LAPORAN FINAL PROJECT AKHIR

“Projek Musik”

Pemrograman Jaringan 202223430058

DOSEN PENGAMPU:

Randi Proska Sandra,S.Pd, M.Sc

OLEH:

Aldhy	**(21343017)**
Egi Yoni Sandra	**(21343005)**
Muhammad Farras Fiqhan	**(21343029)**

PROGRAM STUDI S1 INFORMATIKA

TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2023

Nama program : Aplikasi musik Online

Link Github : Aldhy-dwy23/Project_Music (github.com)

A. LATAR BELAKANG PROGRAM

Aplikasi ini adalah sebuah program yang dibuat untuk membantu pengguna menikmati musik dengan lebih mudah. Kami menggunakan bahasa pemrograman Node.js dan framework Express untuk membuat aplikasi ini. Aplikasi ini memiliki beberapa fitur, seperti membuat akun, login, melihat informasi lagu, dan mengelola playlist lagu.

Dalam membuat aplikasi ini, Kami menggunakan MongoDB sebagai tempat penyimpanan data, seperti nama pengguna, email, dan kata sandi. Saat pengguna membuat akun atau login, aplikasi akan memeriksa data yang dimasukkan untuk memastikan semuanya benar dan aman.

Selain itu, ada fitur otentikasi yang memastikan hanya pengguna yang login yang bisa mengakses beberapa bagian tertentu dari aplikasi, seperti melihat playlist atau informasi lagu.

Aplikasi ini juga memperhatikan tampilan yang disajikan kepada pengguna. kami menggunakan Handlebars untuk membuat halaman-halaman yang interaktif dan menarik. Selain itu, ada folder khusus untuk menyimpan gambar, stylesheet, dan elemen-elemen lain yang membuat tampilan aplikasi lebih menarik.

Fitur lain yang ditawarkan aplikasi ini adalah manajemen playlist, di mana pengguna bisa melihat daftar playlist, membuat playlist baru, dan menambahkan lagu ke dalamnya. Semua data playlist disimpan dengan aman di MongoDB.

Aplikasi ini juga bisa menangani beberapa masalah umum yang mungkin muncul, seperti kesalahan dalam memasukkan data, masalah koneksi ke database, atau masalah saat login. Ini semua dilakukan untuk memastikan pengguna memiliki pengalaman yang lancar dan menyenangkan dalam menggunakan aplikasi musik ini..

B. ELEMEN ATAU KONSEP PEMROGRAMAN JARINGAN YANG DIGUNAKAN

1. Protokol HTTP (Express.js)

   Express.js digunakan sebagai framework untuk menangani protokol HTTP dalam pembangunan aplikasi web. Berikut adalah contoh penggunaan Express untuk menangani permintaan HTTP

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

// Mengimpor modul yang diperlukan

dari Express, HBS, dan Mongoose

const express = require("express");

const hbs = require("hbs");

const path = require("path");
```

```
const session = require("express-
session");
const mongoose =
require('mongoose');

// Mengimpor model LogInCollection
serta modul untuk playlist
const LogInCollection =
require("./utils/mongo");
const forecast =
require("./utils/song");
const playlistUtils =
require("./utils/playlist"); //
Mengimpor playlist.js

// Membuat instance Express
const app = express();

// Menentukan port server
const port = process.env.PORT ||
3030;
```

2. Fetch Data dari API (Axios)

Axios digunakan untuk melakukan permintaan HTTP ke API Spotify . Contoh penggunaan Axios dapat dilihat pada modul utils untuk mendapatkan data dari API

```
const axios = require("axios");

// Fungsi untuk mendapatkan informasi lagu dari Spotify
async function getInfoSong(query, callback) {
  try {
    // Ganti dengan client ID dan client secret Spotify Anda
    const clientId = "a41d1f3274744fc5bb490e0d6f481a22";
    const clientSecret = "0c79a6b0438b4c22a6c6916580a8245c";

    // Mendapatkan access token dari Spotify
    const tokenResponse = await axios.post(
      "https://accounts.spotify.com/api/token",
      "grant_type=client_credentials",
      {
        headers: {
```

