WEBSITE RAMAL CUACA BERDASARKAN LOKASI SECARA REAL-TIME

PEMROGRAMAN FUNGSIONAL

Disusun Oleh:

1. **FRIDA SUKMA (1203210098)**
2. **YAYUK AGUSTINA (1203210118)**

PROGRAM STUDI INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM SURABAYA

2024

BAB 1 PENDAHULUAN

1.1. Latar Belakang

Dalam era digital yang serba cepat seperti ini, informasi cuaca menjadi sangat penting bagi banyak orang. Baik itu untuk merencanakan kegiatan sehari-hari, perjalanan, atau bahkan kegiatan luar ruangan. Dengan memahami pentingnya informasi cuaca bagi masyarakat, kami memutuskan untuk mengembangkan aplikasi web ramalan cuaca yang dapat memberikan informasi cuaca secara akurat dan mudah diakses. Pemantauan suhu lokasi secara real-time juga menjadi sangat penting dalam berbagai sektor, seperti sektor kesehatan dan industri. Misalnya, rumah sakit sebagai tempat pelayanan kesehatan harus memastikan kestabilan suhu ruang penyimpanan alat kesehatan, suhu ruangan tempat pasien berada, dan suhu di lingkungan sekitar agar kenyamanan dan kesehatan pasien terjaga [1]. Di sisi lain, dalam sektor industri, pemantauan suhu lokasi dapat membantu mencegah kerugian materi serta memastikan operasional yang aman dan tetap dalam kendali [2].

Beda dengan pemantauan suhu secara tradisional. Pemantauan suhu secara tradisional memiliki banyak kelemahan, seperti alat yang cepat rusak, perhitungan yang kurang akurat, dan membutuhkan waktu yang lebih lama untuk memberikan hasil [3]. Oleh karena itu, dibutuhkan solusi yang lebih efisien, cepat, otomatis, akurat, dan dapat dilakukan secara real-time di berbagai lokasi. Dalam konteks ini, aplikasi web ramalan cuaca yang kami buat memiliki beberapa tujuan utama. Pertama, kami ingin menyediakan informasi cuaca yang akurat dan terkini untuk membantu pengguna dalam merencanakan aktivitas sehari-hari. Dengan informasi cuaca yang tepat waktu, pengguna dapat menghindari kejadian yang tidak diinginkan dan bersiap-siap dengan baik untuk cuaca yang ekstrem. Selain itu, kami juga ingin membuat aplikasi yang mudah digunakan oleh semua kalangan. Oleh karena itu, kami merancang antarmuka pengguna yang sederhana dan intuitif, sehingga pengguna dapat dengan cepat mendapatkan informasi cuaca yang mereka butuhkan tanpa perlu menghabiskan waktu berlama-lama.

Dalam pengembangan aplikasi ini, kami menggunakan teknologi website modern seperti React.JS untuk memastikan performa dan responsivitas yang baik. Website ramal cuaca juga mengintegrasikan layanan API cuaca dari OpenWeatherMap dan OpenMeteo untuk mendapatkan data cuaca yang akurat dan terpercaya. Fitur-fitur utama dari aplikasi ini mencakup informasi cuaca saat ini seperti suhu, kelembaban, kecepatan angin, dan tekanan udara. Website ramal cuaca juga menyediakan ramalan cuaca untuk beberapa hari ke depan agar pengguna dapat merencanakan aktivitas mereka dengan lebih baik. Selain itu, kami juga menyediakan fitur pencarian lokasi dan penggunaan lokasi secara otomatis berdasarkan geolokasi pengguna untuk memudahkan pengguna mendapatkan informasi cuaca di lokasi yang mereka inginkan. Dengan aplikasi ramalan cuaca ini, kami berharap dapat membantu pengguna dalam mengatasi tantangan cuaca sehari-hari dan membuat aktivitas mereka berjalan lebih aman dan nyaman.

1.2. Rumusan Masalah

Berdasarkan uraian latar belakang diatas dapat diidentifikasikan berbagai permasalahan sebagai berikut:

1. Bagaimana cara menyediakan informasi cuaca yang dapat diakses dengan mudah dan cepat oleh pengguna?
2. Bagaimana cara mengintegrasikan fitur pencarian lokasi dan penggunaan geolokasi untuk memberikan informasi cuaca di lokasi yang diinginkan oleh pengguna?

3. Apakah website ramal cuaca dapat menyediakan informasi cuaca yang cukup detail, seperti suhu, kelembaban, kecepatan angin, tekanan udara, dan ramalan cuaca untuk beberapa hari ke depan?
4. Bagaimana cara kerja framework Flask dalam menjalankan API OpenWeatherMap dan OpenMeteo dalam menyediakan informasi cuaca, dan menyimpan riwayat laporan cuaca di hari sebelumnya dan mencatat eror yang terjadi pada website dalam database MySQL?

1.3. Tujuan

Tujuan yang hendak dicapai dalam pelaksanaan dan penulisan ini adalah:

1. Menyediakan informasi cuaca yang mudah diakses dan cepat oleh pengguna, sehingga mereka dapat merencanakan aktivitas sehari-hari, perjalanan, dan kegiatan luar ruangan dengan lebih baik. Hal ini akan membantu pengguna menghindari kejadian yang tidak diinginkan akibat cuaca ekstrem dan mempersiapkan diri dengan lebih baik.
2. Mengintegrasikan fitur pencarian lokasi dan penggunaan geolokasi untuk memungkinkan pengguna mendapatkan informasi cuaca di lokasi yang mereka inginkan dengan mudah. Dengan demikian, pengguna tidak perlu repot-repot mencari informasi cuaca secara manual dan dapat langsung mendapatkan informasi yang relevan dengan lokasi mereka.
3. Menyediakan informasi cuaca yang detail dan akurat, termasuk suhu, kelembaban, kecepatan angin, tekanan udara, dan ramalan cuaca untuk beberapa hari ke depan. Hal ini akan membantu pengguna dalam merencanakan aktivitas mereka dengan lebih baik serta mengantisipasi perubahan cuaca yang akan terjadi.
4. Menyediakan informasi mengenai cara kerja framework Flask dalam menjalankan API cuaca OpenWeatherMap dan OpenMeteo dalam menyediakan informasi cuaca, dan menyimpan riwayat cuaca hari sebelumnya serta melakukan pencatatan eror untuk kemudian disimpan dalam database MySQL.

