE	/api/mahasiswa/{mahasiswa_id}	mahasiswa_id (integer)	Mengembalikan {success: true, message: 'Mahasiswa deleted successfully'} jika berhasil.
GET	/api/mahasiswa/{mahasiswa_id}/penilaian	mahasiswa_id (integer)	Mengembalikan daftar penilaian mahasiswa.

b) Tabel Endpoint untuk Mata Kuliah

Method	**Endpoint**	**Parameter**	**Hasil yang diharapkan**
GET	/api/mahasiswa	Tidak ada	Mengembalikan daftar semua mahasiswa
POST	/api/mahasiswa	name (string), nim (string), jurusan (string), semester (integer)	Mengembalikan objek mahasiswa yang baru dibuat.
GET	/api/mahasiswa/{mahasiswa_id}	mahasiswa_id (integer)	Mengembalikan informasi tentang mahasiswa dengan ID tertentu atau {error: 'Mahasiswa not found'} jika tidak ditemukan.
PUT	/api/mahasiswa/{mahasiswa_id}	mahasiswa_id (integer), name (string), nim (string), jurusan (string), semester (integer)	Mengembalikan {success: true, message: 'Mahasiswa updated successfully'} jika berhasil.
DELET	/api/mahasiswa/{mahasisw	mahasiswa_id	Mengembalikan

E	a_id}	(integer)	{success: true, message: 'Mahasiswa deleted successfully'} jika berhasil.
GET	/api/mahasiswa/{mahasiswa_id}/penilaian	mahasiswa_id (integer)	Mengembalikan daftar penilaian mahasiswa tertentu.

c) Tabel Endpoint Penilaian

Method	**Endpoint**	**Parameter**	**Hasil yang diharapkan**
GET	/api/penilaian	Tidak ada	Mengembalikan daftar semua penilaian.
POST	/api/penilaian	mahasiswa_id (integer), mata_kuliah_id (integer), nilai (integer)	Mengembalikan objek penilaian yang baru dibuat.
GET	/api/penilaian/{penilaian_id}	penilaian_id (integer)	Mengembalikan informasi tentang penilaian dengan ID tertentu atau {error: 'Penilaian not found'} jika tidak ditemukan.
PUT	/api/penilaian/{penilaian_id}	penilaian_id (integer), mahasiswa_id (integer), mata_kuliah_id (integer), nilai (integer)	Mengembalikan {success: true, message: 'Penilaian updated successfully'} jika berhasil.
DELETE	/api/penilaian/{penilaian_id}	penilaian_id (integer)	Mengembalikan {success: true, message: 'Penilaian deleted successfully'} jika berhasil.

D. PENANGANAN ERROR, KEAMANAN DAN DATABASE

a. Penanganan Error

Terdapat penanganan error untuk permintaan yang tidak ditemukan (404) dan kesalahan server internal (500). Error server internal memberikan respons JSON dengan pesan kesalahan.

b. Keamanan

Menggunakan JSON Web Tokens (JWT). Dengan mengambil token dari header Authorization, memverifikasinya dengan kunci rahasia, dan mengekstrak ID pengguna dari token yang terverifikasi, middleware ini memastikan bahwa rute yang membutuhkan autentikasi hanya dapat diakses oleh pengguna yang sah. Pertimbangan keamanan yang diimplementasikan melibatkan penanganan kesalahan yang baik, verifikasi keberadaan dan keaslian token, serta ekstraksi informasi pengguna dengan tujuan identifikasi. Pastikan untuk menjaga kerahasiaan kunci rahasia dan mengikuti praktik keamanan terbaik dalam penanganan JWT, sambil memvalidasi input pengguna untuk melindungi dari potensi kerentanan keamanan.

c. Database

Menggunakan MySQL sebagai database utama. Terdapat tabel mahasiswa, matakuliah dan tabel penilaian.

