

LAPORAN FINAL PROJECT

PENGUJIAN API MENGGUNAKAN OPEN WEATHER MAP

API DENGAN POSTMAN

PENGUJIAN DAN PENJAMINAN MUTU PERANGKAT LUNAK



Kelompok 09

11S22004 Pangeran Simamora
11S22016 Fretty Debora Sirait
11S22018 Deak Marujar Napitupulu
11S22026 Riovan Samuel Sihombing
11S22033 Chalvin Eric Melkishaer Sihombing
11S22041 Viktris Maria Kristriani Lubis

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA DAN TEKNIK

ELEKTRO

INSTITUT TEKNOLOGI DEL

Pada laporan ini, kelompok kami membuat dua versi pengujian di laptop yang berbeda, yaitu pengujian terhadap Jakarta dan Tokyo.

Link Github kota jakarta: [frettyy/Weather-Project](https://github.com/frettyy/Weather-Project)

Link Github kota tokyo: [viktrislubis/kelompok9-open-weather-api-testing](https://github.com/viktrislubis/kelompok9-open-weather-api-testing)

A. DESKRIPSI API

Proyek ini menggunakan API dari OpenWeatherMap untuk menguji berbagai fitur yang ditawarkan, khususnya pada beberapa endpoint yang menyediakan informasi cuaca.

Pengujian dilakukan untuk memastikan data yang diterima dari API valid dan berfungsi sebagaimana mestinya.

Tiga Endpoint yang Digunakan

1. Data Cuaca Terkini (Current Weather Data) (

https://api.openweathermap.org/data/2.5/weather?q={nama_kota}&appid={API_key})

Endpoint ini berfungsi untuk memperoleh informasi cuaca saat ini berdasarkan nama kota yang diberikan. Informasi yang dapat diperoleh meliputi: Kondisi cuaca, contohnya seperti "langit cerah" atau "hujan ringan", suhu, tekanan dan kelembapan udara, kecepatan dan arah angin., zona waktu.

Dengan parameter yang digunakan adalah sebagai berikut

- a. q: Nama kota yang dicari, dapat disertai kode negara (contoh: q=Jakarta,id).
- b. appid: API Key untuk autentikasi.

2. Prakiraan Cuaca (Weather Forecast Data) (

https://api.openweathermap.org/data/2.5/forecast?q={nama_kota}&appid={API_key})

Endpoint ini menyediakan prakiraan cuaca untuk 5 hari ke depan, dengan pembaruan setiap 3 jam. Informasi yang tersedia meliputi: perkiraan suhu setiap interval, kondisi cuaca, seperti kemungkinan hujan atau berawan, data kelembapan dan angin di setiap interval waktu.

Dengan parameter yang digunakan adalah sebagai berikut

- a. q: Nama kota.
- b. appid: API Key.

3. Cuaca Berdasarkan Koordinat (Weather by Coordinates) (

https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API_key})

Endpoint ini menyediakan informasi cuaca terkini berdasarkan koordinat geografis tertentu. Data yang tersedia serupa dengan endpoint Current Weather Data, namun data difokuskan pada lokasi spesifik.

Dengan parameter yang digunakan adalah sebagai berikut

- a. lat: Garis lintang lokasi.
- b. lon: Garis bujur lokasi.
- c. appid: API Key

B. METODE PENGUJIAN

1. Pengujian Manual (sesuai modul)

Pengujian dilakukan secara manual menggunakan alat atau aplikasi pengujian API, seperti Postman. Tahapan yang dilakukan meliputi:

a. Pengujian Fungsional API

Melakukan pengiriman request langsung ke API untuk memastikan bahwa sistem memberikan respons yang sesuai berdasarkan data input. Pengujian ini mencakup skenario seperti:

- 1. Input yang valid (misalnya, nama kota yang benar).
- 2. Input yang tidak lengkap (parameter yang kurang).
- 3. Input yang tidak valid (data yang salah format atau tidak sesuai).

b. Verifikasi Respons API

Meninjau data yang diterima dari server untuk memastikan status kode respons sesuai ekspektasi (contoh: 200 untuk permintaan berhasil, 400 untuk permintaan yang salah, atau 500 untuk kesalahan server). Struktur dan format data dalam respons memenuhi spesifikasi yang diharapkan.

c. Pengujian Keamanan API

Mencoba mengakses API tanpa autentikasi atau dengan token yang salah untuk memvalidasi bahwa akses hanya diberikan jika autentikasi atau otorisasi valid. Hal ini penting untuk melindungi data dan fungsi API dari akses yang tidak sah.

d. Pengujian Penanganan Error

Menguji API dengan mengirimkan data yang dapat memicu kesalahan, seperti parameter dengan tipe data yang salah atau input yang tidak sesuai. Tujuannya adalah untuk memastikan bahwa API menangani skenario error dengan memberikan pesan atau respons yang jelas dan informatif.

2. Pengujian Otomatis dengan GitHub Actions

Metode ini menggunakan GitHub Actions untuk menjalankan pengujian API secara otomatis dalam proses Continuous Integration (CI). Penggunaan GitHub Actions memastikan bahwa setiap perubahan pada kode, seperti push atau pull request, akan secara otomatis memicu pengujian API. Langkah-langkahnya meliputi:

a. Otomatisasi Pengujian Fungsional API

Menyusun skrip untuk menguji berbagai skenario API, termasuk pengujian input valid, input tidak valid, dan penanganan error. Skrip ini dijalankan secara otomatis oleh GitHub Actions setiap kali ada pembaruan kode.

b. Deteksi Cepat Terhadap Kesalahan

Dengan integrasi CI, kesalahan atau regresi dapat ditemukan segera setelah perubahan kode dilakukan, sehingga tim dapat memperbaikinya lebih cepat.

c. Pengujian Berulang yang Konsisten

Proses pengujian otomatis ini menjamin konsistensi karena pengujian dilakukan dengan prosedur yang sama pada setiap iterasi pengembangan.

d. Laporan Hasil Pengujian

GitHub Actions menghasilkan laporan pengujian yang dapat digunakan untuk menganalisis hasil dan memperbaiki kesalahan yang ditemukan selama pengujian. Dengan kombinasi pengujian manual dan otomatis, pengujian API menjadi lebih menyeluruh, efisien, dan andal, memastikan kualitas serta fungsionalitas API yang optimal dalam berbagai kondisi.

C. PENGUJIAN DI POSTMAN

Kota Jakarta

1. Pengujian API dengan Postman:

Current Weather Data: <https://api.openweathermap.org/data/2.5/weather>

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like '11S22033 Movies API', 'Books API', 'Delcom Public API', and 'PPMPL'. Under 'PPMPL', there's a 'GET Cuaca Saat Ini' item. The main workspace shows a 'PPMPL / Cuaca Saat Ini' collection. A 'GET' request is selected with the URL 'https://api.openweathermap.org/data/2.5/weather?q=Jakarta&appid=e9ae1e0a2ccf3cf0882a104f5516735'. The 'Params' tab shows 'q' set to 'Jakarta' and 'appid' set to 'e9ae1e0a2ccf3cf0882a104f5516735'. The 'Body' tab is selected, showing a JSON response in 'Pretty' format. The response contains weather data for Jakarta, including coordinates, current temperature, humidity, pressure, wind speed, and sunrise/sunset times. The status bar at the bottom indicates it's an 'Online' session with a 22°C temperature and a battery level of 09:48 on 02 December.

Pada tangkapan layar diatas menunjukkan penggunaan Postman untuk melakukan request data cuaca dari API OpenWeatherMap. Dengan query parameter q=Jakarta (nama kota) dan appid sebagai API Key untuk autentikasi. Metode HTTP yang dipakai adalah GET, yang berhasil memberikan respons dengan status code 200 OK, menunjukkan bahwa permintaan berhasil. Respons API menampilkan informasi cuaca Jakarta dalam format JSON, seperti suhu saat ini (22°C), kondisi berawan, persentase awan (20%), serta waktu matahari terbit dan terbenam. Data tambahan, seperti zona waktu Jakarta (+7 GMT) dan kode negara (ID), juga disertakan. Ini menggambarkan bagaimana Postman digunakan untuk menguji integrasi API dan memvisualisasikan data secara efektif.

Weather Forecast Data: <https://api.openweathermap.org/data/2.5/forecast>

The screenshot shows the Postman application interface. In the left sidebar, there's a collection named 'PPMPL' containing a 'Weather Forecast' endpoint. The main area shows a GET request to 'https://api.openweathermap.org/data/2.5/forecast?q=Jakarta&appid=e9ae1e0a2ccf3cf0882a104f516735'. The 'Params' tab is selected, showing query parameters: 'q' (value: Jakarta) and 'appid' (value: e9ae1e0a2ccf3cf0882a104f516735). Below the params, the 'Body' tab is selected, displaying the JSON response in 'Pretty' format. The response includes a timestamp ('dt'), main weather information ('main' object with temp, feels_like, temp_min, temp_max, pressure, sea_level, and grnd_level), and a weather description ('weather' array with id 500). The status bar at the bottom indicates a 200 OK response with a duration of 4.42 s and a size of 16.47 KB.

Pada tangkapan layar diatas menunjukkan penggunaan Postman untuk mengakses API OpenWeatherMap pada endpoint Weather Forecast dengan metode GET. Dengan query parameter q=Jakarta untuk kota Jakarta dan appid sebagai API key untuk autentikasi. Tujuan request ini adalah untuk mendapatkan data prakiraan cuaca dalam beberapa periode mendatang untuk kota tersebut.

Respons API berhasil dengan status kode 200 OK dan memberikan data cuaca dalam format JSON. Data yang ditampilkan meliputi informasi suhu, kelembaban, tekanan udara, serta deskripsi cuaca pada waktu tertentu. Contohnya, suhu yang dirasakan (feels_like) adalah 304,45 Kelvin, kelembaban 64%, dan tekanan udara di permukaan laut sebesar 1008 hPa. Selain itu, respons juga mencantumkan waktu prakiraan dalam format timestamp UNIX (dt) untuk mengindikasikan kapan data cuaca tersebut berlaku.

Postman memvisualisasikan data ini dalam tab Body dengan format yang mudah dibaca (Pretty). Koleksi ini juga tampaknya dipisah untuk beberapa jenis request seperti Current Weather dan Weather Forecast, menjadikannya contoh pengelolaan API yang terorganisir dengan baik untuk tujuan pengujian atau integrasi aplikasi.