1.4. Manfaat

Manfaat yang diharapkan dari penulisan ini adalah sebagai berikut:

1. Untuk memenuhi kebutuhan pengguna akan informasi cuaca yang akurat, terkini, dan mudah diakses.
2. Untuk meningkatkan kesiapan dan keamanan dengan adanya informasi cuaca yang akurat dan terkini, sehingga pengguna dapat lebih siap menghadapi perubahan cuaca yang akan datang.
3. Untuk meningkatkan efisiensi. Dengan adanya website Ramal Cuaca, pengguna tidak perlu lagi mencari informasi cuaca secara manual atau bergantung pada alat pemantauan suhu tradisional yang kurang akurat dan lambat.
4. Dapat digunakan sebagai bahan untuk membuat rancangan baru yang lebih akurat, efisien, aman, cepat, dengan teknologi yang lebih terkini. Juga dapat digunakan sebagai dasar dibentuknya keputusan baru.

1.5. Batasan Masalah

Batasan masalah pada pembuatan proposal Website Ramal Cuaca diberikan dengan tujuan, agar pembahasan penelitian hanya dalam lingkup sesuai pokok permasalahan penelitian saja. Sehingga, masalah-masalah dalam penelitian dapat lebih mudah dimengerti dan dipahami. Berdasarkan rumusan masalah, maka guna membuat arah bahasan yang lebih jelas, penelitian ini hanya membahas bagaimana website dapat bekerja dengan baik memberi keterangan suhu dan lokasi yang tepat dan sesuai secara real-time dengan menggunakan React.JS dan bahasa pemrograman JavaScript dan Python.

BAB 2 LANDASAN TEORI

2.1 Pemantauan Suhu

Sesuai dengan yang ada pada Kamus Besar Bahasa Indonesia (KBBI), suhu merupakan besaran yang menyatakan ukuran derajat dingin dan panasnya suatu udara,yang dapat dinyatakan secara kuantitatif dengan satuan derajat tertentu maupun secara kualitatif. Manusia dapat merasakan suhu hanya dengan bantuan kulit manusia itu sendiri, namun rasa dingin dan panas suatu udara yang dapat dirasakan tidaklah sama dan tidak dapat dipastikan seberapa pastinya. Oleh karena itu, dihadirkan alat yang dapat menyatakan suhu secara kuantitatif, agar dapat menghasilkan pengukuran suhu yang pasti dan memberikan persepsi yang sama bagi semua orang. Alat bantu pengukursuhu yaitu termometer. Termometer memiliki berbagai jenis nama dan bentuk yang disesuaikan dengan kegunaannya, terdapat termometer klinis yang digunakan untuk mengukur suhu badan, termometer ruangan untuk mengecek suhu dalam ruangan, termometer inframerah yang dapat mengukur suhu badan tanpa harus menyentuh tubuh, termokopel yang menggunakan sensor termoelektrik, dan masih banyak lagi [4].

Pemantauan suhu udara yang akurat dapat membantu untuk memberikan pemahaman dan identifikasi dini terhadap perubahan cuaca yang signifikan, memprediksi potensi hujan atau potensi bencana alam lainnya, dan memungkinkan untuk memperoleh informasi yang dapat digunakan untuk pengambilan keputusan terkait kenyamanan dan bahkan keselamatan manusia [5]. Contohnya seperti pada aktivitas penerbangan yang membutuhkan informasi mengenai perkiraan cuaca, perkiraan hujan, kekuatan angin, dan lain sebagainya. Juga berbagai aktivitas luar ruangan lainnya yang mengandalkan informasi cuaca.

Perkembangan teknologi pemantauan suhu saat ini terus mengalami peningkatan dikarenakan kebutuhan manusia yang semakin tinggi dan dengan adanya teknologi yang semakin canggih. Saat ini, sensor suhu sudah dapat diintegrasikan melalui teknologi nirkabel seperti WIFI dan Bluetooth, sehingga pengguna dapat mengakses dan memantau perubahan suhu dari berbagai tempat secara real-time. Jugadengan adanya banyak aplikasi dan website yang menyediakan layanan pengecekan suhu yang memanfaatkan radar serta penggunaan API, yang membuat pengecekan suhu ruang menjadi sangat mudah untuk dilakukan.

2.2 Penggunaan API

API merupakan singkatan dari Application Programming Interface, yang merupakan interface yang dapat menghubungkan satu aplikasi dengan aplikasi lainnya. API memungkinkan sebuah website untuk mengembangkan berbagai fitur yang lebih lengkap, fungsional, dan efisien [6]. Pertama-tama aplikasi akan mengakses API, seperti contoh pada aplikasi Traveloka yang dapat mengakses API maskapai penerbangan yang sudah dihubungkan. Selanjutnya setelah aplikasi berhasil mengakses alamat API, API akan melakukan permintaan pada server, contohnya seperti memberitahukan bahwa pihak Traveloka membutuhkan data penerbangan yang akan digunakan pada tanggal dan tujuan yang sudah dimasukkan pengguna. Server akan memberi respon pada API dengan memberi informasi berupa data yang sesuai dengan permintaan. Dan yang terakhir, API akan menyampaikan respon pada aplikasi dan menampilkan detail sesuai permintaan pengguna.