E. KESIMPULAN

Sistem Manajemen Kampus ini telah mengimplementasikan API untuk manajemen mahasiswa, matakuliah, dan penilaian dengan menggunakan JSON Web Tokens (JWT) untuk menjaga keamanan. Penanganan kesalahan server dan permintaan yang tidak ditemukan (404) sudah diatasi dengan baik, memberikan respons yang informatif kepada pengguna. Namun, untuk meningkatkan dokumentasi, disarankan untuk menyediakan informasi lebih lanjut terkait jenis permintaan API lainnya dan melakukan validasi parameter dengan lebih mendalam. Dokumentasi yang lengkap akan membantu pengembang dan pengguna API memahami dengan baik fungsionalitas yang tersedia, jenis permintaan yang dapat dilakukan, serta aturan dan batasan yang perlu diperhatikan dalam penggunaan API ini. Hal ini akan meningkatkan penggunaan sistem secara keseluruhan dan memfasilitasi pengembangan aplikasi yang terintegrasi dengan API kampus ini.

F. LAMPIRAN

a. Dalam file config terdapat 2 direktori:

- dbconfig.js

```
const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('sistem_management_kampus', 'root', 'herlan', {
  host: 'localhost',
  dialect: 'mysql',
});

module.exports = sequelize;
```

- jwtConfig.js

```
module.exports = {
    secretKey: 'secret_key_for_jwt', // Ganti dengan kunci rahasia yang kuat
  };
```

b. Dalam file middleware terdapat direktori authMiddleware,js

```
const jwt = require('jsonwebtoken');
const { secretKey } = require('../config/jwtConfig');

const authMiddleware = (req, res, next) => {
  // Mendapatkan token dari header Authorization
  const token = req.headers.authorization;

  // Memeriksa keberadaan token
  if (!token) {
    return res.status(401).json({ error: 'Unauthorized - Token not provided' });
  }

  // Verifikasi token
  jwt.verify(token, secretKey, (err, decoded) => {
    if (err) {
      return res.status(401).json({ error: 'Unauthorized - Invalid token' });
    }

    // Menyimpan ID pengguna yang terotentikasi pada objek request
    req.userId = decoded.id;

    // Lanjutkan ke middleware atau rute berikutnya
    next();
  });
};

module.exports = authMiddleware;
```

c. Dalam file controllers terdapat 3 direktori:
    - mahasiswaController.js

```
const Mahasiswa = require('../models/mahasiswaModel');

// Mendapatkan daftar mahasiswa
const getAllMahasiswa = async (req, res) => {
  try {
    const mahasiswaList = await Mahasiswa.findAll();
    res.status(200).json({ mahasiswaList });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

// Mendaftarkan mahasiswa baru
const createMahasiswa = async (req, res) => {
  const { nama, nim, jurusan, semester } = req.body;
  try {
    const newMahasiswa = await Mahasiswa.create({ nama, nim, jurusan, semester });
    res.status(201).json({ message: 'Mahasiswa berhasil terdaftar', newMahasiswa });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

// Mendapatkan detail mahasiswa berdasarkan ID
const getMahasiswaById = async (req, res) => {
  const { mahasiswa_id } = req.params;
  try {
    const mahasiswa = await Mahasiswa.findByPk(mahasiswa_id);
    if (!mahasiswa) {
      res.status(404).json({ error: 'Mahasiswa tidak ditemukan' });
      return;
    }
    res.status(200).json({ mahasiswa });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

// Mengedit data mahasiswa
const updateMahasiswa = async (req, res) => {
  const { mahasiswa_id } = req.params;
  const { nama, semester } = req.body;
  try {
    const mahasiswa = await Mahasiswa.findByPk(mahasiswa_id);
    if (!mahasiswa) {
      res.status(404).json({ error: 'Mahasiswa tidak ditemukan' });
      return;
    }
    await mahasiswa.update({ nama, semester });
    res.status(200).json({ message: 'Data mahasiswa berhasil diperbarui' });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

// Menghapus data mahasiswa
const deleteMahasiswa = async (req, res) => {
  const { mahasiswa_id } = req.params;
  try {
    const mahasiswa = await Mahasiswa.findByPk(mahasiswa_id);
    if (!mahasiswa) {
      res.status(404).json({ error: 'Mahasiswa tidak ditemukan' });
      return;
    }
    await mahasiswa.destroy();
    res.status(200).json({ message: 'Data mahasiswa berhasil dihapus' });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

module.exports = {
  getAllMahasiswa,
  createMahasiswa,
  getMahasiswaById,
  updateMahasiswa,
  deleteMahasiswa,
};
```