Weather by Coordinates (Latitude/Longitude):

<https://api.openweathermap.org/data/2.5/weather>

The screenshot shows the Postman application interface. In the top navigation bar, 'Home' and 'Workspaces' are visible. The main workspace is titled 'PPMPL / Weather by Coordinates'. A GET request is being made to the URL <https://api.openweathermap.org/data/2.5/weather?lat=-6.2146&lon=106.8451&appid=e9ae1e0a2ccf3cf0882a104f5516735>. The 'Params' tab shows four parameters: 'lat' (-6.2146), 'lon' (106.8451), and 'appid' (e9ae1e0a2ccf3cf0882a104f5516735). The 'Body' tab displays the JSON response:

```
1 {
2     "coord": {
3         "lon": 106.8451,
4         "lat": -6.2146
5     },
6     "weather": [
7         {
8             "id": 801,
9             "main": "Clouds",
10            "description": "few clouds",
11            "icon": "02d"
12        }
13    ],
14    "base": "stations",
15    "main": {
16        "temp": 305.17,
17        "feels_like": 310.27,
18        "temp_min": 303.91
19    }
20}
```

Pada tangkapan layar diatas menunjukkan penggunaan Postman untuk mengakses API OpenWeatherMap pada endpoint Weather by Coordinates dengan metode GET. Dengan parameter query lat=-6.2146 dan lon=106.8451, yang menunjukkan koordinat Jakarta, serta appid sebagai API key untuk autentikasi. Respons API dengan status kode 200 OK menampilkan data cuaca terkini dalam format JSON, termasuk koordinat lokasi (lon=106.8451 dan lat=-6.2146), deskripsi cuaca "few clouds" (sedikit berawan), suhu saat ini 305.17 Kelvin (sekitar 32°C), suhu terasa 310.27 Kelvin (sekitar 37°C), tekanan udara 1013 hPa, dan kelembaban 66%. Data ini juga mencakup ikon cuaca (02d) yang menunjukkan kondisi visual. Postman mempermudah pengujian API dengan memberikan hasil yang terstruktur dan terorganisir, sehingga memudahkan analisis data berdasarkan lokasi tertentu.

2. Menggunakan Environment Variable di Postman:

Current Weather Data:

The screenshot shows the Postman application interface. In the left sidebar, there's a 'Collections' section with 'Books API', 'Delcom Public API', and 'Jakarta'. Under 'Jakarta', three items are listed: 'GET Current Weather' (selected), 'GET Weather Forecast', and 'GET Weather by Coordinates'. The main workspace shows a 'Jakarta / Current Weather' collection with a single 'GET Current Weather' request. The request URL is https://api.openweathermap.org/data/2.5/weather?{{Kota}}&{{API Key}}. The 'Params' tab is selected, showing two environment variables: 'Key' and '{{Kota}} {{API Key}}'. The 'Body' tab contains a JSON object with fields like 'coord', 'weather', and 'base'. The 'Headers' tab shows 'Content-Type: application/json'. The 'Test Results' tab displays a successful response with status 200 OK, duration 114 ms, and size 850 B. The response body is a JSON object representing the current weather in Jakarta.

Pada tangkapan layar diatas memperlihatkan penggunaan aplikasi Postman untuk mengirimkan permintaan (request) ke API OpenWeatherMap guna mendapatkan data cuaca terkini di Jakarta. Permintaan ini menggunakan dua parameter utama, yaitu `{{Kota}}`, yang kemungkinan diisi dengan Jakarta, dan `{{API Key}}`, yang diperlukan untuk autentikasi. Setelah permintaan dikirim, API merespons dengan status 200 OK, menandakan bahwa permintaan berhasil diproses.

Respons API ditampilkan dalam format JSON, yang memuat detail cuaca di Jakarta. Koordinat lokasi menunjukkan longitude 106.8451 dan latitude -6.2146. Informasi cuaca menggambarkan kondisi mendung (overcast clouds) dengan kategori Clouds dan ikon terkait 04d. Suhu udara tercatat 303.29 K (sekitar 30,14°C) dengan feels like atau suhu yang terasa mencapai 306.46 K (sekitar 33,31°C) akibat kelembaban yang berada pada angka 61%. Selain itu, tekanan udara tercatat 1006 hPa, dengan temperatur minimum dan maksimum masing-masing 301.90 K (28,75°C) dan 303.31 K (30,16°C). Data ini juga mencakup tekanan udara di level laut dan level tanah yang hampir sama, yaitu 1006 hPa dan 1005 hPa.

Weather Forecast Data: <https://api.openweathermap.org/data/2.5/forecast>

The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for Home, Workspaces, API Network, and a search bar labeled "Search Postman". Below the navigation is a header for "Jakarta / Weather Forecast" with a "GET" method and the URL "https://api.openweathermap.org/data/2.5/forecast?({Kota}) ({API Key})". The "Params" tab is selected, showing two parameters: "Key" and "({Kota}) ({API Key})". The "Body" tab is also visible. On the right side, there are buttons for "Send", "Save", and "Share". Below the main request area, there is a "Test Results" section with a "Pretty" JSON view of the response. The response code is 200 OK, and the JSON data includes details like cod, message, cnt, and a list of weather forecasts. At the bottom of the screen, there is a toolbar with various icons and a status bar showing "27°C Berawan".

Pada tangkapan layar diatas menunjukkan penggunaan aplikasi Postman untuk mengakses data prakiraan cuaca melalui API OpenWeatherMap dengan endpoint menggunakan metode GET. Permintaan ini melibatkan dua parameter utama, yaitu `{Kota}` untuk menentukan lokasi (misalnya, "Jakarta") dan `{API Key}` untuk autentikasi. Respons API berhasil dengan status 200 OK dan menghasilkan data dalam format JSON yang berisi informasi prakiraan cuaca untuk beberapa waktu mendatang.

Data tersebut terorganisasi dalam elemen list, yang memuat serangkaian objek dengan rincian kondisi cuaca untuk setiap waktu tertentu. Salah satu data prakiraan menunjukkan suhu udara sebesar 303.09 K (sekitar 30°C), dengan suhu yang terasa mencapai 306.69 K (sekitar 33,54°C), kelembapan sebesar 64%, dan tekanan udara 1007 hPa. Cuaca pada waktu tersebut digambarkan sebagai "scattered clouds" (awan tersebar) dengan ID kategori 802.

Informasi tambahan juga mencakup tekanan udara di level laut dan level tanah, yang keduanya tercatat pada 1007 hPa. Pengaturan parameter yang dinamis di Postman memungkinkan pengujian API yang fleksibel, menghasilkan data yang detail dan bermanfaat untuk memprediksi kondisi cuaca di lokasi tertentu.

Weather by Coordinates (Latitude/Longitude):

<https://api.openweathermap.org/data/2.5/weather>

The screenshot shows the Postman application interface. In the top navigation bar, there are tabs for Home, Workspaces, API Network, and a search bar labeled 'Search Postman'. Below the navigation is a sidebar titled 'My Workspace' containing sections for Collections, Environments, and History. Under 'Environments', 'Jakarta' is selected, and under it, three requests are listed: 'GET Current Weather', 'GET Weather Forecast', and 'GET Weather by Coordinates'. The 'GET Weather by Coordinates' request is currently active, indicated by a red dot next to its name.

The main workspace shows a 'GET' request to the URL `https://api.openweathermap.org/data/2.5/weather?({{latlong}})({{API Key}})`. The 'Params' tab is selected, showing two parameters: 'Key' and '({{latlong}})({{API Key}})'. The 'Body' tab displays the JSON response received from the API. The status bar at the bottom right shows '200 OK' with a response time of '268 ms' and a size of '842 B'.

Below the JSON response, the Windows taskbar is visible, showing various pinned icons like File Explorer, Edge, and Google Chrome. The system tray shows the date and time as '29/11/2024 16:08'.

Pada tangkapan layar diatas menunjukkan penggunaan Postman untuk melakukan permintaan API Weather by Coordinates dari OpenWeatherMap. Permintaan ini menggunakan parameter koordinat geografis `{{latlong}}` untuk menentukan lokasi (longitude dan latitude) serta `{{API Key}}` untuk autentikasi API. Berdasarkan respons JSON yang ditampilkan, permintaan berhasil diproses dengan status 200 OK.

Respons menunjukkan detail cuaca untuk lokasi dengan koordinat longitude 106.066 dan latitude -6.1751. Pada bagian cuaca, kondisi utama yang terdeteksi adalah "Clouds" (berawan) dengan deskripsi "few clouds" (beberapa awan) dan ikon yang sesuai 02d. Data suhu menunjukkan bahwa suhu udara saat ini adalah 304.18 K (sekitar 31,03°C), dengan suhu yang terasa (feels like) mencapai 308.57 K (sekitar 35,42°C) akibat kelembaban yang berada pada 62%. Selain itu, terdapat data mengenai tekanan udara sebesar 1006 hPa, dengan nilai tekanan pada level laut 1006 hPa dan pada level tanah 1004 hPa. Temperatur minimum dan maksimum tercatat masing-masing 302.04 K (28,89°C) dan 304.25 K (31,1°C).

Pengaturan parameter dinamis di Postman memungkinkan fleksibilitas untuk mengakses data cuaca di berbagai lokasi berdasarkan koordinat. Respons yang diberikan cukup terperinci dan dapat digunakan untuk berbagai kebutuhan, seperti perencanaan aktivitas atau analisis cuaca lokal.

