IMPLEMENTASI FULLSTACK WEB APPLICATION PADA ISLAND PEMILIHAN KOTA HUNIAN TERBAIK UNTUK CO-WORKER BERBASIS ALGORITMA PROMETHEE

"Diajukan Untuk Memenuhi Tugas Pada Mata Kuliah Pemrograman Web Lanjut Dengan Dosen Pahrul Irfan S.Kom., M.Kom."

Anggota Kelompok:

Nabila Noor Azizah (F1D022081)

Abdul Wafa (F1D022104)

Saufan Ridho (F1D02310091)

UNIVERSITAS MATARAM

2025

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi telah membawa perubahan signifikan dalam cara manusia mengelola data dan mengambil keputusan. Pada era digital saat ini, pengambilan keputusan tidak lagi hanya bergantung pada intuisi atau pengalaman subjektif, melainkan membutuhkan dukungan sistem yang mampu mengolah data secara objektif, terstruktur, dan terukur. Hal ini menjadi semakin penting ketika keputusan yang diambil melibatkan banyak alternatif dan kriteria yang saling berkaitan.

Salah satu permasalahan yang sering dihadapi dalam kehidupan sehari-hari adalah menentukan kota terbaik untuk ditinggali, dikembangkan, atau dijadikan pusat aktivitas tertentu. Setiap kota memiliki karakteristik yang berbeda-beda, seperti jumlah *coworking space*, fasilitas pendukung, tingkat perkembangan, serta indikator lain yang relevan. Dalam praktiknya, proses pemilihan kota sering dilakukan secara manual berdasarkan opini pribadi, rekomendasi informal, atau informasi yang tidak terintegrasi, sehingga keputusan yang dihasilkan cenderung subjektif dan sulit dipertanggungjawabkan secara sistematis.

Permasalahan tersebut semakin kompleks ketika jumlah kota yang dibandingkan semakin banyak dan kriteria penilaian semakin beragam. Pengguna akan mengalami kesulitan dalam membandingkan setiap alternatif secara adil, yang pada akhirnya dapat menyebabkan keputusan yang kurang optimal. Kondisi ini menunjukkan perlunya suatu pendekatan yang mampu membantu proses pengambilan keputusan secara rasional dan berbasis data.

Sistem Pendukung Keputusan (SPK) hadir sebagai solusi untuk mengatasi permasalahan tersebut. SPK dirancang untuk membantu pengguna dalam mengambil keputusan dengan memanfaatkan metode dan model matematis tertentu. Salah satu metode yang banyak digunakan dalam SPK adalah Promethee (*Preference Ranking Organization Method for Enrichment Evaluation*), yaitu metode pengambilan keputusan multikriteria yang mampu memberikan peringkat alternatif berdasarkan tingkat preferensi.

Berdasarkan permasalahan tersebut, dikembangkanlah aplikasi Island, sebuah aplikasi berbasis web yang berfungsi sebagai sistem pendukung keputusan dalam menentukan kota terbaik berdasarkan kriteria yang telah ditentukan. Aplikasi ini dirancang untuk mengelola

data kota, kriteria, dan nilai penilaian secara terpusat, serta melakukan perhitungan dan perankingan secara otomatis menggunakan metode Promethee. Dengan adanya aplikasi Island, diharapkan proses pengambilan keputusan menjadi lebih objektif, transparan, dan efisien.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan, maka rumusan masalah dalam pengembangan aplikasi Island adalah sebagai berikut:

1. Bagaimana merancang sebuah sistem pendukung keputusan berbasis web yang mampu mengelola data kota dan kriteria penilaian secara terintegrasi?
2. Bagaimana mengimplementasikan metode Promethee untuk melakukan perhitungan dan perankingan kota berdasarkan banyak kriteria?
3. Bagaimana menyediakan antarmuka aplikasi yang mudah digunakan agar pengguna dapat melakukan analisis kota dengan cepat dan efisien?
4. Bagaimana memastikan proses pengambilan keputusan dilakukan secara objektif dan dapat dipertanggungjawabkan secara sistematis?

1.3 Batasan Masalah

Agar pembahasan dan pengembangan aplikasi tidak meluas serta tetap terfokus pada tujuan utama, maka ditetapkan beberapa batasan masalah sebagai berikut:

1. Aplikasi Island hanya digunakan untuk proses pemilihan dan perankingan kota berdasarkan kriteria yang telah ditentukan.
2. Metode pengambilan keputusan yang digunakan dalam aplikasi ini adalah metode Promethee.
3. Pengelolaan data kota, kriteria, dan nilai penilaian hanya dapat dilakukan oleh admin.
4. Aplikasi dikembangkan berbasis web dengan arsitektur *client-server*.
5. Backend aplikasi menggunakan Node.js dan Express.js, sedangkan frontend menggunakan Vue.js 3.
6. Database yang digunakan adalah MySQL.
7. Sistem autentikasi menggunakan JSON Web Token (JWT).
8. Aplikasi tidak membahas perbandingan metode SPK lain di luar Promethee.

1.4 Tujuan Penelitian

Berdasarkan rumusan masalah di atas, maka tujuan dari pengembangan aplikasi Island adalah:

1. Merancang dan membangun arsitektur sistem pendukung keputusan berbasis web yang mampu mengintegrasikan pengelolaan data kota (alternatif) dan kriteria penilaian secara dinamis menggunakan teknologi Fullstack Modern (Express.js dan MySQL).
2. Mengimplementasikan algoritma metode Promethee ke dalam logika sistem untuk melakukan perhitungan pembobotan multikriteria dan menghasilkan perankingan kota secara otomatis dan akurat.
3. Menyediakan antarmuka pengguna (User Interface) berbasis Single Page Application (Vue.js) yang intuitif dan responsif, sehingga memudahkan pengguna dalam melakukan navigasi dan analisis data kota secara efisien.
4. Menghasilkan rekomendasi keputusan pemilihan lokasi hunian co-worker yang objektif, transparan, dan dapat dipertanggungjawabkan melalui validasi sistematis berdasarkan bobot preferensi yang ditentukan.

BAB II
ANALISIS DAN PERANCANGAN SISTEM

2.1 Analisis Kebutuhan Sistem

Analisis kebutuhan merupakan tahap awal yang sangat penting dalam pengembangan sistem. Tahap ini bertujuan untuk mengidentifikasi fungsi utama, batasan, serta karakteristik sistem yang harus dipenuhi agar aplikasi Island dapat berjalan sesuai dengan tujuan yang diharapkan. Analisis kebutuhan dibagi menjadi dua kategori utama, yaitu kebutuhan fungsional dan kebutuhan non-fungsional.