Pada Website Ramal Cuaca Berdasarkan Lokasi Secara Real-Time, memerlukan beberapa API guna menunjang jalannya website yang responsif, tepat, dan efisien. API openweathermap.org dan openmeteo.org yang digunakan untuk mendapatkan informasi mengenai suhu, kelembaban, kecepatan angin, dan tekanan udara. Serta API openstreetmap.org yang digunakan untuk mendapatkan informasi lokasi terkini.

2.3 Framework Flask

Flask merupakan web application framework dari Python yang fleksibel dan ringan untuk digunakan dalam skala *developing web applications*. Flask dirancang untuk melakukan pekerjaan yang cepat dan mudah, dengan menawarkan beberapa *suggestion* tanpa memaksa pengguna untuk menggunakan *dependency* atau *project layout*. Pengguna dapat dengan bebas memilih *tools* dan *library* yang pengguna ingin gunakan [7]. Berikut merupakan beberapa kelebihan dari Flask, antara lain.

1. Flask merupakan *framework backend* yang ringan dengan *dependency* yang minimal.
2. Penggunaan API yang sederhana dan intuitif membuat Flask menjadi *framework* yang mudah untuk dipelajari dan dinilai cocok untuk pemula.
3. Flask adalah *framework* yang fleksibel karena memungkinkan pengguna untuk menyesuaikan dan memperluas *framework* berdasarkan keinginan pengguna.
4. Flask dapat diterapkan di berbagai database seperti SQL dan NoSQL dan dengan teknologi *frontend* apa saja seperti React atau Angular.
5. Flask cocok digunakan untuk proyek-proyek kecil hingga menengah yang tidak membutuhkan kerumitan framework yang besar.

Penggunaan *framework* Flask dalam perancangan website Ramal Cuaca dinilai lebih cocok dan memungkinkan, karena Flask merupakan *framework* yang ringan dan memiliki tingkat kerumitan rendah. Sehingga dinilai cocok diaplikasikan pada perancangan website Ramal Cuaca yang sederhana dan tidak membutuhkan terlalu banyak *library* atau *dependency*.

BAB 3 PERANCANGAN

3.1 Alur Penelitian

Metode yang akan kami gunakan dalam pengembangan aplikasi web ramalan cuaca ini melibatkan beberapa langkah penting. Pertama, kami akan melakukan analisis mendalam terhadap kebutuhan pengguna dan pemahaman yang kuat tentang informasi cuaca yang dibutuhkan dalam berbagai konteks penggunaan. Selanjutnya, kami akan merancang antarmuka pengguna (UI) yang sederhana, intuitif, dan responsif, sehingga pengguna dapat dengan mudah mengakses informasi cuaca yang mereka butuhkan tanpa kesulitan.

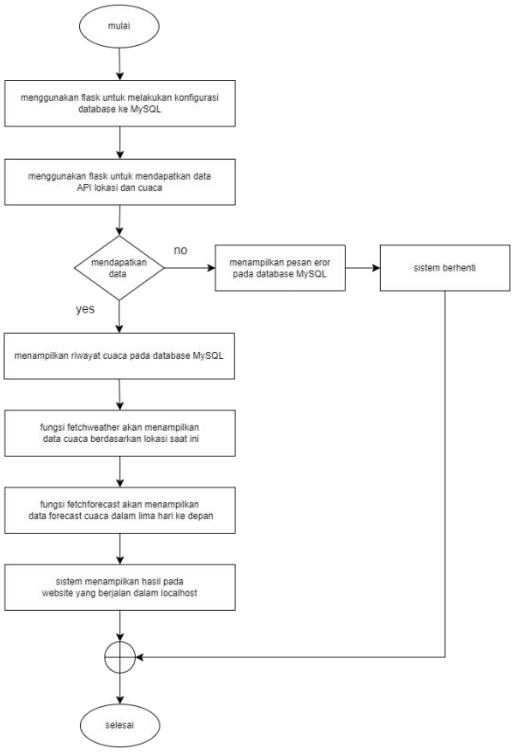
Setelah merancang antarmuka pengguna, kami akan menggunakan teknologi website modern seperti React.JS untuk mengimplementasikan website ini. Penggunaan React.JS akan memastikan performa yang baik dan responsivitas yang cepat dalam mengakses informasi cuaca secara real-time. Selain itu, kami akan mengintegrasikan layanan API cuaca dari OpenWeatherMap dan OpenMeteo untuk mengambil data cuaca yang akurat dan terpercaya.

Proses pengembangan akan melibatkan pengujian berkala untuk memastikan kualitas dan keakuratan informasi cuaca yang disajikan kepada pengguna. Kami juga akan memperhatikan aspek keamanan dan privasi data pengguna dalam pengembangan aplikasi ini, sehingga pengguna dapat merasa aman dan nyaman saat menggunakan aplikasi.

Fitur-fitur utama yang akan dikembangkan termasuk informasi cuaca saat ini seperti suhu, kelembaban, kecepatan angin, dan tekanan udara, serta ramalan cuaca untuk beberapa hari ke depan. Kami juga akan menyediakan fitur pencarian lokasi dan penggunaan geolokasi untuk memudahkan pengguna mendapatkan informasi cuaca di lokasi yang mereka inginkan. Dengan pendekatan ini, kami berharap dapat mengembangkan aplikasi ramalan cuaca yang memenuhi kebutuhan pengguna dengan baik, memberikan informasi cuaca yang akurat dan terkini, serta memberikan pengalaman pengguna yang nyaman dan mudah digunakan.