```
            Authorization: `Basic
${Buffer.from(`${clientId}:${clientSecret}`).toString("base64")}`,
            "Content-Type": "application/x-www-form-urlencoded",
          },
        }
      );

      const accessToken = tokenResponse.data.access_token;

      // Mencari lagu berdasarkan query
      const searchResponse = await
axios.get(`https://api.spotify.com/v1/search?q=${query}&type=track`, {
        headers: {
          Authorization: `Bearer ${accessToken}`,
        },
      });
```

3. Protokol MonggoDB (Database)
   Koneksi ke MonggoDB dilakukan menggunakan Mongoose, sebuah object data modeling.

```
// Menghubungkan ke database MongoDB
mongoose.connect('mongodb://localhost:27017/Song', {
useNewUrlParser: true, useUnifiedTopology: true });
```

4. Handlebars (hbs)
   Handlebars adalah mesin template yang memungkinkan pengembang untuk menyusun tata letak halaman secara dinamis. Handlebars digunakan sebagai view engine dalam Express.js.

```
const hbs = require("hbs");
```

5. Express Session (express-session)
   Express-session digunakan untuk menangani manajemen sesi pengguna pada aplikasi Express.js. Ini membantu dalam menyimpan dan memanipulasi data sesi pengguna.

```
const session = require("express-session");
```

6. Mongoose (mongoose)
   Mongoose adalah modul ODM (Object Data Modeling) untuk MongoDB dan menyediakan cara yang nyaman untuk berinteraksi dengan database MongoDB.

```
// Menghubungkan ke database MongoDB
mongoose.connect('mongodb://localhost:27017/Song', {
useNewUrlParser: true, useUnifiedTopology: true });
```

C. PENJELASAN APLIKASI

Program di atas adalah bagian dari aplikasi musik sederhana yang memungkinkan pengguna untuk membuat akun, login, melihat informasi lagu, dan mengelola playlist.

Aplikasi ini menggunakan Node.js dan Express sebagai backend, MongoDB sebagai database, serta HBS (Handlebars) sebagai view engine untuk menghasilkan tampilan halaman HTML dinamis. Berikut adalah beberapa fitur utama dari aplikasi tersebut:

1. Fitur Login dan Register

   Fitur ini memungkinkan pengguna untuk membuat akun baru atau masuk ke akun yang sudah ada. Data pengguna disimpan dalam database MongoDB. Pengguna dapat mengakses fitur-fitur lainnya setelah berhasil login.

```
app.post("/signup", async (req, res) => {
  // ... (validasi dan penyimpanan data pengguna)
});

app.post("/login", async (req, res) => {
  // ... (validasi login dan pengecekan keberadaan pengguna di
database)
});
```

2. Fitur Dashboard

   Dashboard menyajikan halaman utama yang memberikan gambaran singkat tentang aplikasi dan informasi tentang pengembang. Hal ini bertujuan untuk memberikan pengguna pemahaman yang jelas tentang tujuan dan manfaat dari aplikasi ini.

```
app.get("/", (req, res) => {
  // ... (render halaman utama dengan status otentikasi)
});
```

3. Fitur Informasi Lagu

   Pada halaman Al-Qur'an, pengguna dapat membaca teks Al-Qur'an dan memilih surah tertentu untuk melihat isi seluruh surah tersebut.

```
app.get("/infosong", authenticateMiddleware, async (req, res)
=> {
  // ... (mengambil informasi lagu, memeriksa otentikasi, dan
merender halaman)
});
```

4. Fitur Manajemen Playlist:

   Pada Aplikasi ini memungkinkan pengguna untuk melihat playlist, menambahkan playlist baru, dan menambahkan lagu ke dalam playlist.

```
app.get("/playlist", authenticateMiddleware, async (req, res)
=> {
  // ... (mengambil playlist dari MongoDB dan merender halaman
playlist)
});

app.post("/playlist", authenticateMiddleware, async (req, res)
=> {
  // ... (menambahkan playlist baru ke MongoDB)
});
```

5. Fitur Middleware Otentikasi:

Pada Terdapat middleware authenticateMiddleware untuk memastikan bahwa hanya pengguna yang sudah login yang dapat mengakses beberapa fitur.