2.1.1 Kebutuhan Fungsional

Kebutuhan fungsional menggambarkan layanan, fitur, dan aktivitas yang harus dapat dilakukan oleh sistem. Kebutuhan ini berhubungan langsung dengan interaksi pengguna terhadap sistem.

1. Autentikasi Admin

   Sistem harus menyediakan mekanisme *login* bagi admin untuk mengamankan akses ke fitur manajemen data. Autentikasi dilakukan menggunakan *username* dan *password*, yang kemudian diverifikasi oleh sistem *backend*. Untuk menjaga keamanan sesi, sistem menggunakan JSON Web Token (JWT) sebagai penanda autentikasi yang akan divalidasi pada setiap permintaan ke server.

2. *Dashboard* Admin

   Setelah berhasil *login*, admin diarahkan ke halaman *dashboard*. *Dashboard* berfungsi sebagai pusat navigasi dan memberikan gambaran umum mengenai kondisi sistem, seperti jumlah data kota dan jumlah kriteria yang tersedia. Keberadaan *dashboard* membantu admin dalam memantau dan mengelola sistem secara lebih efisien.

3. Manajemen Data Kota

   Sistem harus memungkinkan admin untuk melakukan pengelolaan data kota secara penuh, meliputi penambahan, penampilan, pembaruan, dan penghapusan data (CRUD). Data kota ini menjadi alternatif dalam sistem pendukung keputusan dan akan digunakan dalam proses perhitungan metode Promethee.

4. Manajemen Data Kriteria

   Admin dapat mengelola data kriteria penilaian yang digunakan untuk membandingkan kota. Setiap kriteria memiliki atribut bobot dan tipe (*benefit* atau *cost*) yang akan mempengaruhi proses perhitungan. Pengelolaan kriteria yang fleksibel memungkinkan sistem untuk menyesuaikan kebutuhan analisis di masa depan.

5. Manajemen Nilai Kota

   Sistem menyediakan fitur untuk memasukkan nilai setiap kota berdasarkan kriteria yang telah ditentukan. Nilai ini merupakan input utama dalam metode Promethee dan menjadi dasar dalam proses perhitungan peringkat kota.

6. Analisis dan Perankingan Kota

   Pengguna dapat memilih minimal dua kota untuk dianalisis. Sistem kemudian memproses data tersebut menggunakan metode Promethee dan menghasilkan peringkat kota berdasarkan nilai *net flow*. Hasil analisis ditampilkan dalam bentuk tabel yang mudah dipahami oleh pengguna.

2.1.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional berkaitan dengan kualitas sistem, teknologi yang digunakan, serta batasan teknis yang harus dipenuhi.

1. Keamanan Sistem

   Sistem harus mampu melindungi data dan akses pengguna. Oleh karena itu, autentikasi berbasis JWT diterapkan untuk memastikan bahwa hanya pengguna yang berwenang yang dapat mengakses fitur tertentu.

2. Kinerja dan Aksesibilitas

   Aplikasi harus dapat diakses melalui browser tanpa memerlukan instalasi tambahan. Sistem dirancang agar mampu merespons permintaan pengguna dengan cepat dan stabil.

3. Teknologi yang Digunakan

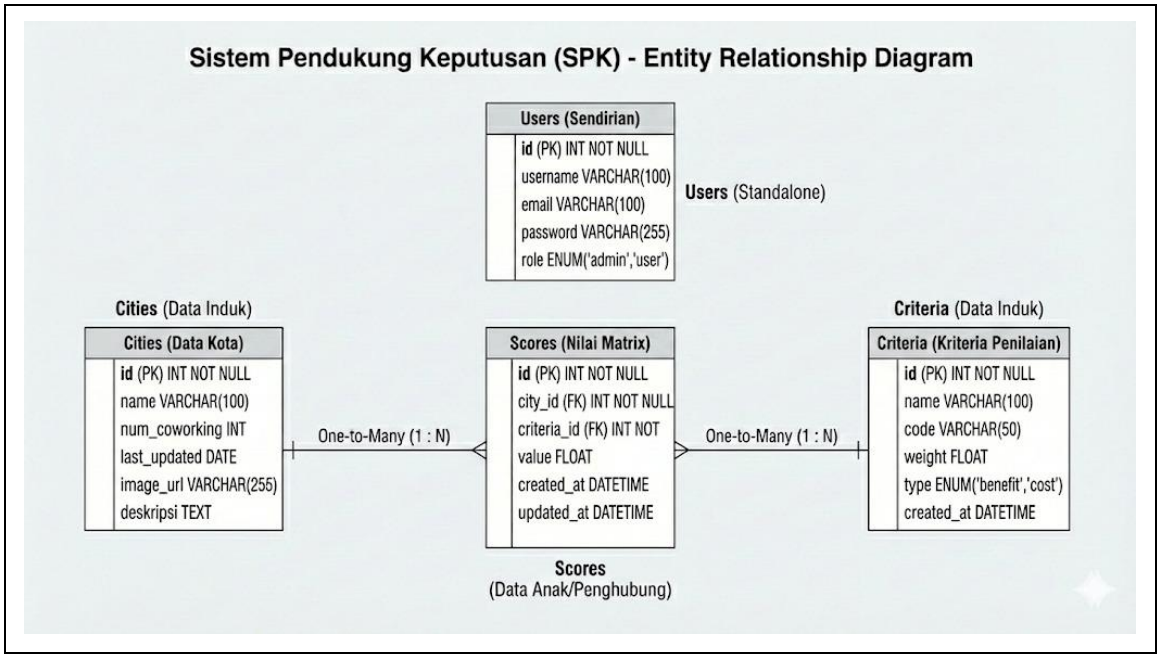
   Backend sistem dibangun menggunakan Node.js dengan *framework* Express.js, sedangkan frontend menggunakan Vue.js 3 dengan pendekatan *Single Page Application* (SPA). *Database* yang digunakan adalah MySQL sebagai penyimpanan data relasional.

4. Kemudahan Penggunaan (Usability)

Antarmuka aplikasi dirancang agar mudah digunakan dan responsif. Sistem juga memberikan umpan balik kepada pengguna melalui dialog atau notifikasi ketika terjadi kesalahan atau proses berhasil dilakukan.