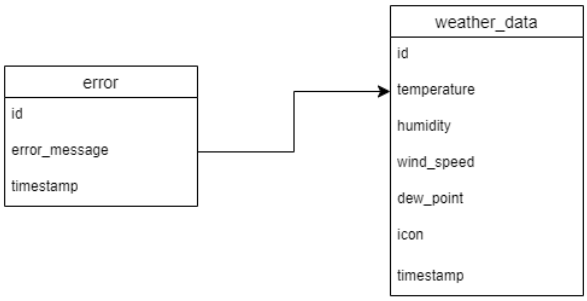
3.2 Flowchart Sistem

Dalam perancangan website Ramal Cuaca Berdasarkan Lokasi Secara Real-Time terdapat dua folder yang dirancang secara terpisah, yaitu satu untuk bagian *coding backend* dan satu untuk bagian *frontend*. Pada bagian *backend*, berisi fungsi-fungsi yang digunakan untuk konfigurasi database ke MySQL serta melakukan pemanggilan API lokasi dan cuaca dengan mengimplementasikan penggunaan *framework* Flask. Selanjutnya, pada bagian *coding frontend*, terdapat fungsi-fungsi yang bertugas untuk menampilkan data-data yang telah dipanggil sebelumnya menggunakan *library* React.JS. Berikut merupakan alur jalannya program Website Ramal Cuaca Berdasarkan Lokasi Secara Real-Time.



3.2 Diagram Database

Berikut merupakan diagram database yang menunjukkan relasi antara tabel error dengan tabel weather_data. Tabel error merupakan tabel yang akan menampilkan informasi error yang terjadi pada website yang sedang berjalan, dan tabel weather_data merupakan tabel yang akan menampilkan hasil data yang telah dipanggil sebelumnya. Tabel error tidak akan menampilkan hasil apapun apabila tidak ditemukan kesalahan dalam sistem, sedangkan tabel weather_data akan terus mencatat hasil selama sistem masih aktif berjalan. Kedua tabel ini berelasi dengan menggunakan primary key ‘id’.



3.3 Implementasi Sistem

Pada Bab implementasi sistem, kami akan menjabarkan mengenai detail penggunaan code program yang digunakan dalam rancangan Website Ramal Cuaca Berdasarkan Lokasi Secara Real-Time.

1. Folder front-end pada file App.js
   Kode program berikut berisi fungsi-fungsi yang akan menampilkan data API cuaca dan lokasi, serta menghandle tombol *refresh* pada laman web Ramal Cuaca Berdasarkan Lokasi Secara Real-Time. Dengan mengimplementasikan penggunaan *library* React.JS.

```
import React, { useEffect, useState, useCallback } from 'react';
import './App.css';
import axios from 'axios';

function App() {
  const [weatherData, setWeatherData] = useState(null);
  const [forecastData, setForecastData] = useState([]);
  const [dateTime, setDateTime] = useState({ date: '', time: '' });
  const [location, setLocation] = useState('');


  const updateDateTime = () => {
    const now = new Date();
    setDateTime({
      date: now.toDateString(),
      time: now.toLocaleTimeString()
    });
  };

  const fetchWeather = async () => {
    try {
      const response = await axios.get(`http://127.0.0.1:5000/get_weather`);
      setWeatherData(response.data);
    } catch (error) {
      console.error('Error fetching weather data:', error);
    }
  };

  const fetchForecast = async () => {
    try {
      const response = await axios.get(`http://localhost:5000/get_forecast`);
      setForecastData(response.data);
    } catch (error) {
      console.error('Error fetching forecast data:', error);
    }
  };
```

```
  const handleGeolocationError = (error) => {
    console.error('Error getting geolocation:', error);
  };

  const showPosition = useCallback(async (position) => {
    const newPosition = { latitude: position.coords.latitude, longitude:
position.coords.longitude };
    try {
      const response = await axios.get(
        `https://nominatim.openstreetmap.org/reverse?format=jsonv2&lat=${newPosi
tion.latitude}&lon=${newPosition.longitude}&accept-language=en`
      );
      const location = response.data.display_name;
      setLocation(location);
      fetchWeather();
      fetchForecast();
    } catch (error) {
      console.error('Error fetching reverse geocoding data:', error);
    }
  }, []);

  const getLocation = useCallback(() => {
    if (navigator.geolocation) {
      navigator.geolocation.watchPosition(showPosition, handleGeolocationError,
{
        enableHighAccuracy: true,
        timeout: 5000,
        maximumAge: 0,
      });
    } else {
      console.log('Geolocation is not supported by this browser.');
    }
  }, [showPosition]);

  useEffect(() => {
    getLocation();
    updateDateTime();
    const intervalId = setInterval(updateDateTime, 1000);
    return () => clearInterval(intervalId);
  }, [getLocation]);

  const handleRefresh = () => {
    window.location.reload();
  };

  if (!weatherData || forecastData.length === 0) {
```

```
    return <div>Loading...</div>;
  }

  return (
    <div className="weather-container">
      <h1>My Weather App</h1>
      <div className="date-time">
        <p id="currentDate">{dateTime.date}</p>
        <div className="clock-container">
          <div className="clock-section">{dateTime.time}</div>
        </div>
      </div>
      <div className="location">
        <p id="selectedCity">{location}</p>
      </div>
      <div className="feels">
        <p id="temperature">{weatherData.temperature}°C</p>
        <p><span className="label">Feels like: </span><span
id="feelsLike">{weatherData.feels_like}°C</span></p>
      </div>
      <div className="icon">
        <img id="icon"
src={`http://openweathermap.org/img/wn/${weatherData.icon}.png`} alt="weather
icon" />
      </div>
      <div className="weather-details">
        <p><span className="label">Pressure: </span><span
id="pressure">{weatherData.pressure} hPa</span></p>
        <p><span className="label">Humidity: </span><span
id="humidity">{weatherData.humidity}%</span></p>
        <p><span className="label">Wind Speed: </span><span
id="windSpeed">{weatherData.wind_speed} m/s</span></p>
        <p><span className="label">Dew Point: </span><span
id="dewPoint">{weatherData.dew_point}°C</span></p>
      </div>
      <div className="forecast-container">
        {forecastData.map((forecast, index) => (
          <div className="forecast-day" key={index}>
            <p className="day">{new
Date(forecast.dt_txt).toLocaleDateString('en-US', { weekday: 'short' })}</p>
            <img src={`http://openweathermap.org/img/wn/${forecast.icon}.png`}
alt="weather icon" />
            <p className="temperature">{Math.round(forecast.temp)}°C</p>
          </div>
        ))}
      </div>
      <button id="refreshWeatherData" onClick={handleRefresh}>
```

```
        Refresh
      </button>
    </div>
  );
}

export default App;
```