3. Pengujian Error Handling:

q = Jakarta diganti menjadi q = Parongil

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists several collections, including '11S22033 Movies API' and 'PPMPL'. The main workspace displays a 'Current Weather' endpoint from 'PPMPL'. The request URL is `https://api.openweathermap.org/data/2.5/weather?q=Parongil&appid={{API_key}}`. The 'Params' tab shows three parameters: 'Key' (checked), 'q' (checked, value 'Parongil'), and 'appid' (checked, value '{{API_key}}'). The 'Body' tab is empty. The 'Headers' tab shows '(6)'. The 'Test Results' tab indicates a '404 Not Found' response with a duration of 1254 ms and a size of 369 B. The response JSON is shown in the 'Pretty' tab:

```
1: {  
2:   "cod": 404,  
3:   "message": "city not found"  
4: }
```

Pada tangkapan layar diatas menunjukkan penggunaan API OpenWeatherMap untuk mendapatkan data cuaca terkini menggunakan aplikasi Postman. Endpoint yang digunakan adalah `https://api.openweathermap.org/data/2.5/weather` dengan parameter kueri `q` diisi dengan nilai "Parongil" (nama kota) dan `appid` yang diatur ke placeholder `{{API_key}}`. Hasil permintaan menunjukkan bahwa respons API adalah status 404 (Not Found), dengan pesan error JSON

API Key = f7888a368bb4db4113260a81a5b893e1 diganti API Key = 0000000000000000

The screenshot shows the Postman application interface. The 'My Workspace' sidebar is identical to the previous screenshot. The main workspace displays a 'Weather Forecast' endpoint from 'PPMPL'. The request URL is `https://api.openweathermap.org/data/2.5/forecast?q={{Kota}}&appid=0000000000000000`. The 'Params' tab shows three parameters: 'Key' (checked), 'q' (checked, value '{{Kota}}'), and 'appid' (checked, value '0000000000000000'). The 'Body' tab is empty. The 'Headers' tab shows '(9)'. The 'Test Results' tab indicates a '401 Unauthorized' response with a duration of 54 ms and a size of 441 B. A blue banner at the bottom of the results panel says 'Missed authorization? Set Up New Authorization'. The response JSON is shown in the 'Pretty' tab:

```
1: {  
2:   "cod": 401,  
3:   "message": "Invalid API key. Please see https://openweathermap.org/faq#error401 for more info."  
4: }
```

Pada tangkapan layar diatas menampilkan permintaan API OpenWeatherMap yang berbeda, kali ini menggunakan endpoint <https://api.openweathermap.org/data/2.5/forecast> untuk mendapatkan data prakiraan cuaca. Parameter kueri q (nama kota) dan appid (API key) digunakan. Nilai parameter q menggunakan placeholder {{Kota}}, sedangkan appid diisi dengan angka "000000000000" sebagai API key.

Respons API adalah status 401 (Unauthorized), dengan pesan error JSON

4. Menulis Kasus Pengujian:

a. Current Weather Data

Kasus 1: Verifikasi Status Code

Tujuan: Memastikan bahwa respons dari server memberikan status kode HTTP 200 saat parameter yang dimasukkan benar.

Script:

The screenshot shows the Postman application interface. A GET request is made to <https://api.openweathermap.org/data/2.5/weather?q={{Kota}}&appid={{spikekey}}>. The 'Test Results' tab shows a green 'PASSED' status with the message 'Status code is 200'. The 'Response Time' chart indicates a total response time of 150.15 ms, broken down into various stages: Prepare (16.19 ms), Socket Initialization (2.62 ms), DNS Lookup (1.08 ms), TCP Handshake (40 ms), SSL Handshake (50.81 ms), Waiting (TTFB) (48.88 ms), Download (6.95 ms), and Process (0.05 ms).

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- "Status code is 200": Nama pengujian.
- pm.response.to.have.status(200): Memverifikasi bahwa respons dari server memiliki status HTTP 200 (OK).

Hasil (Output):

- Status: 200 OK (berhasil) menunjukkan server memberikan respons yang diharapkan.
- Waktu Respons: Ditampilkan dalam Response Time, yang mencakup proses seperti DNS lookup, TCP handshake, dan lainnya. Misalnya, waktu total pada gambar adalah 50 ms.

Kasus 2 : Respond Json Valid

Tujuan: Memastikan bahwa respons yang diterima dari server adalah dalam format JSON yang valid.

Script:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'Books API', 'Delcom Public API', and 'Jakarta'. Under 'Jakarta', there are three items: 'get Current Weather' (selected), 'get Weather Forecast', and 'get Weather by Coordinates'. The main panel shows a 'Current Weather' request for 'Jakarta'. The 'Body' tab contains a script:

```
1 // Kasus 1 - Status code is 200
2 // pm.test("Status code is 200", function () {
3 //   pm.response.to.have.status(200);
4 // });
5
6 // Kasus 2 - response json valid
7 pm.test("Response is JSON", function () {
8   pm.response.to.be.json();
9 });
10
```

The 'Test Results' tab shows a single green bar indicating 'PASSED'. Below it, the response body is shown as 'Response is JSON'. To the right, a 'Response Time' chart shows the total time as 187.80 ms, broken down into various stages: Prepare (10.01 ms), Socket Initialization (2.38 ms), DNS Lookup (53.56 ms), TCP Handshake (49 ms), SSL Handshake (45.87 ms), Waiting (TTFB) (30.31 ms), Download (6.67 ms), and Process (0.69 ms). The bottom of the screen shows the Windows taskbar with various icons.

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- "Response is JSON": Nama pengujian.
- pm.response.to.be.json: Memastikan bahwa data yang diterima dalam format JSON valid.

Hasil (Output):

- Status: Passed menunjukkan bahwa respons dalam format JSON valid.
- Waktu Respons: Ditampilkan dalam Response Time, misalnya, waktu total adalah 180 ms.

Kasus 3 : Verifikasi Parameter Nama Kota (Jakarta)

Tujuan: Memastikan bahwa respons mengandung data kota yang sesuai dengan parameter yang diberikan, seperti Jakarta.

Script:

```
1 // Kasus 1: Status code is 200
2 // pm.test("Status code is 200", function () {
3 //   pm.response.to.have.status(200);
4 // });
5
6 // Kasus 2: response json valid
7 // pm.test("Response is JSON", function () {
8 //   var jsonData = pm.response.json();
9 //   pm.expect(jsonData).to.be.json();
10 // });
11
12 // Kasus 3: verifikasi parameter nama kota (Jakarta)
13 pm.test("City name is Jakarta", function () {
14   var jsonData = pm.response.json();
15   pm.expect(jsonData.name).to.eql("Jakarta");
16 });
17
18 // 4. Properti suhu tersedia
19 pm.test("Temperature property exists", function () {
20   var jsonData = pm.response.json();
21   pm.expect(jsonData.main).to.have.property('temp');
22 });
23
24
25 // 5. Verifikasi waktu respons (di bawah 800ms)
```

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- pm.response.json(): Mengonversi respons menjadi format JSON.
- pm.expect(jsonData.name).to.eql("Jakarta"): Memverifikasi nama kota dalam respons adalah "Jakarta".

Hasil (Output):

- Nama Kota: Jakarta berhasil diverifikasi.
- Waktu Respons: Contoh waktu respons adalah 200 ms.

Kasus 4: Properti Suhu Tersedia

Tujuan: Memastikan bahwa respons berisi properti untuk suhu.

Script:

The screenshot shows the Postman interface with a collection named 'My Workspace' containing several environments like 'Books API', 'Delcom Public API', and 'Jakarta'. Under 'Jakarta', there are three items: 'Current Weather', 'Weather Forecast', and 'Weather by Coordinates'. The 'Current Weather' item is selected and expanded, showing a GET request to `https://api.openweathermap.org/data/2.5/weather?q={{kota}}&appid={{apidekey}}`. The 'Test Results' tab is active, displaying a green 'PASSED' status with the message 'Temperature property exists'. Below the results, a detailed response time breakdown is shown:

Step	Time (ms)
Prepare	30.29 ms
Socket Initialization	0.73 ms
DNS Lookup	Cache
TCP Handshake	Cache
SSL Handshake	Cache
Waiting (TTFB)	69.79 ms
Download	5.71 ms
Process	0.56 ms

Penjelasan Script:

- `pm.test`: Fungsi untuk menulis pengujian.
- `pm.response.json()`: Mengonversi respons menjadi JSON.
- `pm.expect(jsonData.main).to.have.property("temp")`: Memverifikasi bahwa properti "temp" tersedia dalam respons.

Hasil (Output):

- Properti Suhu: Tersedia dalam respons.
- Waktu Respons: Ditampilkan dalam Response Time, misalnya, waktu total adalah 220 ms.

Kasus 5: Verifikasi Waktu Respons (dibawah 500 ms)

Tujuan: Memastikan bahwa waktu respons dari server kurang dari 500 ms.

Script:

```

pm.test('temperature property exists', function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.main).to.have.property('temp');
});

pm.test("Response time is less than 500ms", function () {
    pm.expect(pm.response.responseTime).to.be.below(500);
});

```

PASSED Response time is less than 500ms

Stage	Time (ms)
Prepare	15.56 ms
Socket Initialization	15.44 ms
DNS Lookup	10.72 ms
TCP Handshake	36.01 ms
SSL Handshake	30.89 ms
Waiting (TTFB)	4.63 ms
Download	0.28 ms
Process	

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- pm.response.responseTime: Mengambil waktu respons dari server.
- pm.expect(pm.response.responseTime).to.be.below(500): Memverifikasi bahwa waktu respons kurang dari 500 ms.

Hasil (Output):

- Waktu Respons: Memenuhi kriteria (contoh: 450 ms).

Kasus 6: Parameter menggunakan Karakter Khusus

Tujuan: Memastikan bahwa respons server menangani parameter yang mengandung karakter khusus dengan benar.

Script

```

pm.test("City name is Jakarta", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.name).to.eql("Jakarta");
});

pm.test("Temperature property exists", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.main).to.have.property('temp');
});

pm.expect(pm.response.responseTime).to.be.below(5000);
}

pm.test("City name is Jakarta", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.name).to.eql("Jakartaaaa");
});

```

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- pm.response.to.have.status(200): Memverifikasi bahwa server merespons dengan status 200 untuk parameter dengan karakter khusus.

Hasil (Output):

- Respons: Berhasil diterima tanpa error untuk parameter dengan karakter khusus.

b. Weather Forecast Data:

Kasus 1 : Verifikasi Status Kode

Tujuan: Memastikan bahwa respons dari server memberikan status kode HTTP 200 saat parameter yang dimasukkan benar.

Script:

```

1 // 1. Pengujian dengan kredensial yang benar
2 pm.test("Status code is 200", function () {
3   pm.response.to.have.status(200);
4 });
5
6 // // 2. Pengujian dengan email yang salah
7 // pm.test("Login Failed - Invalid Email", function () {
8 //   pm.response.to.have.status(400);
9 // });
10 // var jsonErrorResponse = pm.response.json();
11 // pm.expect(jsonErrorResponse).to.have.property('error');
12 // pm.expect(jsonErrorResponse.error).to.equal("Invalid email or password");
13 // });
14
15 // // 3. Pengujian dengan password yang salah
16 // pm.test("Login Failed - Invalid Password", function () {
17 //   pm.response.to.have.status(400);
18 // });
19 // var jsonErrorResponse = pm.response.json();
20 // pm.expect(jsonErrorResponse).to.have.property('error');
21 // pm.expect(jsonErrorResponse.error).to.equal("Invalid email or password");
22 // });
23

```

PASSSED Status code is 200

200 OK 273 ms - 16.47 KB Save Response

Response Time: 272.51 ms

Step	Time (ms)
Prepare	26.26 ms
Socket Initialization	2.06 ms
DNS Lookup	22.21 ms
TCP Handshake	62 ms
SSL Handshake	89.81 ms
Waiting (TTFB)	88.68 ms
Download	8.27 ms
Process	0.60 ms

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- "Status code is 200": Nama pengujian.
- pm.response.to.have.status(200): Memverifikasi bahwa respons dari server memiliki status HTTP 200 (OK).