```
function authenticateMiddleware(req, res, next) {
  // ... (pengecekan otentikasi)
}
```

6. Fitur Logout

Pengguna dapat logout dari akun mereka, yang menghapus sesi dan mengarahkannya kembali ke halaman utama.

```
app.get("/logout", (req, res) => {
  // ... (menghapus sesi dan mengarahkan kembali ke halaman utama)
});
```

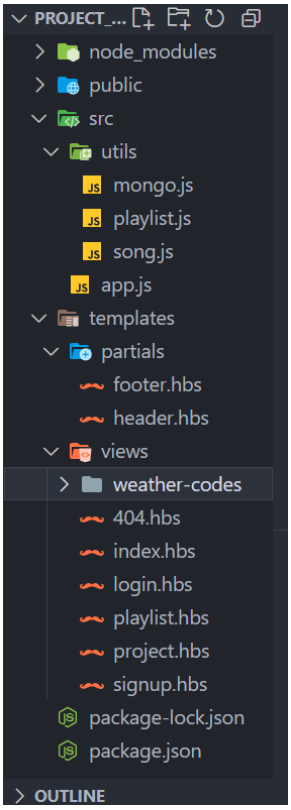
D. LANGKAH-LANGLAH PEMBUATAN APLIKASI

1. Persiapaan Awal

Instal Node.js dan MongoDB:
Pastikan Node.js dan MongoDB terinstal di komputer Anda.

Inisialisasi Proyek:
Buat direktori proyek baru dan jalankan npm init untuk menginisialisasi proyek Node.js

2. Struktur Proyek



2. Instalasi Modul dan Pustaka:
   - Instal Gunakan perintah npm install express mongoose express-session hbs untuk menginstal modul-modul yang diperlukan.

3. Konfigurasi Struktur proyek
    - Atur struktur direktori proyek dengan folder untuk views (templates), public (asets statis), dan utils (modul-modul bantuan).

4. Pembuatan Server dan Koneksi Database
   Buat file server menggunakan Express dan atur koneksi ke MongoDB.

```
const express = require("express");
const mongoose = require("mongoose");
const app = express();
const port = process.env.PORT || 3030;

// Koneksi ke MongoDB
mongoose.connect('mongodb://localhost:27017/Song', {
useNewUrlParser: true, useUnifiedTopology: true });

// ... (konfigurasi middleware dan routing)

app.listen(port, () => {
  console.log("Server is running on port", port);
});
```

5. Pmbuatan model database dan middleware
   Tentukan model MongoDB untuk koleksi pengguna dan playlist.

```
const mongoose = require("mongoose");

const LogInCollection = mongoose.model("LogInCollection", {
  name: String,
  email: String,
  password: String,
});

// ... (definisikan model lainnya jika diperlukan)
```

   Buat middleware untuk memeriksa status otentikasi pengguna sebelum memberikan akses ke beberapa endpoint.

```
function authenticateMiddleware(req, res, next) {
  if (!req.session.isAuthenticated) {
    return res.redirect("/login");
  }
  next();
}
```

6. Pembuatan routing dan endpoint
   Tentukan endpoint untuk signup, login, dan logout.

```
app.post("/signup", async (req, res) => {
  // ... (logika signup)
});

app.post("/login", async (req, res) => {
  // ... (logika login)
});
```

```
app.get("/logout", (req, res) => {
  // ... (logika logout)
});
```

Atur endpoint untuk halaman utama, playlist, dan informasi lagu.