2. Folder front-end pada file App.css
   Pada kode program berikut berisi fungsi-fungsi yang membantu penataan tampilan komponen-komponen yang akan ditampilkan pada laman web Ramal Cuaca Berdasarkan Lokasi Secara Real-Time. Dengan mengimplementasikan penggunaan bahasa CSS.

```
.weather-container {
  font-family: Arial, sans-serif;
  color: white;
  background: url('/public/bg4.jpg') no-repeat center center fixed;
  background-size: cover;
  display: flex;
  flex-direction: column;
  align-items: center;
  text-align: center;
  padding: 20px;
  border-radius: 15px;
  width: 100%;
  height: 100vh;
  position: relative;
  box-sizing: border-box;
}

h1 {
  font-size: 4em;
  margin-bottom: 20px;
  position: relative;
}

.today-weather {
  font-size: 1.5em;
  position: absolute;
  top: 10px;
  left: 20px;
  background: rgba(0, 0, 0, 0.5);
  padding: 10px;
  border-radius: 10px;
}
```

```css
.current-temperature {
  font-size: 2em;
  position: absolute;
  top: 60px;
  left: 20px;
  background: rgba(0, 0, 0, 0.5);
  padding: 10px;
  border-radius: 10px;
}

.date-time {
  font-size: 1.5em;
  margin-bottom: 10px;
}

.location {
  font-size: 2em;
  font-weight: bold;
  margin-bottom: 20px;
}

.feels {
  font-size: 2.5em;
  display: flex;
  flex-direction: column;
  align-items: center;
  position: absolute;
  top: 20px;
  right: 20px;
  text-align: right;
}

.feels .label {
  font-size: 0.6em;
}

.icon {
  margin: 20px 0;
  font-size: 3em ;
}

.weather-details {
  font-size: 1.2em;
  margin: 20px 0;
  position: absolute;
  bottom: 20px;
```

```
  left: 20px;
  text-align: left;
}

.weather-details .label {
  font-weight: bold;
}

.clock-container {
  display: flex;
  align-items: center;
  justify-content: center;
  background: rgba(0, 0, 0, 0.5);
  border-radius: 50%;
  padding: 20px;
  font-size: 1.5em;
  margin: 20px 0;
  width: 150px;
  height: 150px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.5);
  text-align: center;
}

.clock-section {
  padding: 0 5px;
}

button#refreshWeatherData {
  padding: 10px 20px;
  font-size: 1em;
  background-color: #333;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  position: absolute;
  bottom: 20px;
  right: 20px;
}

button#refreshWeatherData:hover {
  background-color: #555;
}

/* New CSS for the weather forecast section */
.forecast-container {
  display: flex;
```

```css
  justify-content: space-around;
  width: 100%;
  margin-top: 20px;
}

.forecast-day {
  display: flex;
  flex-direction: column;
  align-items: center;
  background: rgba(0, 0, 0, 0.5);
  padding: 10px;
  border-radius: 10px;
  width: 100px;
  margin: 0 5px;
}

.forecast-day img {
  width: 50px;
  height: 50px;
}

.forecast-day p {
  margin: 5px 0;
}

.forecast-day .day {
  font-weight: bold;
}

.forecast-day .temperature {
  font-size: 1.2em;
}

@media (min-width: 768px) {
  .weather-container {
    width: 100%;
    height: 100vh;
  }
}

@media (max-width: 767px) {
  .weather-container {
    width: 100%;
    height: 100vh;
    padding: 10px;
  }
```

```
  .clock-container {
    width: 150px;
    height: 150px;
    font-size: 1.5em;
  }

  .feels {
    position: static;
    text-align: center;
  }

  .weather-details {
    position: static;
    text-align: center;
  }

  button#refreshWeatherData {
    position: static;
    margin-top: 20px;
  }

  .forecast-container {
    flex-direction: column;
    align-items: center;
  }

  .forecast-day {
    width: 80%;
    margin-bottom: 10px;
  }
}
```