2.2 Perancangan Basis Data

Perancangan basis data bertujuan untuk menentukan struktur penyimpanan data yang efisien dan mendukung seluruh proses dalam sistem. Basis data aplikasi Island dirancang menggunakan model relasional.



Gambar 1 Struktur Basis Data

2.2.1 Struktur Tabel

Basis data terdiri dari beberapa tabel utama, yaitu tabel *users, cities, criteria,* dan *scores.* Tabel *users* digunakan untuk menyimpan data admin, tabel *cities* menyimpan data kota, tabel *criteria* menyimpan data kriteria penilaian, dan tabel *scores* menyimpan nilai kota berdasarkan kriteria.

2.2.2 Relasi Antar Tabel

Relasi antar tabel dirancang dengan konsep *one-to-many*. Satu kota dapat memiliki banyak nilai penilaian, dan satu kriteria dapat digunakan pada banyak nilai. Tabel *scores* berperan sebagai penghubung antara tabel *cities* dan *criteria*, sehingga mendukung proses perhitungan metode Promethee secara fleksibel.

2.3 Perancangan Arsitektur Sistem

2.3.1 Arsitektur *Client–Server*

Aplikasi Island menggunakan arsitektur *client-server*. *Frontend* bertindak sebagai *client* yang menangani tampilan dan interaksi pengguna, sedangkan *backend* bertindak sebagai server yang memproses logika bisnis dan perhitungan metode Promethee.

2.3.2 Penerapan Konsep MVC

Pada sisi backend, sistem menerapkan konsep *Model-View-Controller* (MVC). Model bertanggung jawab terhadap pengelolaan data dan interaksi dengan *database*, *controller* mengelola logika aplikasi dan alur data, sedangkan *view* direpresentasikan oleh *frontend* Vue.js. Penerapan arsitektur ini bertujuan untuk meningkatkan keteraturan kode, memudahkan pengembangan, serta mempermudah proses pemeliharaan sistem di masa mendatang.

BAB III
IMPLEMENTASI DAN EVALUASI SISTEM

3.1 Implementasi *Backend*

Backend aplikasi Island dibangun menggunakan Node.js dengan *framework* Express.js dan menerapkan konsep *Model-View-Controller* (MVC). Struktur folder *backend* disusun secara modular agar kode mudah dipahami, dikembangkan, dan dipelihara.

3.1.1 Struktur Folder *Backend*

Struktur folder *backend* disusun secara terorganisir agar mudah dipahami dan dikembangkan. Pemisahan folder dilakukan berdasarkan fungsi utama, seperti *controller*, *routes*, *middleware*, dan konfigurasi *database*.

Gambar 2 Struktur Folder Backend (MVC)

- config/

  Berisi konfigurasi *database* (database.js) yang mengatur koneksi ke MySQL.

- controllers/

  Berisi logika bisnis aplikasi, seperti autentikasi, pengelolaan kota, kriteria, penilaian, dan perhitungan Promethee.

- middleware/

  Berisi *middleware* untuk validasi input dan autentikasi JWT.
- routes/

  Berisi definisi endpoint API yang menghubungkan *request* ke *controller*.
- app.js

  File utama untuk inisialisasi server Express.

3.1.2 Konfigurasi Environment (.env)

Aplikasi *backend* menggunakan file .env untuk menyimpan konfigurasi sensitif seperti kredensial *database* dan *secret key* JWT. Penggunaan .env bertujuan meningkatkan keamanan sistem dan memisahkan konfigurasi dari kode program.

Variabel penting yang digunakan antara lain:

```
DB_HOST=localhost
DB_USER=root
DB_PASSWORD=********
DB_NAME=db_spk_island
JWT_SECRET=********
```

3.1.3 Middleware Autentikasi JWT

Middleware digunakan untuk melindungi endpoint tertentu agar hanya dapat diakses oleh pengguna yang telah terautentikasi. Sistem menggunakan JSON Web Token (JWT) untuk autentikasi.

Contoh potongan kode middleware:

```
const jwt = require("jsonwebtoken");

const authMiddleware = (req, res, next) => {
      const authHeader = req.headers.authorization;

      if (!authHeader || !authHeader.startsWith("Bearer ")) {
            return res.status(401).json({ msg: "Akses ditolak, token
tidak tersedia" });
      }
      const token = authHeader.split(" ")[1];

      try {
            const decoded = jwt.verify(token, process.env.JWT_SECRET);
            req.user = decoded;
            next();
      } catch (error) {
            res.status(401).json({ msg: "Token tidak valid atau
kadaluarsa" });
      }
};

module.exports = authMiddleware;
```

3.1.4 Implementasi Controller (CRUD)

Controller bertanggung jawab mengelola proses CRUD (*Create, Read, Update, Delete*). Salah satu contoh implementasi ditunjukkan pada *controller* data kota.

Contoh potongan kode controller:

```
const City = require("../models/City");

// GET semua kota
exports.getCities = async (req, res) => {
      const cities = await City.findAll();
      res.json(cities);
};

// CREATE kota
exports.createCity = async (req, res) => {
      const { name, num_coworking, last_updated, image_url } = req.body;
      const city = await City.create({
            name,
            num_coworking,
            last_updated,
            image_url
      });
      res.json(city);
};

// DELETE kota
exports.deleteCity = async (req, res) => {
      const { id } = req.params;
      await City.destroy({ where: { id } });
      res.json({ msg: "Kota berhasil dihapus" });
};
```

3.2 Implementasi *Frontend* (Vue.js 3)

Frontend aplikasi Island dibangun menggunakan Vue.js 3 dengan pendekatan *Single Page Application* (SPA). Vue digunakan untuk mengelola tampilan antarmuka pengguna (admin dan user), mengatur navigasi halaman, serta melakukan komunikasi dengan *backend* melalui REST API.

3.2.1 Routing dan *Protected Route*

Frontend aplikasi memiliki dua bagian utama, yaitu:

1. Halaman Publik (*User*)
    - Home
    - Tools (analisis kota)
    - Tutorial
    - Login Admin

2. Halaman Admin
    - Dashboard
    - Manajemen Kota
    - Manajemen Kriteria
    - Input Nilai Kota
    - Hasil Perankingan
    - Profil Admin

Struktur file frontend disusun secara modular berdasarkan konsep komponen dan view agar mudah dikembangkan dan dipelihara.