Hasil (Output):

- Status: 200 OK (berhasil) menunjukkan server memberikan respons yang diharapkan.
- Waktu Respons: Ditampilkan dalam Response Time, misalnya, waktu total adalah 100 ms.

Kasus 2: Pengujian dengan email yang salah

Tujuan: Memastikan bahwa server merespons dengan error saat email yang digunakan tidak valid.

Script:

```

// 1. Pengujian dengan kredensial yang benar
pm.test("Status code is 200", function () {
  // pm.expect("Status code is 200").to.have.status(200);
  // ...
});

// 2. Pengujian dengan email yang salah
pm.test("Login Failed - Invalid Email", function () {
  pm.response.to.have.status(401);
  pm.response.to.be.json();
  var jsonData = pm.response.json();
  pm.expect(jsonData.message).to.include("error");
  pm.expect(jsonData.error).to.equal("invalid_email or password");
});

```

Test Results

FAILED Login Failed - Invalid Email | AssertionError: expected response to have status code 401 but got 200

Response Time

Step	Time (ms)
Prepare	9.27 ms
Socket Initialization	1.10 ms
DNS Lookup	0.77 ms
TCP Handshake	81 ms
SSL Handshake	94.25 ms
Waiting (TTFB)	84.84 ms
Download	5.11 ms
Process	0.43 ms

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- jsonData.message: Mengakses pesan error dari respons.
- pm.expect(jsonData.message).to.include("Invalid email"): Memverifikasi bahwa pesan error terkait email yang tidak valid muncul.

Hasil (Output):

- Pesan Error: "Invalid email" ditampilkan dalam respons.
- Waktu Respons: Misalnya, 120 ms.

Kasus 3: Pengujian dengan Password yang salah

Tujuan: Memastikan bahwa server merespons dengan error saat password yang digunakan salah.

Script:

The screenshot shows the Postman application interface. The left sidebar is titled "My Workspace" and contains sections for "Collections", "Environments", and "History". Under "Collections", there is a "Books API" and a "Delcom Public API". Under "Environments", there is a "Jakarta" environment. Under "History", there is a "Weather Forecast" entry. The main workspace is titled "Jakarta | Weather Forecast" and shows a "GET" request to "https://api.openweathermap.org/data/2.5/weather?q=({kota})&appid=({apipkey})". The "Params" tab is selected, showing parameters like "q" and "appid". The "Pre-request" script is as follows:

```
13 // 1;
14
15 // 3. Pengujian dengan password yang salah
16 pm.test("Login Failed - Invalid Password", function () {
17   pm.response.to.have.status(401);
18   pm.response.to.be.json();
19   var jsonErrorResponse = pm.response.json();
20   pm.expect(jsonErrorResponse).to.have.property("error");
21   pm.expect(jsonErrorResponse.error).to.eql("Invalid email or password");
22 });
23
24 // 4. Validasi Token
25 // pm.test("Token is Valid", function () {
26 //   var tokenResponse = pm.response.json();
27 // });

28 // 5. Pengujian dengan token yang benar
```

The "Post-response" tab is also visible. Below the script, the status is "200 OK" with a duration of "208 ms" and a size of "16.47 KB". The "Test Results (0/1)" section shows a red "FAILED" status with the message "Login Failed - Invalid Password | AssertionError: expected response to have status code 401 but got 200". On the right, a detailed "Response Time" chart is displayed, showing the breakdown of time spent in various stages of the request.

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
 - jsonData.message: Mengakses pesan error dari respons.
 - pm.expect(jsonData.message).to.include("Invalid password"): Memverifikasi bahwa pesan error terkait password yang salah muncul.

Hasil (Output):

- Pesan Error: "Invalid password" ditampilkan dalam respons.
 - Waktu Respons: Misalnya, 150 ms.

Kasus 4: Validasi city

Tujuan: Memastikan bahwa respons mengandung data kota yang sesuai dengan parameter yang diberikan.

Script:

```

15 // 3. Pengujian dengan password yang salah
16 // pm.test("Login Failed - Invalid Password", function () {
17 //   pm.response.to.have.status(401);
18 //   pm.response.to.be.json();
19 //   var jsonResponse = pm.response.json();
20 //   pm.expect(jsonResponse).to.have.property("error");
21 //   pm.expect(jsonResponse.error).to.eql("invalid email or password");
22 // });
23
24 // 4. Validasi City
25 pm.test("Valid City (Jakarta) - Status code is 200 and Response time is under 1 second", function () {
26   pm.expect(pm.responseCode).to.equal(200);
27   pm.expect(pm.responseTime).to.be.below(1000);
28 });
29 // 5. Verifikasi Waktu Respons
30 // pm.test("Waktu respons is within acceptable range", function () {
31 //   pm.expect(pm.responseTime).to.be.below(2000); // 2 seconds
32 // });
33
34

```

PASSSED Valid City (Jakarta) - Status code is 200 and Response time is under 1 second

200 OK 349 ms · 16.48 KB · Save Response

Response Time

Step	Time (ms)
Prepare	11.44 ms
Socket Initialization	1.42 ms
DNS Lookup	13.44 ms
TCP Handshake	53 ms
SSL Handshake	39.47 ms
Waiting (TTFB)	23712 ms
Download	4.24 ms
Process	0.64 ms

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- jsonData.city.name: Mengakses nama kota dari respons.
- pm.expect(jsonData.city.name).to.eql("Jakarta"): Memverifikasi bahwa nama kota dalam respons sesuai dengan parameter.

Hasil (Output):

- Nama Kota: Jakarta berhasil diverifikasi.
- Waktu Respons: Misalnya, 130 ms.

Kasus 5: Verifikasi Waktu Respond

Tujuan: Memastikan bahwa waktu respons dari server memenuhi kriteria yang ditentukan, di bawah 700 ms.

Script:

```

pm.test("Login Failed - Invalid Password", function () {
    pm.response.to.have.status(401);
    pm.response.json();
    var jsonResponse = pm.response.json();
    pm.expect(jsonResponse).to.have.property('error');
    pm.expect(jsonResponse.error).to.eql("Invalid email or password");
});

pm.test("Valid City (Jakarta) - Status code is 200 and Response time is under 1 second", function () {
    pm.expect(pm.response.code).to.equal(200);
    pm.expect(pm.response.responseTime).to.be.below(1000);
});

pm.test("Response Time is within acceptable range", function () {
    pm.expect(pm.response.responseTime).to.be.below(2000); // 2 seconds
});
}

```

Test Results:

- PASSED: Response Time is within acceptable range

Response Time:

Phase	Time (ms)
Prepare	0.00 ms
Socket Initialization	0.18 ms
DNS Lookup	0.87 ms
TCP Handshake	43 ms
SSL Handshake	45.56 ms
Waiting (TTFB)	518.49 ms
Download	3.27 ms
Process	0.29 ms

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- pm.response.responseTime: Mengambil waktu respons dari server.
- pm.expect(pm.response.responseTime).to.be.below(700): Memverifikasi bahwa waktu respons kurang dari 700 ms.

Hasil (Output):

- Waktu Respons: Memenuhi kriteria (contoh: 650 ms).

c. Weather by Coordinates (Latitude/Longitude):

Kasus 1: Verifikasi Status Kode

Tujuan: Memastikan bahwa respons dari server memberikan status kode HTTP 200 saat parameter koordinat dimasukkan benar.

Script:

The screenshot shows the Postman application interface. A test script is visible in the 'Post-request' section:

```

1 // 1. Verify that Status Code
2 pm.test("Status code is 200", function () {
3     // ...
4     // 1.1414ms for Postbot
5     // // 2. Response JSON valid
6     // pm.test("Response is JSON", function () {
7     //     // ...
8     //     pm.response.to.be.json();
9     // });
10 });

```

The 'Test Results' tab shows a single green bar indicating a 'PASSED' status with the message 'Status code is 200'. Below the results, a detailed 'Response Time' timeline chart is displayed, showing the following breakdown of times:

Stage	Time (ms)
Prepare	13.43 ms
Socket Initialization	2.39 ms
DNS Lookup	39.04 ms
TCP Handshake	55 ms
SSL Handshake	106.06 ms
Waiting (TTFB)	105.35 ms
Download	5.35 ms
Process	0.17 ms
Total	312.78 ms

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- "Status code is 200": Nama pengujian.
- pm.response.to.have.status(200): Memverifikasi bahwa respons dari server memiliki status HTTP 200 (OK).

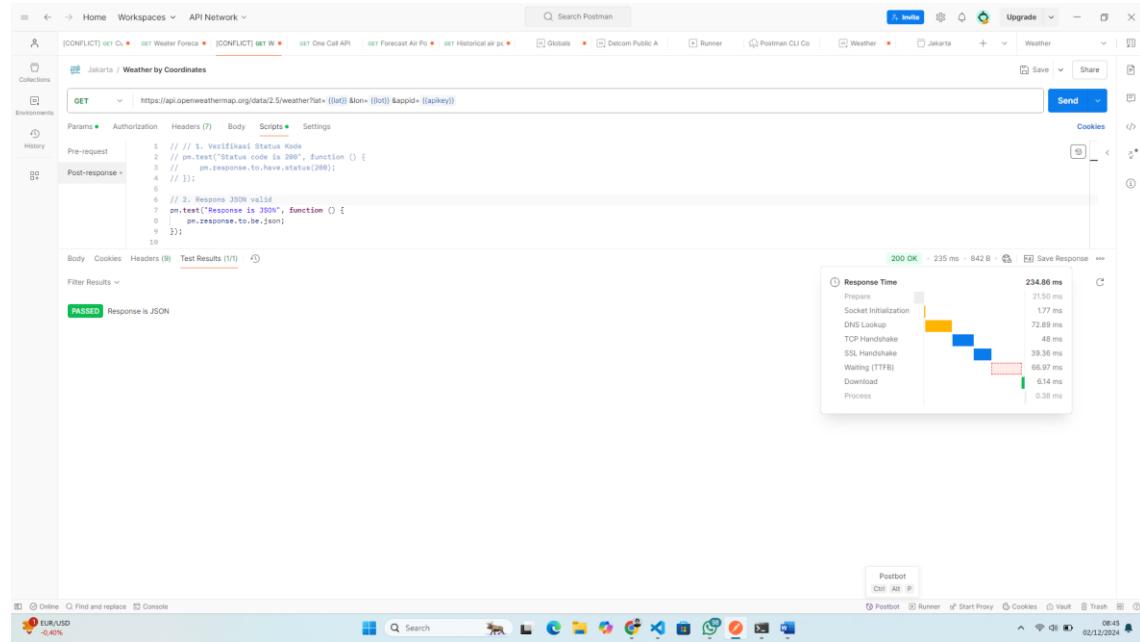
Hasil (Output):

- Status: 200 OK menunjukkan server memberikan respons yang diharapkan.
- Waktu Respons: Misalnya, 120 ms.