- matakuliahController.js

```
const MataKuliah = require('../models/matakuliahModel');

// Mendapatkan daftar mata kuliah
const getAllMataKuliah = async (req, res) => {
  try {
    const mataKuliahList = await MataKuliah.findAll();
    res.status(200).json({ mataKuliahList });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

// Menambahkan mata kuliah baru
const createMataKuliah = async (req, res) => {
  const { kode_matkul, nama_matkul, sks } = req.body;
  try {
    const newMataKuliah = await MataKuliah.create({ kode_matkul, nama_matkul, sks });
    res.status(201).json({ message: 'Mata kuliah berhasil ditambahkan', newMataKuliah });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

// Mendapatkan detail mata kuliah berdasarkan ID
const getMataKuliahById = async (req, res) => {
  const { mata_kuliah_id } = req.params;
  try {
    const mataKuliah = await MataKuliah.findByPk(mata_kuliah_id);
    if (!mataKuliah) {
      res.status(404).json({ error: 'Mata kuliah tidak ditemukan' });
      return;
    }
    res.status(200).json({ mataKuliah });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

// Mengedit data mata kuliah
const updateMataKuliah = async (req, res) => {
  const { mata_kuliah_id } = req.params;
  const { kode_matkul, nama_matkul, sks } = req.body;
  try {
    const mataKuliah = await MataKuliah.findByPk(mata_kuliah_id);
    if (!mataKuliah) {
      res.status(404).json({ error: 'Mata kuliah tidak ditemukan' });
      return;
    }
    await mataKuliah.update({ kode_matkul, nama_matkul, sks });
    res.status(200).json({ message: 'Data mata kuliah berhasil diperbarui' });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

// Menghapus mata kuliah
const deleteMataKuliah = async (req, res) => {
  const { mata_kuliah_id } = req.params;
  try {
    const mataKuliah = await MataKuliah.findByPk(mata_kuliah_id);
    if (!mataKuliah) {
      res.status(404).json({ error: 'Mata kuliah tidak ditemukan' });
      return;
    }
    await mataKuliah.destroy();
    res.status(200).json({ message: 'Mata kuliah berhasil dihapus' });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

module.exports = {
  getAllMataKuliah,
  createMataKuliah,
  getMataKuliahById,
  updateMataKuliah,
  deleteMataKuliah,
};
```

- penilaianController.js

```
const Penilaian = require('../models/penilaianModel');

// Mendapatkan daftar penilaian
const getAllPenilaian = async (req, res) => {
  try {
    const penilaianList = await Penilaian.findAll();
    res.status(200).json({ penilaianList });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

// Menambahkan data penilaian baru
const createPenilaian = async (req, res) => {
  const { mahasiswa_id, mata_kuliah_id, nilai } = req.body;
  try {
    const newPenilaian = await Penilaian.create({ mahasiswa_id, mata_kuliah_id, nilai });
    res.status(201).json({ message: 'Data penilaian berhasil ditambahkan', newPenilaian });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

// Mendapatkan detail penilaian berdasarkan ID
const getPenilaianById = async (req, res) => {
  const { penilaian_id } = req.params;
  try {
    const penilaian = await Penilaian.findByPk(penilaian_id);
    if (!penilaian) {
      res.status(404).json({ error: 'Data penilaian tidak ditemukan' });
      return;
    }
    res.status(200).json({ penilaian });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

// Mengedit data penilaian
const updatePenilaian = async (req, res) => {
  const { penilaian_id } = req.params;
  const { nilai } = req.body;
  try {
    const penilaian = await Penilaian.findByPk(penilaian_id);
    if (!penilaian) {
      res.status(404).json({ error: 'Data penilaian tidak ditemukan' });
      return;
    }
    await penilaian.update({ nilai });
    res.status(200).json({ message: 'Data penilaian berhasil diperbarui' });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

// Menghapus data penilaian
const deletePenilaian = async (req, res) => {
  const { penilaian_id } = req.params;
  try {
    const penilaian = await Penilaian.findByPk(penilaian_id);
    if (!penilaian) {
      res.status(404).json({ error: 'Data penilaian tidak ditemukan' });
      return;
    }
    await penilaian.destroy();
    res.status(200).json({ message: 'Data penilaian berhasil dihapus' });
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: 'Internal Server Error' });
  }
};

module.exports = {
  getAllPenilaian,
  createPenilaian,
  getPenilaianById,
  updatePenilaian,
  deletePenilaian,
};
```

d. Dalam file models terdapat 3 direktori:

- mahasiswaModel.js

```
const { DataTypes, Sequelize } = require("sequelize");
const sequelize = require("../config/dbconfig");

const Mahasiswa = sequelize.define("Mahasiswa", {
  mahasiswa_id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  nama: {
    type: DataTypes.STRING(255),
    allowNull: false,
  },
  nim: {
    type: DataTypes.STRING(20),
    allowNull: false,
  },
  jurusan: {
    type: DataTypes.STRING(50),
    allowNull: false,
  },
  semester: {
    type: DataTypes.INTEGER,
    allowNull: false,
  },
  created_at: {
    type: DataTypes.DATE,
    defaultValue: Sequelize.fn('NOW'),
  },
  updated_at: {
    type: DataTypes.DATE,
    defaultValue: Sequelize.fn('NOW'),
    onUpdate: Sequelize.fn('NOW'),
  },
});

module.exports = Mahasiswa;
```

- matakuliahModel.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/dbconfig");

const MataKuliah = sequelize.define("MataKuliah", {
  mata_kuliah_id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  kode_matkul: {
    type: DataTypes.STRING(20),
    allowNull: false,
  },
  nama_matkul: {
    type: DataTypes.STRING(255),
    allowNull: false,
  },
  sks: {
    type: DataTypes.INTEGER,
    allowNull: false,
  },
  created_at: {
    type: DataTypes.DATE,
    defaultValue: DataTypes.NOW,
  },
  updated_at: {
    type: DataTypes.DATE,
    defaultValue: DataTypes.NOW,
    onUpdate: DataTypes.NOW,
  },
});

module.exports = MataKuliah;
```

- penilaianModel.js

```
const { DataTypes } = require("sequelize");
const sequelize = require("../config/dbconfig");

const Penilaian = sequelize.define("Penilaian", {
  penilaian_id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    autoIncrement: true,
  },
  mahasiswa_id: {
    type: DataTypes.INTEGER,
  },
  mata_kuliah_id: {
    type: DataTypes.INTEGER,
  },
  nilai: {
    type: DataTypes.INTEGER,
    allowNull: false,
  },
  created_at: {
    type: DataTypes.DATE,
    defaultValue: DataTypes.NOW,
  },
  updated_at: {
    type: DataTypes.DATE,
    defaultValue: DataTypes.NOW,
    onUpdate: DataTypes.NOW,
  },
});

module.exports = Penilaian;
```

e. Dalam file routers terdapat 3 direktori:

- mahasiswaRouter.js

```
const express = require('express');
const router = express.Router();
const mahasiswaController = require('../controllers/mahasiswaController');

// Define Mahasiswa routes
router.get('/', mahasiswaController.getAllMahasiswa);
router.post('/', mahasiswaController.createMahasiswa);
router.get('/:mahasiswa_id', mahasiswaController.getMahasiswaById);
router.put('/:mahasiswa_id', mahasiswaController.updateMahasiswa);
router.delete('/:mahasiswa_id', mahasiswaController.deleteMahasiswa);

module.exports = router;

```

- matakuliahRouter.js

```
const express = require('express');
const router = express.Router();
const mataKuliahController = require('../controllers/matakuliahController');

// Define Mata Kuliah routes
router.get('/', mataKuliahController.getAllMataKuliah);
router.post('/', mataKuliahController.createMataKuliah);
router.get('/:mata_kuliah_id', mataKuliahController.getMataKuliahById);
router.put('/:mata_kuliah_id', mataKuliahController.updateMataKuliah);
router.delete('/:mata_kuliah_id', mataKuliahController.deleteMataKuliah);

module.exports = router;
```

- penilaianRouter.js

```
const express = require('express');
const router = express.Router();
const penilaianController = require('../controllers/penilaianController');

// Define Penilaian routes
router.get('/', penilaianController.getAllPenilaian);
router.post('/', penilaianController.createPenilaian);
router.get('/:penilaian_id', penilaianController.getPenilaianById);
router.put('/:penilaian_id', penilaianController.updatePenilaian);
router.delete('/:penilaian_id', penilaianController.deletePenilaian);

module.exports = router;
```