3.2.2 Routing Menggunakan Vue *Router*

Navigasi antar halaman dalam aplikasi Island diatur menggunakan Vue *Router*. Vue *Router* memungkinkan aplikasi berpindah halaman tanpa *reload* ulang browser.

Routing dibagi menjadi:

- *Public Route*: dapat diakses tanpa login
- *Protected Route*: hanya dapat diakses oleh admin yang sudah login

Contoh Kode *Routing* (Vue *Router*)

```
import { createRouter, createWebHistory } from "vue-router";
import Login from "../views/Login.vue";
import Dashboard from "../views/admin/Dashboard.vue";
import CityManager from "../views/admin/CityManager.vue";
import { sessionManager } from "../utils/sessionManager";

const routes = [
  { path: "/", component: () => import("../views/public/Home.vue") },
  { path: "/login", component: Login },

  {
    path: "/admin/dashboard",
    component: Dashboard,
    meta: { requiresAuth: true }
  },
  {
    path: "/admin/cities",
    component: CityManager,
    meta: { requiresAuth: true }
  }
];

const router = createRouter({
  history: createWebHistory(),
  routes
});
```

```
// Protected Route
router.beforeEach((to, from, next) => {
  if (to.meta.requiresAuth && sessionManager.isSessionExpired()) {
    next("/login");
  } else {
    next();
  }
});

export default router;
```

Penjelasan:

- meta.requiresAuth menandai halaman admin
- beforeEach digunakan sebagai route guard
- User yang belum login akan diarahkan kembali ke halaman login

3.2.3 Autentikasi dan Proteksi Halaman

Autentikasi admin dilakukan menggunakan JWT (JSON Web Token) yang disimpan di localStorage. Status login diperiksa setiap kali pengguna mengakses halaman admin.

Fungsi: Mengirim data login ke backend dan menyimpan token.

Potongan kode:

```
const handleLogin = async () => {
  const response = await axios.post(
    "http://localhost:3000/api/auth/login",
    {
      username: username.value,
      password: password.value,
    }
  );

  localStorage.setItem("token", response.data.token);
  sessionManager.initSession();
  router.push("/admin/dashboard");
};
```

Penjelasan: Axios digunakan untuk mengirim request HTTP ke backend. Token JWT yang diterima disimpan untuk autentikasi request selanjutnya.

3.2.4 Komponen Manajemen Data (CRUD)

Komponen Manajemen Data (CRUD) merupakan bagian inti dari implementasi frontend aplikasi Island, khususnya pada sisi admin. CRUD merupakan singkatan dari *Create, Read, Update, dan Delete*, yang merepresentasikan operasi dasar dalam pengelolaan data pada sebuah sistem informasi. Pada aplikasi ini, komponen CRUD digunakan untuk mengelola data kota, data kriteria, serta data pengguna admin.

Pada sisi frontend, komponen CRUD dibangun menggunakan Vue.js 3 dengan pendekatan *component-based*. Setiap jenis data memiliki komponen tersendiri sehingga struktur kode menjadi lebih modular, mudah dipahami, dan mudah dikembangkan di masa mendatang.

1. Komponen Data Kota (CityManager.vue)

   Komponen CityManager.vue digunakan untuk mengelola data kota yang menjadi alternatif dalam sistem pendukung keputusan. Pada komponen ini, admin dapat melakukan operasi sebagai berikut:

   a. Menampilkan Data Kota (*Read*)

      Data kota ditampilkan dalam bentuk tabel yang memuat informasi seperti nama kota, jumlah *coworking space*, dan tanggal pembaruan data. Data ini diambil dari *backend* melalui REST API menggunakan metode GET dan ditampilkan secara dinamis menggunakan *reactive state* Vue.

   b. Menambah Data Kota (*Create*)

      Admin dapat menambahkan data kota baru melalui sebuah form yang ditampilkan dalam *modal dialog*. Data yang diinput kemudian dikirim ke *backend* menggunakan metode POST. Sebelum dikirim, sistem melakukan validasi sederhana untuk memastikan seluruh field wajib telah diisi.

   c. Menghapus Data Kota (*Delete*)

      Fitur penghapusan data kota dilakukan dengan metode DELETE. Untuk menjaga keamanan data, sistem menampilkan dialog konfirmasi sebelum proses penghapusan dijalankan.

   d. Integrasi Token JWT

      Seluruh *request* yang berkaitan dengan pengelolaan data kota menyertakan token JWT pada header Authorization. Hal ini memastikan bahwa hanya admin yang telah terautentikasi yang dapat mengakses fitur CRUD.

2. Komponen Data Kriteria (CriteriaManager.vue)

   Komponen CriteriaManager.vue digunakan untuk mengelola kriteria penilaian yang akan digunakan dalam proses pengambilan keputusan. Setiap kriteria memiliki atribut kode, nama, bobot, dan tipe (*benefit* atau *cost*).

Fungsi utama dari komponen ini meliputi:

a. Menampilkan Daftar Kriteria

   Data kriteria ditampilkan dalam bentuk tabel yang memuat kode kriteria, nama, bobot, dan tipe. Tipe kriteria ditampilkan dengan label visual (badge) untuk membedakan antara kriteria *benefit* dan *cost*.

b. Menambahkan Kriteria Baru

   Admin dapat menambahkan kriteria melalui form input. Data yang dimasukkan akan dikirim ke backend menggunakan metode POST. Setiap kriteria memiliki kode unik yang menjadi identitas kriteria dalam sistem.

c. Menghapus Kriteria

   Admin dapat menghapus kriteria tertentu dengan konfirmasi terlebih dahulu. Penghapusan dilakukan menggunakan metode DELETE dan akan langsung memperbarui tampilan tabel setelah data berhasil dihapus.

3. Komponen Input Nilai Kota (*Scoring*)

Selain data kota dan kriteria, aplikasi juga menyediakan fitur untuk mengelola nilai kota berdasarkan kriteria. Fitur ini biasanya terintegrasi dalam komponen manajemen data kota.

Admin dapat:

- Memilih kota tertentu.
- Menginput nilai untuk setiap kriteria.
- Menyimpan nilai ke dalam database melalui REST API.

Nilai-nilai ini kemudian digunakan sebagai input utama dalam proses perhitungan metode Promethee. Dengan adanya komponen ini, proses penilaian kota dapat dilakukan secara terstruktur dan konsisten.