3. Folder back-end pada file app.py
   Pada kode program berikut berisi fungsi-fungsi yang akan memanggil data API cuaca dan lokasi, serta menghandle tampilan dan pemanggilan riwayat dan eror ke database MySQL. Dengan mengimplementasikan penggunaan *framework* Flask.

```
from flask import Flask, jsonify, request
import requests
import pymysql
from flask_cors import CORS

app = Flask(__name__)

CORS(app)
```

```
# Konfigurasi MySQL
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '',  # Ganti dengan password MySQL Anda
    'database': 'weather1_db',
    'charset': 'utf8mb4',
    'cursorclass': pymysql.cursors.DictCursor
}

# Fungsi untuk menghubungkan ke database dan membuat tabel jika belum ada
def create_tables():
    connection = pymysql.connect(**db_config)
    try:
        with connection.cursor() as cursor:
            cursor.execute("""
            CREATE TABLE IF NOT EXISTS error (
                id INT AUTO_INCREMENT PRIMARY KEY,
                error_message TEXT NOT NULL,
                timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            );
            """)
            cursor.execute("""
            CREATE TABLE IF NOT EXISTS weather_data (
                id INT AUTO_INCREMENT PRIMARY KEY,
                temperature FLOAT NOT NULL,
                humidity FLOAT NOT NULL,
                wind_speed FLOAT NOT NULL,
                dew_point FLOAT NOT NULL,
                icon VARCHAR(50) NOT NULL,
                timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
            );
            """)
            connection.commit()
    finally:
        connection.close()

# Fungsi untuk menyimpan log error ke database
def log_error(message):
    connection = pymysql.connect(**db_config)
    try:
        with connection.cursor() as cursor:
            cursor.execute("INSERT INTO error (error_message) VALUES (%s)",
(message,))
            connection.commit()
    finally:
        connection.close()
```

```
# Fungsi untuk menyimpan data cuaca ke database
def save_weather_data(weather_data):
    connection = pymysql.connect(**db_config)
    try:
        with connection.cursor() as cursor:
            cursor.execute("INSERT INTO weather_data (temperature, humidity,
wind_speed, dew_point, icon) VALUES (%s, %s, %s, %s, %s)",
                           (weather_data['temperature'],
weather_data['humidity'], weather_data['wind_speed'], weather_data['dew_point'],
weather_data['icon']))
            connection.commit()
    finally:
        connection.close()

# Fungsi untuk mendapatkan geolokasi berdasarkan IP
def get_ip_geolocation():
    try:
        # Gunakan layanan IP geolocation untuk mendapatkan lokasi berdasarkan IP
        response = requests.get('http://ip-api.com/json/')
        response.raise_for_status()
        data = response.json()
        return {'latitude': data['lat'], 'longitude': data['lon']}
    except requests.exceptions.RequestException as e:
        log_error(str(e))
        return None

# Fungsi untuk mendapatkan ramalan cuaca
def get_weather_forecast(latitude, longitude):
    try:
        api_key = '22b45438c064c52dee7e17bd34c64c0f'
        url =
f'https://api.openweathermap.org/data/2.5/forecast?lat={latitude}&lon={longitude
}&appid={api_key}&units=metric'
        response = requests.get(url)
        response.raise_for_status()
        forecast_data = response.json()['list']
        daily_forecast = [forecast for index, forecast in
enumerate(forecast_data) if index % 8 == 0]
        return daily_forecast
    except requests.exceptions.RequestException as e:
        log_error(str(e))
        return None

# Endpoint untuk mendapatkan ramalan cuaca
@app.route('/get_forecast')
def get_forecast():
```

```
    try:
        # Dapatkan geolokasi berdasarkan IP
        location = get_ip_geolocation()
        if location:
            latitude = location['latitude']
            longitude = location['longitude']

            # Dapatkan ramalan cuaca berdasarkan lokasi
            forecast_data = get_weather_forecast(latitude, longitude)
            if forecast_data:
                return jsonify(forecast_data)
            else:
                return jsonify({'error': 'Gagal mendapatkan ramalan cuaca'})
        else:
            return jsonify({'error': 'Gagal mendapatkan geolokasi'})

    except Exception as e:
        log_error(str(e))
        return jsonify({'error': 'Terjadi kesalahan dalam memproses
permintaan'})

# Endpoint untuk mendapatkan data cuaca
@app.route('/get_weather')
def get_weather():
    try:
        # Ganti YOUR_API_KEY dengan API key OpenWeatherMap Anda
        api_key = '22b45438c064c52dee7e17bd34c64c0f'

        # Dapatkan geolokasi berdasarkan IP
        location = get_ip_geolocation()
        if location:
            latitude = location['latitude']
            longitude = location['longitude']

            # Dapatkan data cuaca berdasarkan lokasi
            url =
f'https://api.openweathermap.org/data/2.5/weather?lat={latitude}&lon={longitude}
&appid={api_key}&units=metric'
            response = requests.get(url)
            response.raise_for_status()

            weather_data = {
                'temperature': response.json()['main']['temp'],
                'humidity': response.json()['main']['humidity'],
                'wind_speed': response.json()['wind']['speed'],
                'dew_point': response.json()['main']['temp'] - ((100 -
response.json()['main']['humidity']) / 5),
```

```
                'icon': response.json()['weather'][0]['icon']
            }

            # Simpan data cuaca ke dalam tabel weather_data
            save_weather_data(weather_data)

            return jsonify(weather_data)

        else:
            return jsonify({'error': 'Gagal mendapatkan geolokasi'})

    except requests.exceptions.RequestException as e:
        # Simpan log error ke dalam database
        log_error(str(e))

        # Kirim respons dengan pesan kesalahan
        return jsonify({'error': 'Tidak tersambung ke internet atau terjadi
kesalahan dalam mengambil data cuaca'})

    except Exception as e:
        log_error(str(e))
        return jsonify({'error': 'Terjadi kesalahan dalam memproses
permintaan'})

# Endpoint untuk mendapatkan log error
@app.route('/logs')
def get_logs():
    try:
        connection = pymysql.connect(**db_config)
        try:
            with connection.cursor() as cursor:
                cursor.execute("SELECT * FROM error")
                logs = cursor.fetchall()
        finally:
            connection.close()
        return jsonify(logs)

    except requests.exceptions.RequestException as e:
        log_error(str(e))
        return jsonify({'error': 'Tidak tersambung ke internet atau terjadi
kesalahan lain'})

    except Exception as e:
        log_error(str(e))
        return jsonify({'error': 'Terjadi kesalahan dalam memproses
permintaan'})
```

```python
if __name__ == '__main__':
    create_tables()
    app.run(debug=True)
```