```
app.get("/", (req, res) => {
  // ... (logika halaman utama)
});

app.get("/playlist", authenticateMiddleware, async (req, res)
=> {
  // ... (logika playlist)
});

app.get("/infosong", authenticateMiddleware, async (req, res)
=> {
  // ... (logika informasi lagu)
});
```

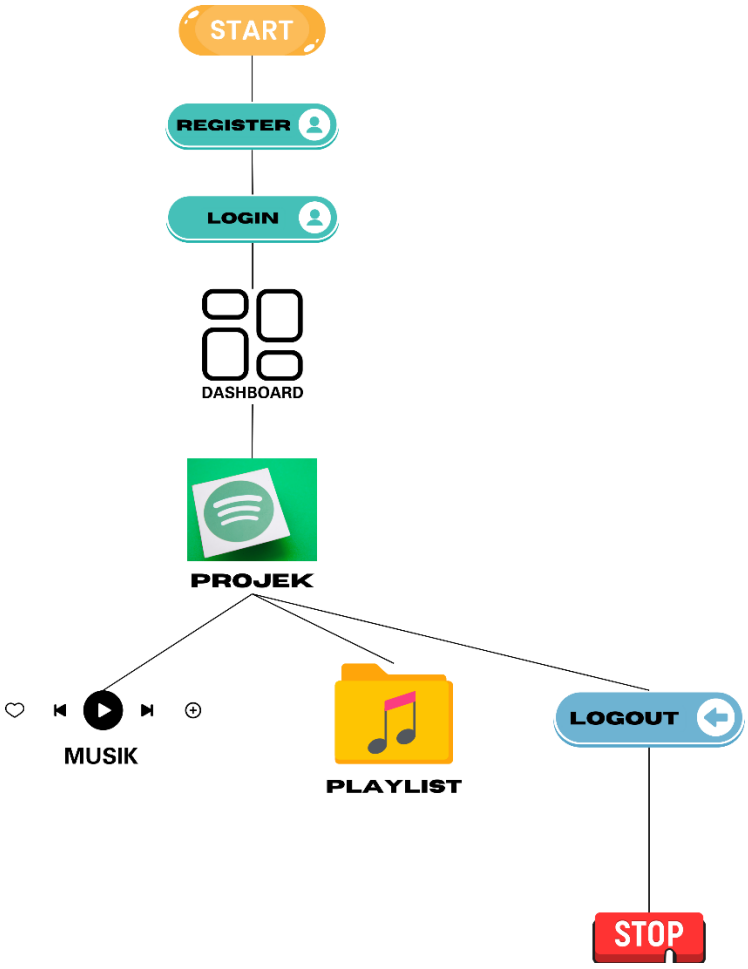
7. Pembuatan tampilan dengan handlebars
   Buat tampilan HBS untuk halaman signup, login, playlist, informasi lagu, dll.

8. Pembuatan fungsi fungsi bantuan
   Buat Fungsi-Fungsi untuk Manajemen Playlist: Implementasikan fungsi-fungsi bantuan untuk menambahkan, mengambil, atau menghapus playlist dan lagu dari database

9. Penanganan error dan validasi
   Tambahkan logika untuk menangani error, seperti validasi data pengguna dan penanganan kesalahan koneksi database.

10. Uji coba dan optimalisasi
    Uji coba aplikasi secara menyeluruh dan lakukan optimalisasi jika diperlukan.

E. DIAGRAM APLIKASI



F. REFERENSI

Anderson, A., Maystre, L., Anderson, I., Mehrotra, R., & Lalmas, M. (2020). Algorithmic effects on the diversity of consumption on Spotify. Proceedings of The Web Conference 2020 (S. 2155–2165)

Spotify, 2019. What is Spotify?. [online] Spotify, Inc. Tersedia di: https://support.spotify.com/us/using_spot ify/the_basics/what-is-spotify/

Dalton, Brian & Behm, David. 2007. Effects of noise and music on human and task performance: A systematic review. Occupational Ergonomics. 7. 143-152.Setiawan, W., & Rochim, A. F. (n.d.). APLIKASI ALQURAN DAN TERJEMAHAN PADA

Statista, 2019. Number of Spotify monthly active users (MAUs) worldwide from 1st quarter 2015 to 4th quarter 2018 (in millions)

2023. A Unified API Wrapper to Simplify Web Data Collection| PyData Global 2020.
G. A. V. M. Giri, ―Klasifikasi Musik Berdasarkan Genre dengan Metode K-Nearest Neighbor,‖ Jurnal Ilmu Komputer, vol. 11, no. 2, Art. no. 2, Oct. 2018