4. Pola Implementasi CRUD pada *Frontend*

Seluruh komponen CRUD pada frontend memiliki pola implementasi yang serupa, yaitu:

a. Menggunakan Axios untuk komunikasi dengan *backend*.
b. Menggunakan ref() atau reactive() untuk menyimpan *state data*.
c. Menggunakan *modal dialog* untuk input data agar tampilan tetap bersih.
d. Menyertakan token JWT pada setiap request yang bersifat *protected*.

e. Memberikan *feedback* visual kepada pengguna berupa dialog atau notifikasi setelah proses berhasil atau gagal.

5. Keterkaitan dengan Sistem Pendukung Keputusan

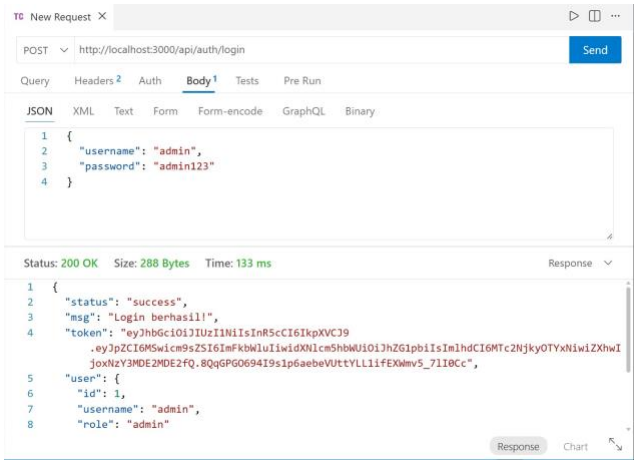
Komponen CRUD tidak hanya berfungsi sebagai pengelola data, tetapi juga berperan penting dalam keseluruhan alur sistem pendukung keputusan. Data kota, kriteria, dan nilai yang dikelola melalui komponen ini menjadi input utama dalam proses analisis dan perankingan menggunakan metode Promethee. Oleh karena itu, keakuratan dan kelengkapan data sangat bergantung pada implementasi komponen CRUD ini.

BAB IV
PENGUJIAN

4.1 Pengujian API

4.1.1 Pengujian Login (Authentication)

Pengujian login dilakukan dengan mengirimkan request POST ke endpoint:



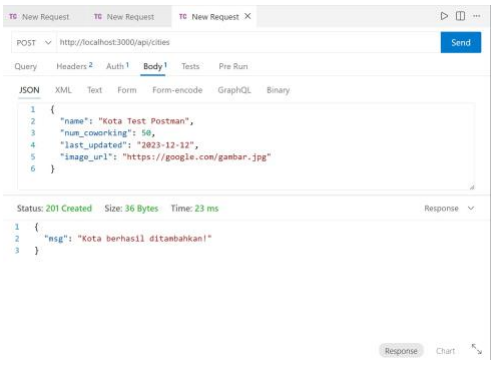
Gambar 3 Pengujian Login (Authentication)

Fitur login berjalan dengan baik dan berhasil menghasilkan token JWT yang digunakan untuk autentikasi endpoint berikutnya. Berdasarkan hasil analisis, didapatkan bahwa:

- Server berhasil memverifikasi kredensial pengguna.
- Sistem mengembalikan status 200 OK.
- Response berisi token JWT dan data user.

4.1.2 Pengujian Tambah Data Kota (POST)

Pengujian penambahan data kota dilakukan menggunakan metode POST ke endpoint:



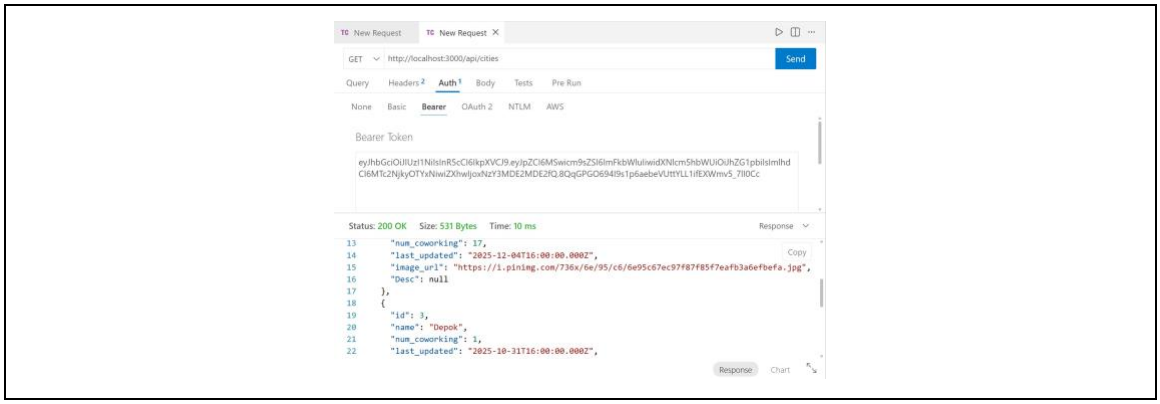
Gambar 4 Pengujian Tambah Data Kota (POST)

Fitur login berjalan dengan baik dan berhasil menghasilkan token JWT yang digunakan untuk autentikasi endpoint berikutnya. Berdasarkan hasil analisis, didapatkan bahwa:

- Server berhasil memverifikasi kredensial pengguna.
- Sistem mengembalikan status 200 OK.
- Response berisi token JWT dan data user.

4.1.3 Pengujian Endpoint Data Kota (GET)

Pengujian dilakukan dengan metode GET pada endpoint:



Gambar 5 Pengujian Endpoint Data Kota (GET)

Fitur penambahan data kota berjalan dengan baik dan data langsung tersimpan ke database. Berdasarkan hasil analisis, didapatkan bahwa:

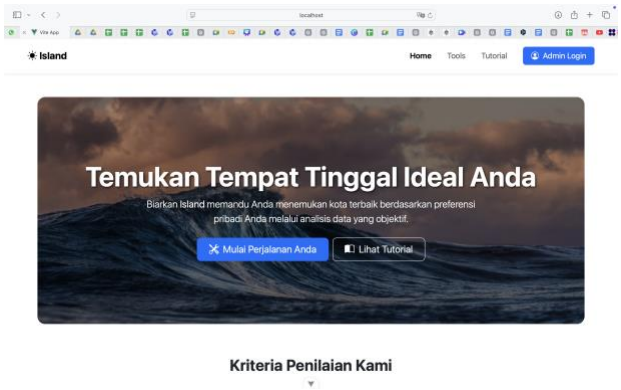
- Data berhasil ditambahkan ke database.
- Server memberikan response 201 Created.