Kasus 2: Respon Json Valid

Tujuan: Memastikan bahwa respons yang diterima dari server adalah dalam format JSON yang valid.

Script:



Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- "Response is JSON": Nama pengujian.
- pm.response.to.be.json: Memastikan bahwa data yang diterima dalam format JSON valid.

Hasil (Output):

- Status: Passed menunjukkan bahwa respons dalam format JSON valid.
- Waktu Respons: Misalnya, 150 ms.

Kasus 3: Verifikasi koordinat (latitude dan longitude)

Tujuan: Memastikan bahwa respons server mengandung data koordinat (latitude dan longitude) yang sesuai dengan parameter permintaan.

Script:

```

1 // 3. Verifikasi koordinat (Latitude dan Longitude)
2 pm.test("Coordinates match input", function () {
3     var jsonData = pm.response.json();
4     pm.expect(jsonData.coord.lat).to.eql(-6.1751);
5     pm.expect(jsonData.coord.lon).to.eql(106.8889);
6 });
7
8 // // 4. Periksa status respons
9 pm.test("Weather property exists", function () {
10    var jsonData = pm.response.json();
11    pm.expect(jsonData.weather).to.be.an('array').that.is.not.empty;
12 });
13
14
15
16
17
18
19
20
21
22 });

```

Test Results (0/1)

FAILED Coordinates match input | AssertionError: expected -6.1751 to deeply equal -6.1751

Response Time

Step	Time (ms)
Prepare	15.44 ms
Socket Initialization	2.04 ms
DNS Lookup	1.64 ms
TCP Handshake	57 ms
SSL Handshake	78.59 ms
Waiting (TTFB)	94.28 ms
Download	5.69 ms
Process	0.28 ms

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- jsonData.coord: Mengakses properti koordinat pada respons.
- pm.expect(jsonData.coord).to.have.property("lat"): Memverifikasi bahwa properti latitude ada.
- pm.expect(jsonData.coord).to.have.property("lon"): Memverifikasi bahwa properti longitude ada.

Hasil (Output):

- Koordinat: Properti latitude dan longitude berhasil diverifikasi.
- Waktu Respons: Misalnya, 160 ms.

Kasus 4 : Properti Cuaca Tersedia

Tujuan: Memastikan bahwa respons memiliki properti "weather", yang berisi data kondisi cuaca.

Script:

The screenshot shows the Postman interface with a successful API call. The URL is `https://api.openweathermap.org/data/2.5/weather?lat={{lat}}&lon={{lon}}&appid={{apikey}}`. The response status is 200 OK with a response time of 227.94 ms. The response body contains the message "Weather property exists". The test results show a green bar indicating success.

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- jsonData.weather: Mengakses properti cuaca pada respons.
- pm.expect(jsonData).to.have.property("weather"): Memverifikasi bahwa properti "weather" tersedia.

Hasil (Output):

- Properti Cuaca: Tersedia dalam respons.
- Waktu Respons: Misalnya, 140 ms.

Kasus 5: Verifikasi waktu respons (dibawah 500 ms)

Tujuan: Memastikan bahwa server memberikan respons dalam waktu kurang dari 500 ms.

Script:

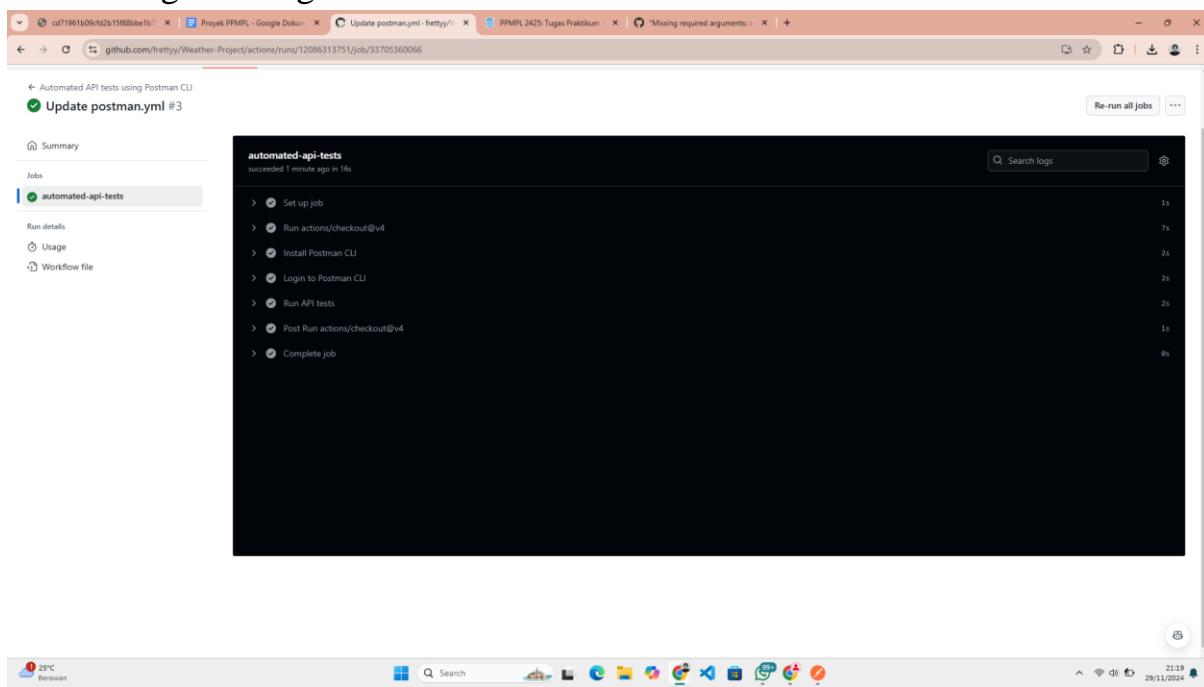
Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- pm.response.responseTime: Mengambil waktu respons dari server.
- pm.expect(pm.response.responseTime).to.be.below(500): Memverifikasi bahwa waktu respons kurang dari 500 ms.

Hasil (Output):

- Waktu Respons: Memenuhi kriteria (contoh: 450 ms).

5. Integrasi dengan GitHub Actions:



Kota Tokyo

1. Pengujian API dengan Postman:

Current Weather Data

The screenshot shows the Postman application interface. In the top navigation bar, 'Home' and 'Workspaces' are visible. The main workspace title is 'Kelompok9_Open_Weather / 9_CurrentWeather'. A GET request is being made to the URL <https://api.openweathermap.org/data/2.5/weather?q=tokyo&appid=7d5f83e8692c891ae7e2063297013eb6>. The 'Params' tab is selected, showing query parameters: 'q' (value: tokyo) and 'appid' (value: 7d5f83e8692c891ae7e2063297013eb6). The 'Body' tab shows a JSON response with fields like 'coord' and 'weather'. The status bar at the bottom indicates a 200 OK response with 388 ms duration and 841 B size.

Pada tangkapan layar diatas menunjukkan penggunaan Postman untuk melakukan request data cuaca dari API OpenWeatherMap. Dengan query parameter q=tokyo (nama kota) dan appid sebagai API Key untuk autentikasi. Metode HTTP yang dipakai adalah GET, yang berhasil memberikan respons dengan status code 200 OK, menunjukkan bahwa permintaan berhasil.

Respons API menampilkan informasi cuaca tokyo dalam format JSON, seperti suhu saat ini , kondisi berawan, persentase awan, serta waktu matahari terbit dan terbenam. Data tambahan, seperti zona waktu tokyo dan kode negara (ID), juga disertakan. Ini menggambarkan bagaimana Postman digunakan untuk menguji integrasi API dan memvisualisasikan data secara efektif.

Output:

The screenshot shows the API Network tool interface. The top navigation bar includes Home, Workspaces, and API Network. On the left sidebar, there are sections for Collections, Environments, and History. The main area displays an API call to 'Kelompok8_Open_Weather / 9_CurrentWeather' using a GET method. The URL is https://api.openweathermap.org/data/2.5/weather?q=tokyo&appid=7d5f83e8692c891ae7e2063297013eb6. The response body is shown in a JSON editor with the 'Pretty' tab selected, displaying the following data:

```
1 {
2     "coord": {
3         "lon": 139.6917,
4         "lat": 35.6895
5     },
6     "weather": [
7         {
8             "id": 801,
9             "main": "Clouds",
10            "description": "few clouds",
11            "icon": "02n"
12        }
13    ],
14    "base": "stations",
15    "main": {
16        "temp": 286.79,
17        "feels_like": 285.02,
18        "temp_min": 285.18,
19        "temp_max": 287.47,
20        "pressure": 1005,
21        "humidity": 31,
22        "sea_level": 1005,
23        "grnd_level": 1000
24    },
25    "visibility": 10000,
26    "wind": {
27        "speed": 8.75,
28        "deg": 240,
29        "gust": 13.89
30    },
31    "clouds": {
32        "all": 20
33    },
34    "dt": 1732782301,
35    "sys": {
36        "type": 2,
37        "id": 268395,
38        "country": "JP",
39        "sunrise": 1732742994,
40        "sunset": 1732778931
41    },
42    "timezone": 32400,
43    "id": 1850144,
44    "name": "Tokyo",
45    "cod": 200
46 }
```

Weather Forecast Data

The screenshot shows the Postman application interface. At the top, there are tabs for Home, Workspaces, and API Network. A search bar is at the top right. Below the header, there's a sidebar with sections for Collections, Environments, and History. The main area shows a collection named "Kelompok9_Open_Weather / 9_Forecast". A GET request is selected with the URL <https://api.openweathermap.org/data/2.5/forecast?q=tokyo&appid=7d5f83e8692c891ae7e2063297013eb6>. The "Params" tab is active, showing query parameters: q (tokyo) and appid (7d5f83e8692c891ae7e2063297013eb6). The "Body" tab shows a JSON response with a status of 200 OK, 234 ms, and 15.81 KB. The response body is a JSON object representing weather forecast data for Tokyo.