BAB 4 HASIL DAN PEMBAHASAN

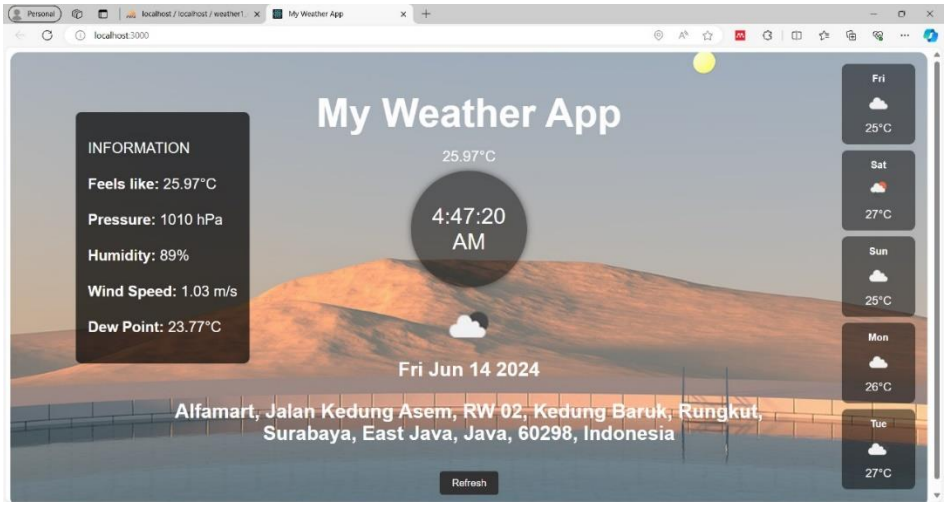
4.1 Jobdesk dan Link GITHUB

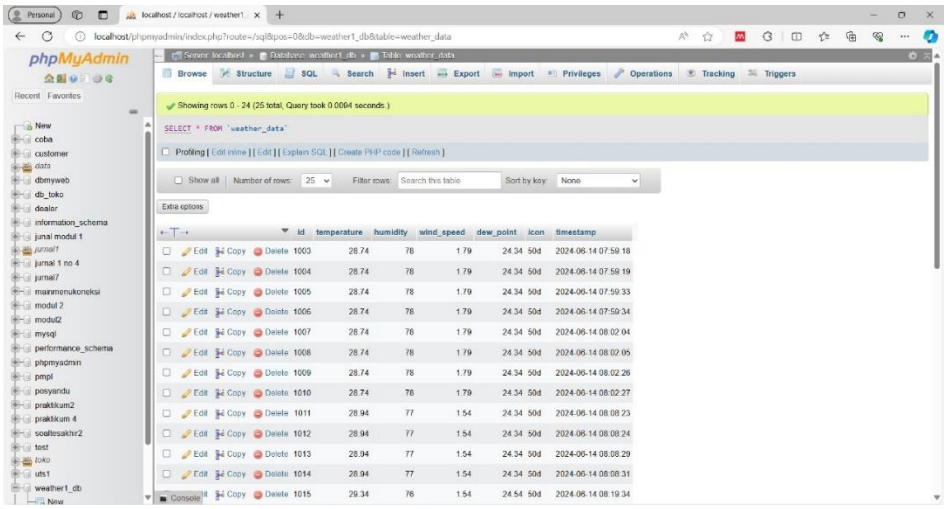
1. Frida Sukma	a. Kontribusi dalam pembentukkan web • Menyusun bagian front-end 50% b. Kontribusi dalam pembuatan laporan • Menyusun bab 1, 2 dan 5.
2. Yayuk Agustina	a. Kontribusi dalam pembentukkan web • Menyusun bagian front-end 50% • Menyusun bagian back-end b. Kontribusi dalam pembuatan laporan • Menyusun bab 3 dan 4.
Link github:	https://github.com/yayukagustd/UAS_PEMFUNG