4.2 Pengujian Aplikasi

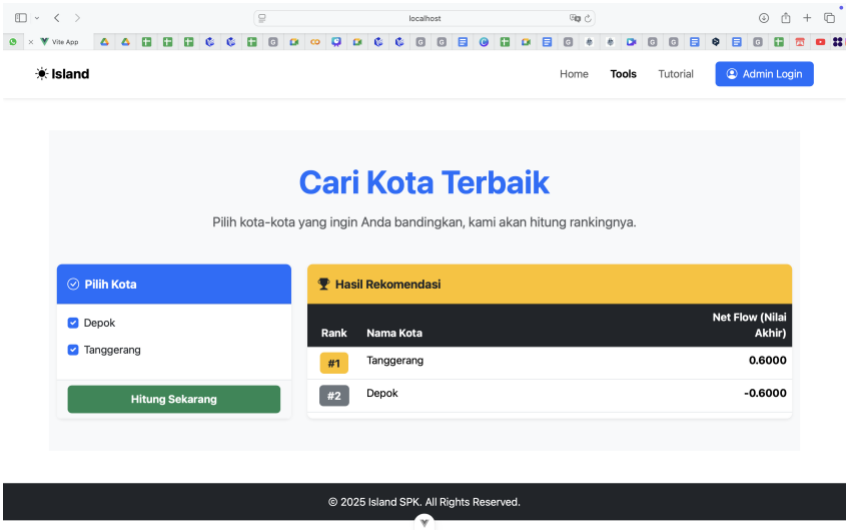
Berikut adalah alur penggunaan aplikasi dari awal hingga menghasilkan rekomendasi kota terbaik:

1. Halaman Awal (Landing Page)

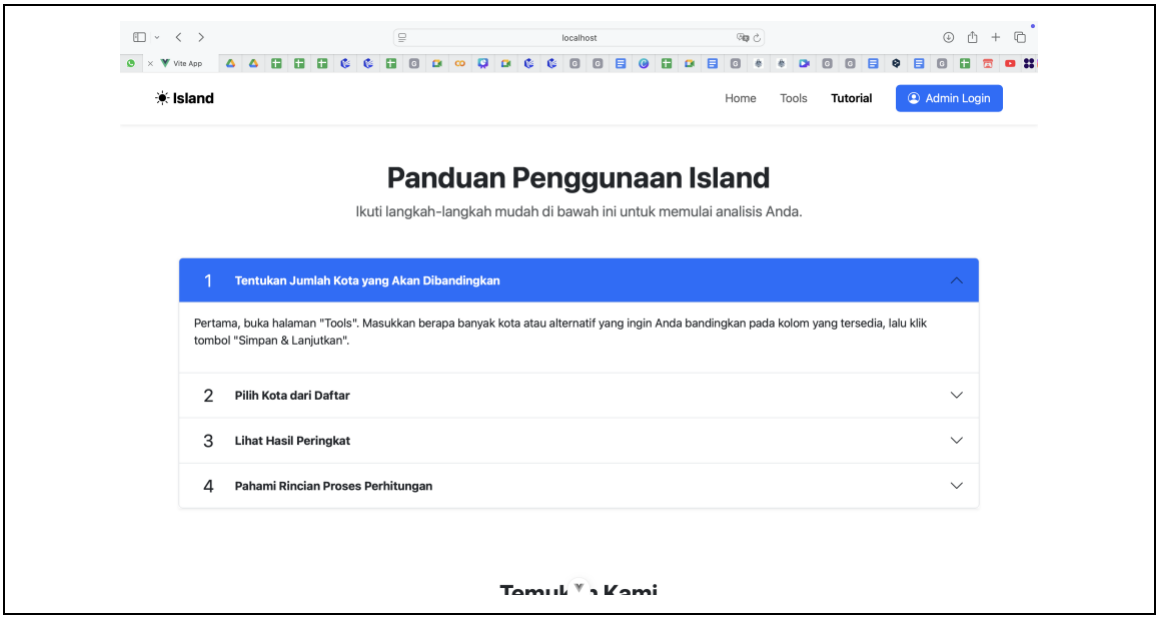
Pengguna pertama kali akan melihat halaman utama aplikasi yang menampilkan informasi singkat tentang sistem serta tujuan aplikasi. Berikut halaman yang tersedia:



Gambar 6 Halaman Dashboard



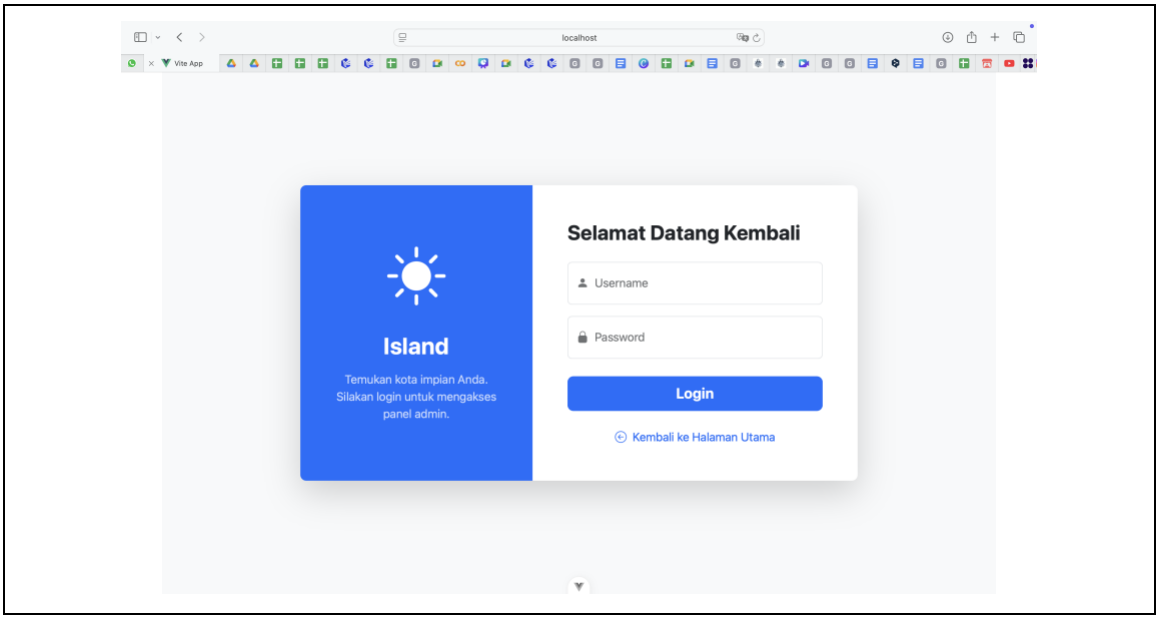
Gambar 7 Halaman Tools



Gambar 8 Halaman Tutorial

2. Halaman Login

Admin melakukan autentikasi dengan memasukkan username dan password. Jika data valid, sistem akan mengembalikan JWT Token dan mengarahkan pengguna ke dashboard admin.