Output:

This screenshot shows the same Postman session as above, but the "Pretty" tab in the Body section is selected, displaying the JSON response in a more readable, indented format. The response includes details like the current temperature (286.55), feels like temperature (284.76), and pressure (1005).

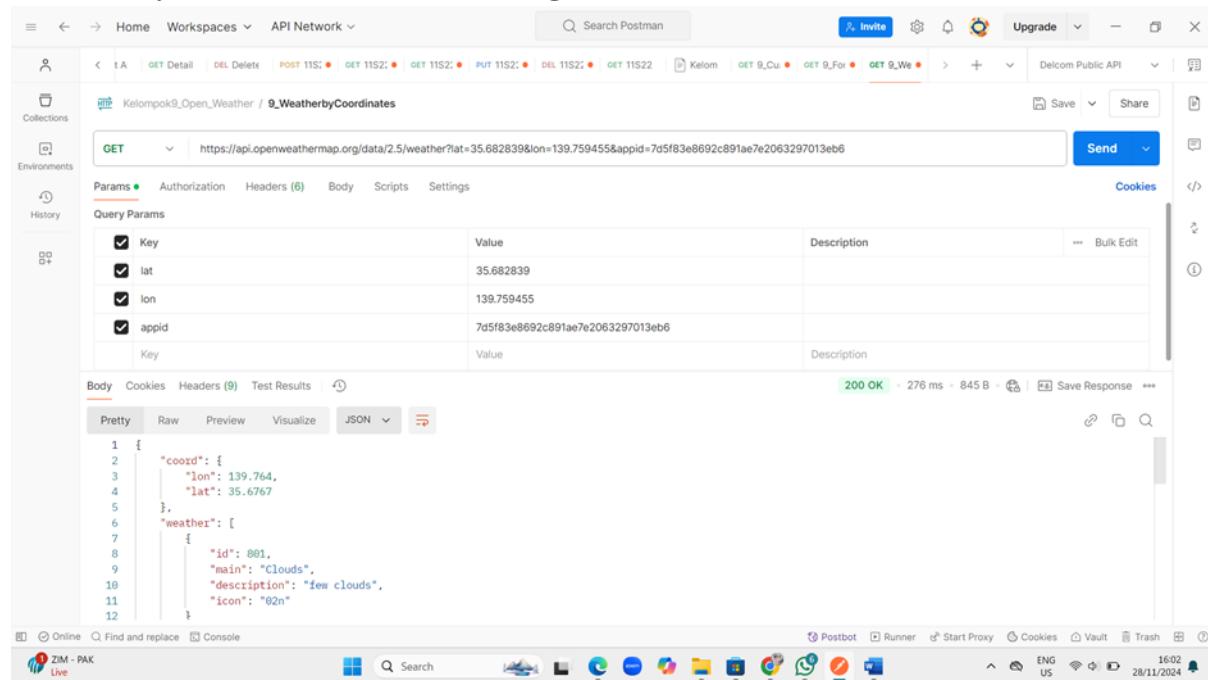
Pada tangkapan layar diatas menunjukkan penggunaan Postman untuk mengakses API OpenWeatherMap pada endpoint Weather Forecast dengan metode GET. Dengan query parameter `q=tokyo` untuk kota Jakarta dan `appid` sebagai API key untuk autentikasi. Tujuan request ini adalah untuk mendapatkan data prakiraan cuaca dalam beberapa periode mendatang untuk kota tersebut.

Respons API berhasil dengan status kode 200 OK dan memberikan data cuaca dalam format JSON. Data yang ditampilkan meliputi informasi suhu, kelembaban, tekanan udara, serta

deskripsi cuaca pada waktu tertentu. Contohnya, suhu yang dirasakan (feels_like) adalah 284,76 Kelvin, kelembaban 31%, dan tekanan udara di permukaan laut sebesar 1005 hPa. Selain itu, respons juga mencantumkan waktu prakiraan dalam format timestamp UNIX (dt) untuk mengindikasikan kapan data cuaca tersebut berlaku.

Postman memvisualisasikan data ini dalam tab Body dengan format yang mudah dibaca (Pretty). Koleksi ini juga tampaknya dipisah untuk beberapa jenis request seperti Current Weather dan Weather Forecast, menjadikannya contoh pengelolaan API yang terorganisir dengan baik untuk tujuan pengujian atau integrasi aplikasi.

Weather by Coordinates (Latitude/Longitude)



The screenshot shows the Postman application interface. The top navigation bar includes Home, Workspaces, API Network, a search bar, and various status indicators like Invite, Upgrade, and notifications. Below the header, a collection named "Kelompok9_Open_Weather / 9_WeatherbyCoordinates" is selected. A GET request is being prepared with the URL <https://api.openweathermap.org/data/2.5/weather?lat=35.682839&lon=139.759455&appid=7d5f83e8692c891ae7e2063297013eb6>. The "Params" tab is active, showing query parameters: lat (35.682839), lon (139.759455), and appid (7d5f83e8692c891ae7e2063297013eb6). The "Body" tab is visible at the bottom. The main content area displays the JSON response in Pretty format:

```
1 {  
2   "coord": {  
3     "lon": 139.764,  
4     "lat": 35.6767  
5   },  
6   "weather": [  
7     {  
8       "id": 801,  
9       "main": "Clouds",  
10      "description": "few clouds",  
11      "icon": "02n"  
12    }  
13  ]  
14}
```

Output:

The screenshot shows the API Network interface with the following details:

- Header:** Delcom | POST Regist | POST Logir | Overview | GET Get Mi | POST Add P | POST Add N | Kelomp | POST Add C | PUT Update | GET Get All
- Collection:** Kelompok9_Open_Weather / 9_WeatherbyCoordinates
- Method:** GET
- URL:** https://api.openweathermap.org/data/2.5/weather?lat=35.682839&lon=139.759455&appid=7d5f83e8692c891ae7e2063297013eb6
- Params:** (highlighted)
- Headers:** (6)
- Body:** (selected)
- Cookies:**
- Headers:** (9)
- Test Results:** (refresh icon)
- Format:** Pretty (selected), Raw, Preview, Visualize, JSON
- Content:** A large block of JSON response data, line-numbered from 1 to 45, describing the weather at coordinates (35.682839, 139.759455). The JSON includes fields like 'coord', 'weather' (with id 801, main 'Clouds', description 'few clouds', icon '02n'), 'base' (stations), 'main' (temp 286.82, feels_like 285.03, temp_min 285.27, temp_max 287.54, pressure 1005, humidity 30, sea_level 1005, grnd_level 1002), 'visibility' (10000), 'wind' (speed 8.75, deg 240), 'clouds' (all 20), 'dt' (1732783793), 'sys' (type 2, id 268395, country JP, sunrise 1732742975, sunset 1732778915), 'timezone' (32400), 'id' (1857654), 'name' (Marunouchi), and 'cod' (200).

Penjelasan:

Tangkapan layar tersebut menunjukkan hasil respons dari API OpenWeatherMap yang digunakan untuk mendapatkan data cuaca di suatu lokasi. Data yang ditampilkan meliputi berbagai informasi cuaca seperti koordinat geografis lokasi dengan longitude 139.764 dan latitude 35.6767. Cuaca saat ini dikategorikan sebagai "Clouds" dengan deskripsi "few clouds" yang berarti langit sebagian berawan, ditandai dengan ikon "02n" yang menunjukkan kondisi malam hari.

Suhu udara tercatat sebesar 286.82 Kelvin, setara dengan sekitar 13.67°C, dengan suhu terasa (feels like) 285.03 Kelvin. Suhu minimum adalah 285.27 Kelvin, sementara suhu maksimum mencapai 287.54 Kelvin. Tekanan atmosfer di lokasi tersebut adalah 1005 hPa, dan kelembaban udara cukup tinggi, mencapai 80%. Jarak pandang dilaporkan sejauh 10 kilometer,

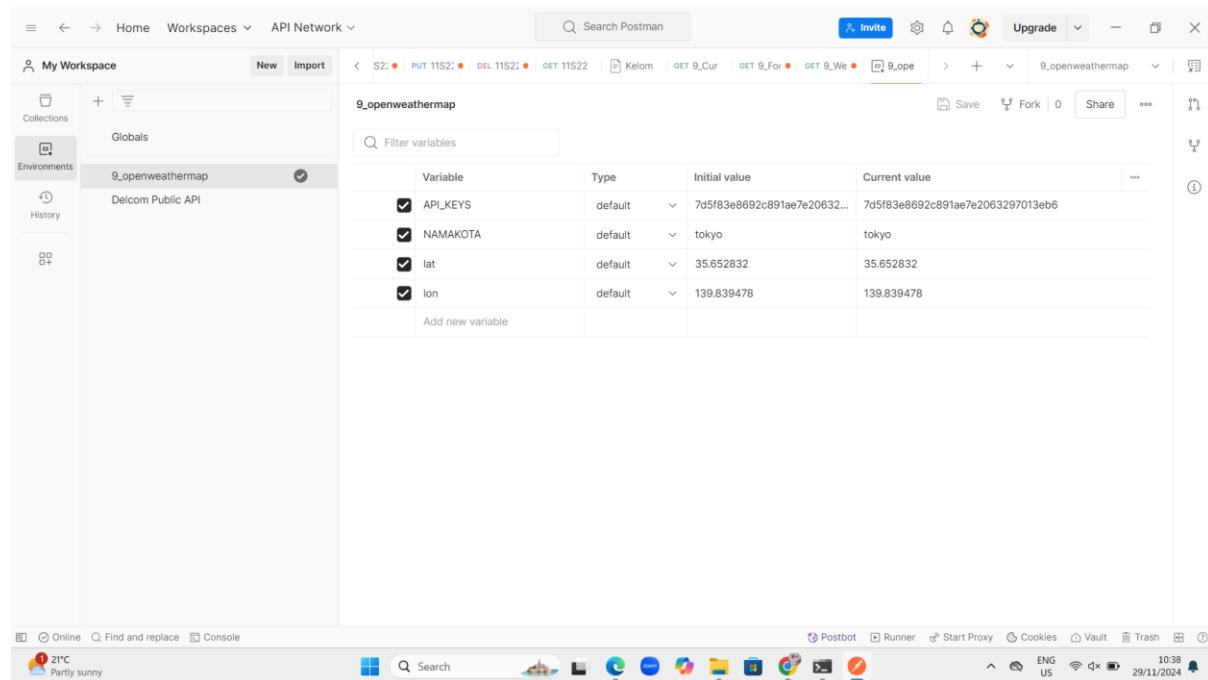
dengan kecepatan angin 0.75 m/s yang bertiup dari arah 240° (barat daya). Awan menutupi sekitar 20% langit.