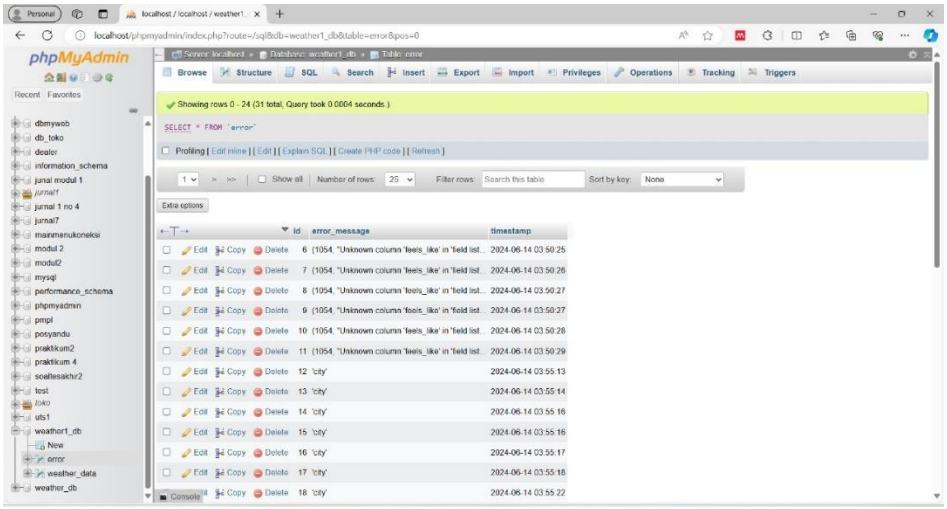
4.2 Penerapan Modul

Pada rancangan Website Ramal Cuaca Berdasarkan Lokasi Secara Real-Time, penulis menerapkan penggunaan modul Flask yang diterapkan pada bagian *coding back-end*. Karena penggunaan modul Flask yang sederhana dan mudah, sehingga mampu untuk membantu pengimplementasian Website Ramal Cuaca Berdasarkan Lokasi Secara Real-Time yang penulis buat.

4.3 Tangkap Layar Hasil Tampilan Web




phpMyAdmin
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Tracking
Triggers
Showing rows 0 - 24 (25 total, Query took 0.0004 seconds.)
SELECT * FROM `weather_data`
id temperature humidity wind_speed dew_point icon timestamp
1003 28.74 78 1.79 24.34 50d 2024-06-14 07:59:18
1004 28.74 78 1.79 24.34 50d 2024-06-14 07:59:19
1005 28.74 78 1.79 24.34 50d 2024-06-14 07:59:33
1006 28.74 78 1.79 24.34 50d 2024-06-14 07:59:34
1007 28.74 78 1.79 24.34 50d 2024-06-14 08:02:04
1008 28.74 78 1.79 24.34 50d 2024-06-14 08:02:05
1009 28.74 78 1.79 24.34 50d 2024-06-14 08:02:26
1010 28.74 78 1.79 24.34 50d 2024-06-14 08:02:27
1011 28.94 77 1.54 24.34 50d 2024-06-14 08:08:23
1012 28.94 77 1.54 24.34 50d 2024-06-14 08:08:24
1013 28.94 77 1.54 24.34 50d 2024-06-14 08:08:29
1014 28.94 77 1.54 24.34 50d 2024-06-14 08:08:31
1015 29.34 76 1.54 24.54 50d 2024-06-14 08:19:34


phpMyAdmin
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Tracking
Triggers
Showing rows 0 - 24 (31 total, Query took 0.0004 seconds.)
SELECT * FROM `error`
id error_message timestamp
6 (1054, "Unknown column 'feels_like' in 'field list... 2024-06-14 03:50:25
7 (1054, "Unknown column 'feels_like' in 'field list... 2024-06-14 03:50:26
8 (1054, "Unknown column 'feels_like' in 'field list... 2024-06-14 03:50:27
9 (1054, "Unknown column 'feels_like' in 'field list... 2024-06-14 03:50:27
10 (1054, "Unknown column 'feels_like' in 'field list... 2024-06-14 03:50:28
11 (1054, "Unknown column 'feels_like' in 'field list... 2024-06-14 03:50:29
12 'city' 2024-06-14 03:55:13
13 'city' 2024-06-14 03:55:14
14 'city' 2024-06-14 03:55:16
15 'city' 2024-06-14 03:55:16
16 'city' 2024-06-14 03:55:17
17 'city' 2024-06-14 03:55:18
18 'city' 2024-06-14 03:55:22

BAB 5 PENUTUP

Pada rancangan Website Ramal Cuaca Berdasarkan Lokasi Secara Real-Time, penulis menerapkan penggunaan modul Flask yang dianggap mudah dan ringan untuk mengimplementasikan rancangan web yang penulis buat. Dengan bahasa pemrograman Javascript dan Python, membuat web dapat diintegrasikan sesuai dengan harapan penulis. Serta dengan penggunaan database MySQL yang digunakan untuk penyimpanan riwayat cuaca dan eror, membantu mempermudah pengguna untuk mendapatkan informasi mengenai riwayat cuaca yang telah lampau, untuk kemudian dapat digunakan sebagai bahan penelitian atau untuk keperluan lain sesuai dengan keinginan pengguna. Pencatatan eror akan membantu mempermudah penulis menemukan kesalahan saat terjadi eror pada laman web Ramal Cuaca.

Dibentuknya rancangan Website Ramal Cuaca Berdasarkan Lokasi Secara Real-Time diharapkan dapat membantu mempermudah pengguna dalam mendapatkan informasi cuaca yang cepat, detail, dan akurat. Dengan begitu, pengguna dapat menjalankan aktivitasnya dengan maksimal dan dapat melakukan perencanaan untuk kegiatan mendatang yang memerlukan informasi cuaca.

DAFTAR PUSTAKA

[1] saka.co.id, "Monitoring Suhu dan Kelembaban Ruangan di Rumah Sakit," 2021.

[2] microthings.id, "Pemantauan Suhu dan Kelembaban di Lingkungan Industri," 2023.

[3] CoWorking, "coworking.co.id," January 2021. [Online]. Available: https://www.coworking.co.id. [Accessed June 2024].

[4] K. N, "Pengertian Suhu: Rumus, Faktor, Alat Ukur dan Skala," *Gramedia.com.*

[5] S. K. Industri, "Fungsi dan Manfaat Weather Station (Stasiun Cuaca)," 2023.

[6] A. Lawrence, "API: Pengertian, Fungsi, dan Cara Kerjanya," *niagahoster.co.id,* 2020.

[7] g. f. geeks, "geeksforgeeks.org," 4 April 2024. [Online]. Available: https://www.geeksforgeeks.org/python-introduction-to-web-development-using-flask/. [Accessed 17 June 2024].