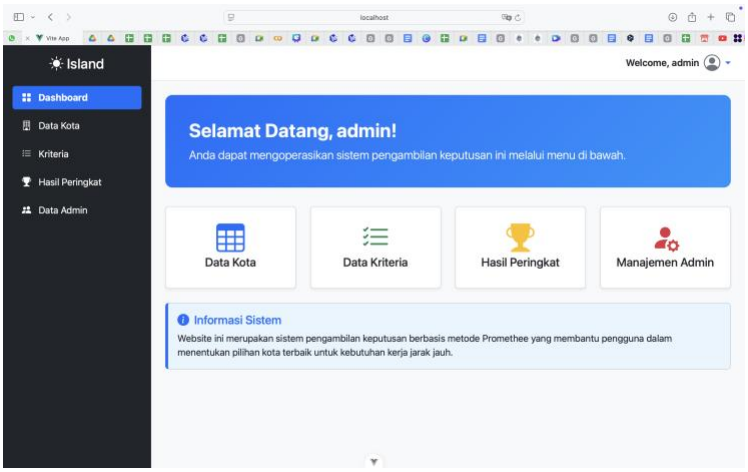


Gambar 9 Halaman Login

3. Dashboard Admin

Setelah login berhasil, admin diarahkan ke halaman dashboard yang menampilkan:

- Ringkasan sistem
- Navigasi ke menu Data Kota, Data Kriteria, Hasil Peringkat, dan Manajemen Admin

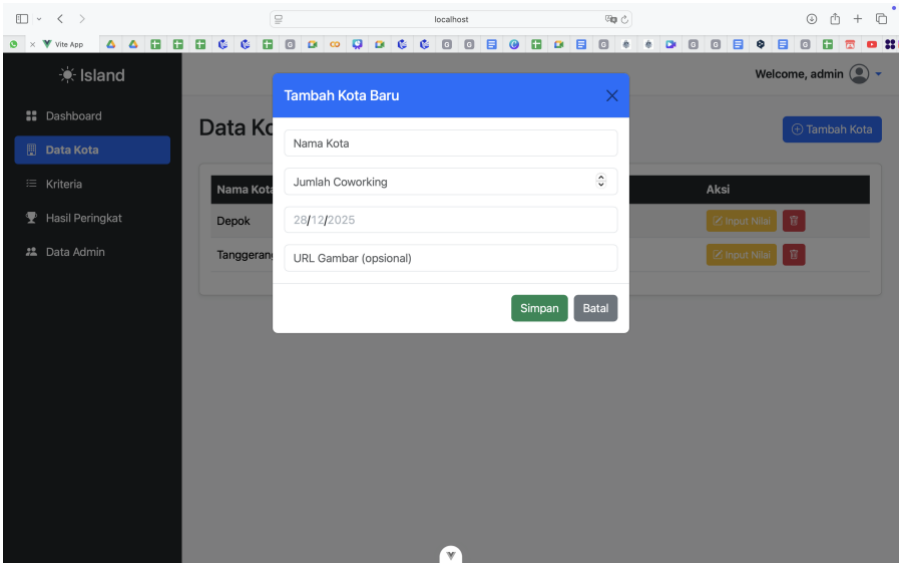


Gambar 10 Halaman Admin

4. Manajemen Data Kota

Admin dapat:

- Menambahkan kota baru
- Mengubah data kota
- Menghapus data kota

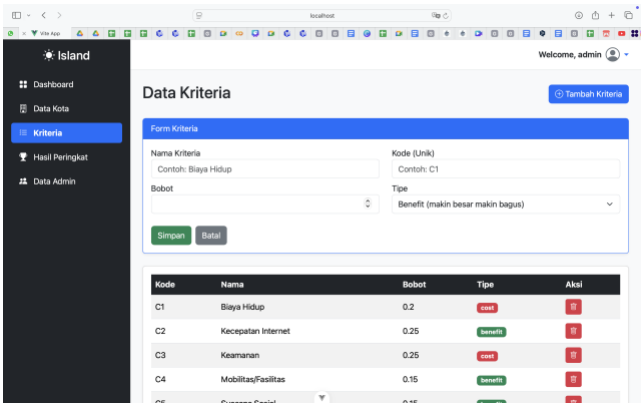


Gambar 11 Halaman Admin – Data Kota

5. Manajemen Data Kriteria

Admin dapat:

- Menambahkan kriteria penilaian
- Menentukan bobot kriteria
- Menentukan tipe kriteria (benefit / cost)

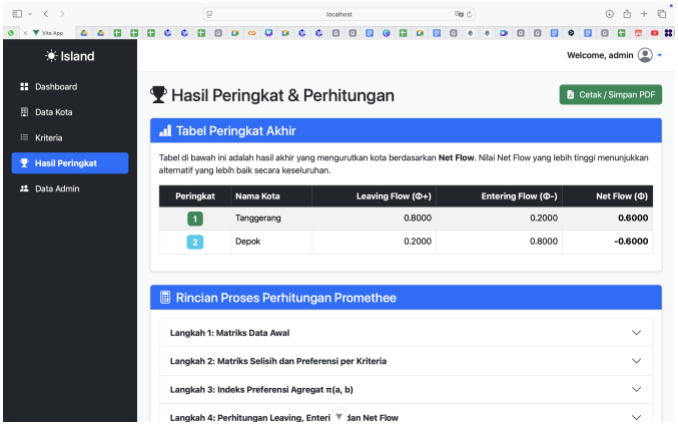


Gambar 12 Halaman Admin – Kriteria

6. Proses Perhitungan dan Hasil Peringkat

Setelah data kota dan kriteria tersedia, sistem akan menghitung:

- Matriks perbandingan
- Nilai preferensi
- Leaving Flow
- Entering Flow
- Net Flow