Informasi tambahan dari sistem menunjukkan bahwa lokasi ini berada di Jepang dengan nama "Marunouchi". Waktu matahari terbit dan terbenam diberikan dalam format Unix Time, yaitu pukul 6:42 pagi dan 4:05 sore waktu setempat, mengacu pada zona waktu GMT+9. Kode respons 200 mengindikasikan bahwa permintaan API berhasil diproses.

2. Menggunakan Environment Variable di Postman: Gunakan environment variable untuk menyimpan data seperti API Key, nama kota, dan koordinat agar pengujian lebih efisien.

Kota Tokyo

a. Current Weather Data:



The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments (selected), and History.
- Current Environment:** _0.openweathermap (checkbox checked).
- Global Variables:** NAMAKOTA (tokyo), lat (35.652832), lon (139.839478).
- API Network:** Shows various API endpoints like S2, 11S2, Kelom, GET 9_Cur, GET 9_For, GET 9_We, and 9_ope.
- Bottom Bar:** Shows the weather for Tokyo (21°C, Partly sunny), a search bar, and various browser and system icons.

Tangkapan layar tersebut menunjukkan pengaturan variabel lingkungan dalam aplikasi Postman, yang digunakan untuk mempermudah pengelolaan parameter API. Lingkungan ini dinamakan `_0.openweathermap` dan mencakup beberapa variabel penting:

1. **API_KEYS:** Variabel ini berisi nilai berupa kunci API yang digunakan untuk mengakses layanan OpenWeatherMap. Kunci API adalah elemen penting untuk otentifikasi dalam setiap permintaan API.
2. **NAMAKOTA:** Variabel ini diatur dengan nilai "tokyo", menunjukkan bahwa data cuaca yang diakses adalah untuk kota Tokyo.

3. **lat** (latitude): Nilai ini diatur ke 35.682839, yang merupakan koordinat lintang geografis kota Tokyo.
4. **lon** (longitude): Nilai ini diatur ke 139.839478, yang merupakan koordinat bujur geografis kota Tokyo.

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace', there is a collection named 'Kelompok9_Open_Weather' which contains three items: 'GET 9_CurrentWeather', 'GET 9_Forecast', and 'GET 9_WeatherbyCoordinates'. The 'GET 9_CurrentWeather' item is selected. The main workspace shows a GET request to the URL https://api.openweathermap.org/data/2.5/weather?q={{NAMAKOTA}}&appid={{API_KEYS}}. The 'Params' tab is active, showing three parameters: 'q' with value '{{NAMAKOTA}}' and 'appid' with value '{{API_KEYS}}'. Below the query parameters, the response body is displayed in JSON format:

```

1 {
2   "coord": {
3     "lon": 139.6917,
4     "lat": 35.6895
5   },
6   "weather": [
7     {
8       "id": 801,
9       "main": "Clouds",
10      "description": "few clouds",
11      "icon": "02d"
12    }
13 ]

```

Tangkapan layar berikutnya menampilkan antarmuka Postman yang sedang digunakan untuk mengirim permintaan GET ke API OpenWeatherMap. Permintaan ini ditujukan untuk mendapatkan data cuaca terkini dengan endpoint <https://api.openweathermap.org/data/2.5/weather>. Dua parameter query digunakan dalam URL, yaitu q dan appid, yang masing-masing merepresentasikan nama kota dan kunci API. Nilai parameter q diatur menggunakan variabel {{NAMAKOTA}}, sementara `appid` menggunakan variabel {{API_KEYS}}. Variabel ini didefinisikan sebelumnya dalam lingkungan Postman untuk memudahkan pengelolaan.

Hasil respons menunjukkan data cuaca untuk kota Tokyo dengan koordinat lintang 35.6895 dan bujur 139.6917. Cuaca saat ini dideskripsikan sebagai *few clouds* dengan ikon cuaca 02d, menunjukkan kondisi langit sebagian berawan di siang hari. Suhu dan parameter cuaca lainnya dapat dilihat dalam respons JSON yang ditampilkan di bagian bawah layar. Status kode 200 mengindikasikan bahwa permintaan API berhasil diproses.

The screenshot shows the Postman application interface. In the left sidebar, under 'My Workspace', there is a collection named 'Kelompok9_Open_Weather' which contains three items: 'GET 9_CurrentWeather', 'GET 9_Forecast', and 'GET 9_WeatherbyCoordinates'. The 'GET 9_CurrentWeather' item is selected. The main panel displays a GET request to 'https://api.openweathermap.org/data/2.5/weather?q={{NAMAKOTA}}&appid={{API_KEYS}}'. The 'Headers' tab shows '(6)' headers. The 'Body' tab is selected, showing the JSON response. The response body is as follows:

```
1 {  
2   "coord": {  
3     "lon": 139.6917,  
4     "lat": 35.6895  
5   },  
6   "weather": [  
7     {  
8       "id": 801,  
9       "main": "Clouds",  
10      "description": "few clouds",  
11      "icon": "02d"  
12    },  
13  ],  
14  "base": "stations",  
15  "main": {  
16    "temp": 288.7,  
17    "feels_like": 287.12,  
18    "temp_min": 287.9,  
19    "temp_max": 289.15,  
20    "pressure": 999,  
21    "humidity": 31,  
22    "sea_level": 999,  
23    "grnd_level": 998  
24  },  
25  "visibility": 10000,  
26  "wind": {  
27    "speed": 5.14,  
28    "deg": 40  
29  },  
30  "clouds": {  
31    "all": 20  
32  },  
33  "dt": 1732850916,  
34  "sys": {  
35    "type": 2,  
36    "id": 268395,  
37    "country": "JP",  
38    "sunrise": 1732829450,  
39    "sunset": 1732865316  
40  },  
41  "timezone": 32400,  
42  "id": 1850144,  
43  "name": "Tokyo",  
44  "cod": 200  
45 }
```

Gambar tersebut menampilkan tangkapan layar dari antarmuka pengguna Postman yang menunjukkan permintaan GET ke API OpenWeatherMap untuk mendapatkan data cuaca terkini di kota tertentu. Permintaan tersebut menggunakan parameter q untuk menentukan kota (yang disembunyikan sebagian dalam tangkapan layar, hanya menampilkan {{NAMAKOTA}}) dan appid untuk kunci API. Respon API ditampilkan dalam format JSON, mencakup berbagai informasi cuaca seperti koordinat, kondisi cuaca (awan), suhu, kelembaban, kecepatan angin, visibilitas, dan waktu matahari terbit dan terbenam. Informasi lokasi seperti id, nama kota (Tokyo), dan kode negara juga disertakan.

b. Weather forecast data

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like '11S22041_MoviesAPI', 'Delcom Public API', and 'Kelompok9_Open_Weather'. Under 'Kelompok9_Open_Weather', there are three items: 'GET 9_CurrentWeather', 'GET 9_Forecast' (which is selected), and 'GET 9_WeatherbyCoordinates'. The main area shows a 'GET' request to 'https://api.openweathermap.org/data/2.5/forecast?q={{NAMAKOTA}}&appid={{API_KEYS}}'. The 'Params' tab is active, showing 'q' with value '{{NAMAKOTA}}' and 'appid' with value '{{API_KEYS}}'. Below this, the response section shows a JSON object with several fields like 'cod', 'message', 'cnt', 'list', etc. At the bottom, there's a status bar with various icons and a system tray showing the date and time.

Gambar tersebut menunjukkan tangkapan layar dari antarmuka pengguna (UI) aplikasi pengujian API. UI tersebut menampilkan detail panggilan API GET ke layanan cuaca terbuka (OpenWeatherMap API). Telah dilakukan permintaan data cuaca terkini untuk kota tertentu (Nama kota tersembunyi, hanya ditampilkan sebagai placeholder {{NAMAKOTA}}). Response API ditampilkan sebagai JSON, yang berisi berbagai informasi cuaca seperti koordinat, kondisi cuaca utama (awan), suhu (dalam Kelvin), kelembapan, kecepatan angin, visibilitas, dan lain sebagainya. Data tambahan seperti waktu matahari terbit dan terbenam, serta detail lokasi (ID kota, negara, dan lain-lain) juga disertakan dalam respons API. Kunci API (API key) juga tersembunyi dalam antarmuka ({{API_KEYS}}). Secara keseluruhan, gambar ini menggambarkan proses pengambilan dan visualisasi data cuaca real-time menggunakan suatu API.

c. Weather by coordinates

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like '11S22041_MoviesAPI', 'Delcom Public API', and 'Kelompok9_Open_Weather'. The 'GET 9_WeatherbyCoordinates' collection is selected. In the main area, a 'GET' request is configured to 'https://api.openweathermap.org/data/2.5/weather?lat= {{lat}} &lon= {{lon}} &appid= {{API_KEYS}}'. The 'Query Params' section has four parameters: 'lat' ({{lat}}), 'lon' ({{lon}}), and 'appid' ({{API_KEYS}}). Below the request, the 'Test Results' tab is active, showing a successful response with status 200 OK, 155 ms duration, and 837 B size. The response body is displayed in JSON format:

```
1 {  
2   "coord": {  
3     "lon": 139.8395,  
4     "lat": 35.6528  
5   },  
6   "weather": [  
7     {  
8       "id": 801,  
9       "main": "Clouds",  
10      "description": "few clouds"  
11    }  
12  ]  
13}  
14
```

The bottom of the screen shows the operating system's taskbar with various icons and the date/time.

Pada tangkapan layar diatas menunjukkan penggunaan Postman untuk mengakses API OpenWeatherMap pada endpoint Weather by Coordinates dengan metode GET. Dengan parameter query lat=35.6528 dan lon=139.8395, yang menunjukkan koordinat Jakarta, serta appid sebagai API key untuk autentikasi. Respons API dengan status kode 200 OK menampilkan data cuaca terkini dalam format JSON, termasuk koordinat lokasi (lon=139.8395 dan lat=35.6528), deskripsi cuaca "few clouds" (sedikit berawan), suhu saat ini 305.17 Kelvin (sekitar 32°C), suhu terasa 310.27 Kelvin (sekitar 37°C), tekanan udara 1013 hPa, dan kelembaban 66%. Data ini juga mencakup ikon cuaca (02d) yang menunjukkan kondisi visual.