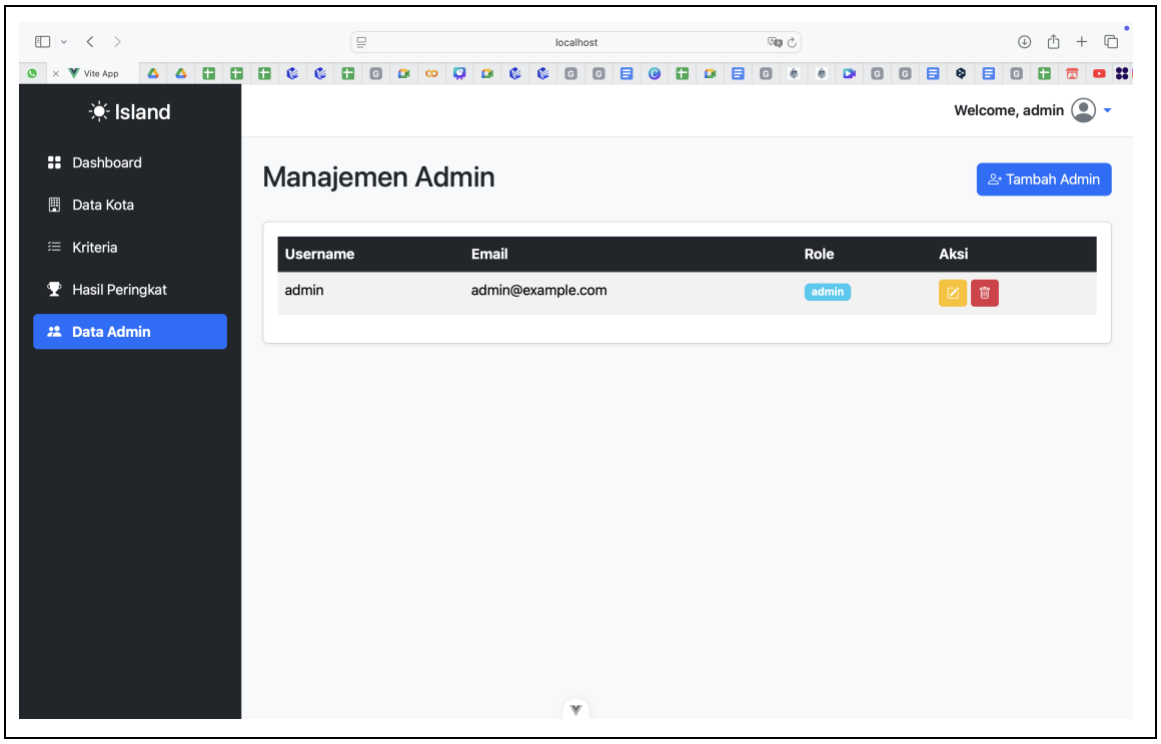
Gambar 13 Halaman Admin – Hasil Peringkat

7. Manajemen Data Admin

Pada halaman Manajemen Admin, sistem menyediakan fitur untuk mengelola akun administrator yang memiliki akses penuh ke aplikasi.

Admin dapat:

- Melihat daftar akun admin yang terdaftar dalam sistem.
- Menambahkan admin baru melalui tombol “Tambah Admin”.
- Mengubah data admin seperti username atau email.
- Menghapus akun admin tertentu apabila tidak lagi digunakan.
- Melihat peran (role) setiap admin yang terdaftar.



Gambar 14 Halaman Admin – Data Admin

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil analisis, perancangan, implementasi, dan pengujian yang telah dilakukan pada bab-bab sebelumnya, dapat ditarik beberapa kesimpulan sebagai berikut:

1. Aplikasi Island berhasil dibangun sebagai sistem pendukung keputusan berbasis web yang mampu membantu pengguna dalam menentukan kota terbaik berdasarkan kriteria tertentu menggunakan metode Promethee.
2. Aplikasi telah mengimplementasikan fitur utama sesuai tujuan yang ditetapkan, tujuan meliputi:
    - Autentikasi admin menggunakan JSON Web Token (JWT).
    - Pengelolaan data kota (CRUD).
    - Pengelolaan data kriteria beserta bobot dan tipe kriteria.
    - Input nilai kota berdasarkan kriteria.
    - Proses analisis dan perankingan kota secara otomatis.
    - Tampilan hasil rekomendasi kota untuk pengguna umum.
3. Dari sisi teknologi, aplikasi ini dibangun menggunakan stack MEVN (MySQL, Express.js, Vue.js, dan Node.js), yang terbukti mampu mendukung pengembangan aplikasi web modern dengan arsitektur *client-server* yang terstruktur.
4. Penerapan arsitektur *Model-View-Controller* (MVC) pada backend berhasil memisahkan logika bisnis, pengelolaan data, dan alur request sehingga kode lebih terorganisir, mudah dipahami, serta mudah dikembangkan.
5. Berdasarkan hasil pengujian API menggunakan Postman, seluruh *endpoint* berjalan dengan baik, termasuk:
    - *Endpoint* autentikasi (login).
    - *Endpoint* CRUD data kota dan kriteria.
    - *Endpoint* input nilai dan analisis Promethee.

5.2 Saran

Untuk pengembangan aplikasi Island ke tahap selanjutnya, terdapat beberapa saran yang dapat dipertimbangkan, antara lain:

1. Pengembangan fitur lanjutan, seperti:
    - Fitur manajemen *user* (*multi-role*: admin dan operator).
    - Fitur visualisasi hasil perankingan dalam bentuk grafik atau *chart*.
    - Fitur pencarian dan filter data yang lebih kompleks.
2. Peningkatan aspek keamanan, misalnya:
    - Implementasi *refresh* token JWT.
    - Pembatasan akses berbasis *role* (*role-based access control*).
3. Optimalisasi performa, terutama pada proses perhitungan ketika jumlah kota dan kriteria semakin besar.
4. Pengembangan ke arah *deployment*, dimana aplikasi dapat:
    - Di-*deploy* menggunakan *Docker* untuk memudahkan distribusi.
    - Dikembangkan menjadi *Progressive Web App* (PWA) agar dapat diakses secara *offline* dan memiliki pengalaman pengguna yang lebih baik.
5. Untuk penelitian selanjutnya, aplikasi ini juga dapat dikembangkan dengan perbandingan metode Sistem Pendukung Keputusan lainnya, seperti SAW, AHP, atau TOPSIS, guna mengetahui metode yang paling optimal untuk kasus pemilihan kota.