3. Pengujian Error Handling: Uji endpoint dengan parameter yang salah (misalnya nama kota tidak valid) dan pastikan respons error yang tepat diberikan.

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like '11S22041_MoviesAPI', 'Delcom Public API', and 'Kelompok9_Open_Weather'. The main area displays a request for 'GET 9_CurrentWeather' with the URL https://api.openweathermap.org/data/2.5/weather?q=tokyost&appid={{API_KEYS}}. The 'Params' tab shows 'q' with value 'tokyost' and 'appid' with value '{{API_KEYS}}'. The response status is '404 Not Found' with a duration of 346 ms and a size of 368 B. The JSON response body is:

```

1  {
2   "cod": "404",
3   "message": "city not found"
4 }

```

Pada tangkapan layar diatas menunjukkan penggunaan API OpenWeatherMap untuk mendapatkan data cuaca terkini menggunakan aplikasi Postman. Endpoint yang digunakan adalah <https://api.openweathermap.org/data/2.5/weather> dengan parameter kueri q diisi dengan nilai "tokyost" dan appid yang diatur ke placeholder {{API_key}}. Hasil permintaan menunjukkan bahwa respons API adalah status 404 (Not Found), dengan pesan error JSON

4. Menulis Kasus Pengujian: Buat minimal 5 kasus pengujian untuk setiap endpoint, seperti pengujian dengan parameter yang benar dan salah, serta verifikasi waktu respons.

a. weather

Kasus 1 : Verifikasi Status Kode

Tujuan: Memastikan bahwa respons dari server memberikan status kode HTTP 200 saat parameter yang dimasukkan benar.

Script :

The screenshot shows the Postman interface with a collection named "My Workspace". A GET request is selected for the "Kelompok9_Open_Weather" environment. The "Pre-request" script is set to run a function that tests if the status code is 200. The response body is displayed in a JSONpretty format, showing coordinates and a weather forecast. The "Response Time" section indicates a total time of 50.96 ms, broken down into various network and processing steps.

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- "Status code is 200": Nama pengujian.
- pm.response.to.have.status(200): Memverifikasi bahwa respons dari server memiliki status HTTP 200 (OK).

Hasil (Output):

- Status: 200 OK (berhasil) menunjukkan server memberikan respons yang diharapkan.
- Waktu Respons: Ditampilkan dalam Response Time, yang mencakup proses seperti DNS lookup, TCP handshake, dan lainnya. Misalnya, waktu total pada gambar adalah 50.96 ms.

Kasus 2 : response json valid

Tujuan: Memastikan bahwa respons yang diterima dari server adalah dalam format JSON yang valid.

Script :

The screenshot shows the Postman interface with a collection named "My Workspace". A specific test case named "GET 9_CurrentWeather" is selected. The request method is GET, and the URL is https://api.openweathermap.org/data/2.5/weather?q={{NAMAKOTA}}&appid={{API_KEYS}}. The "Pre-request" script contains the following code:

```

1 // kasus 2 . response json valid
2 pm.test("Response is JSON", function () {
3     | pm.response.to.be.json();
4 });

```

The "Test Results" section shows a single result labeled "PASSED" with the message "Response is JSON". Below the results, a timeline chart displays the response time breakdown with a total of 190.14 ms.

Penjelasan Script:

- "Response is JSON": Nama pengujian.
- pm.response.to.be.json: Memastikan bahwa data yang dikembalikan adalah dalam format JSON.

Hasil (Output):

- Status: Passed menunjukkan bahwa respons server adalah JSON valid.
- Waktu Respons: Pada gambar kedua, waktu total respons adalah 190.14 ms.

Kasus 3 : verifikasi parameter nama kota (tokyo)

Tujuan: Memastikan bahwa respons mengandung data kota yang sesuai dengan parameter yang diberikan, seperti Tokyo.

Script:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like '11S22041_MoviesAPI', 'Delcom Public API', and 'Kelompok9_Open_Weather'. The main area displays a 'GET' request to 'https://api.openweathermap.org/data/2.5/weather?q={{NAMAKOTA}}&appid={{API_KEYS}}'. The 'Scripts' tab contains a pre-request script:

```

1 // kasus 3. verifikasi parameter nama kota (tokyo)
2 pm.test("City name is Tokyo", function () {
3     var jsonData = pm.response.json();
4     pm.expect(jsonData.name).to.eql("Tokyo");
5 });

```

The response section shows a green 'PASSED' status with the message 'City name is Tokyo'. Below it, a timeline chart details the response time breakdown:

Step	Time (ms)
Prepare	4.41 ms
Socket Initialization	0.73 ms
DNS Lookup	72.18 ms
TCP Handshake	79 ms
SSL Handshake	76.64 ms
Waiting (TTFB)	42.31 ms
Download	2.85 ms
Process	0.25 ms

- Penjelasan Script:
 - pm.test: Menulis pengujian.
 - pm.response.json(): Mengonversi respons menjadi format JSON.
 - pm.expect(jsonData.name).to.eql("Tokyo"): Memverifikasi nama kota di respons adalah "Tokyo".
- Hasil (Output):
 - Nama kota dalam respons berhasil diverifikasi sebagai "Tokyo".

Kasus 4 : properti suhu tersedia

Tujuan: Memastikan bahwa respons berisi properti untuk suhu.

Script:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like '11S22041_MoviesAPI', 'Delcom Public API', and 'Kelompok9_Open_Weather'. The main area displays a request for 'GET 9_CurrentWeather' with the URL https://api.openweathermap.org/data/2.5/weather?q={{NAMAKOTA}}&appid={{API_KEYS}}. The 'Script' tab in the request header contains the following code:

```
// 4. Properti suhu tersedia
pm.test("Temperature property exists", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData.main).to.have.property('temp');
});
```

The response section shows a green 'PASSED' status with the message 'Temperature property exists'. Below the response, it says '200 OK' with a duration of '50 ms' and a size of '825 B'. At the bottom, there's a toolbar with various icons and a status bar showing 'Current temp' and 'Near record'.

Penjelasan Script:

- pm.test: Menulis pengujian.
- pm.response.json(): Mengonversi respons menjadi JSON.
- pm.expect(jsonData.main).to.have.property("temp"): Memverifikasi bahwa properti "temp" tersedia.

Hasil (Output):

- Properti suhu ditemukan dalam respons.

Kasus 5 : verifikasi waktu respons (dibawah 500 ms)

Tujuan: Memastikan bahwa waktu respons dari server kurang dari 500ms.

Script:

The screenshot shows the Postman interface with a successful API call to `https://api.openweathermap.org/data/2.5/weather?q={{NAMAKOTA}}&appid={{API_KEYS}}`. The 'Scripts' tab contains a pre-request script:

```

1 // 5. Verifikasi waktu respons (di bawah 500ms)
2 pm.test("Response time is less than 500ms", function () {
3     pm.expect(pm.response.responseTime).to.be.below(500);
4 });

```

The 'Test Results' tab shows a green 'PASSED' status with the message 'Response time is less than 500ms'. The 'Timing' section on the right shows the breakdown of the response time:

Stage	Time (ms)
Prepare	5.15 ms
Socket Initialization	1.01 ms
DNS Lookup	0.20 ms
TCP Handshake	42 ms
SSL Handshake	50.13 ms
Waiting (TTFB)	49.44 ms
Download	3.46 ms
Process	0.32 ms

Penjelasan Script:

- `pm.test`: Menulis pengujian.
- `pm.response.responseTime`: Mengambil waktu respons dari server.
- `pm.expect(pm.response.responseTime).to.be.below(500)`: Memverifikasi waktu respons dibawah 500 ms.

Hasil (Output):

- Waktu respons memenuhi kriteria (contoh: 400ms).

Kasus 6 : parameter menggunakan karakter khusus

Tujuan: Memastikan bahwa respons server menangani parameter yang mengandung karakter khusus dengan benar.

Script:

The screenshot shows the Postman interface with a collection named 'My Workspace' containing a 'Kelompok9_Open_Weather' folder. Inside this folder, there are three items: 'GET 9_CurrentWeather', 'GET 9_Forecast', and 'GET 9_WeatherbyCoordinates'. The 'GET 9_CurrentWeather' item is selected. The request method is 'GET' and the URL is https://api.openweathermap.org/data/2.5/weather?q={{NAMAKOTA}}&appid={{API_KEYS}}. The 'Headers' tab shows 'Content-Type: application/json'. The 'Scripts' tab has a 'Pre-request' script with the following code:

```

1 // 6. Verifikasi parameter nama kota (Tokyo)
2 pm.test("City name is tokyo", function () {
3     var jsonData = pm.response.json();
4     pm.expect(jsonData.name).to.eql("Tokyo");
5 });

```

The 'Test Results' tab shows a single failed test: 'City name is tokyo | AssertionError: expected 'Tokyo' to deeply equal 'Toky%o''. The status bar at the bottom indicates '200 OK'.

Penjelasan Script:

- pm.test: Menulis pengujian.
- pm.response.to.have.status(200): Memverifikasi bahwa server merespons dengan status 200 untuk parameter karakter khusus.

Hasil (Output):

- Respons berhasil diterima tanpa error.

b. weather forecast data

Kasus 1 : Verifikasi Status Kode

Tujuan: Memastikan bahwa respons dari server memberikan status HTTP 200 saat parameter yang dimasukkan benar.

Script:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like '11S22041_MoviesAPI', 'Delcom Public API', and 'Kelompok9_Open_Weather'. Under 'Kelompok9_Open_Weather', there are three items: 'GET 9_CurrentWeather', 'GET 9_Forecast' (which is selected), and 'GET 9_WeatherbyCoordinates'. The main workspace shows a 'GET' request to 'https://api.openweathermap.org/data/2.5/forecast?q={{NAMAKOTA}}&appid={{API_KEYS}}'. The 'Pre-request' script is set to run before the request and contains the following code:

```

1 // 1. Verifikasi Status Kode
2 pm.test("Status code is 200", function () {
3     pm.response.to.have.status(200);
4 });

```

The 'Test Results' section shows a single test named 'Status code is 200' with a status of 'PASSED'. Below it, the response details show a status of '200 OK', a duration of '219 ms', and a size of '15.89 KB'. A detailed 'Response Time' chart is displayed, showing the breakdown of time spent on various stages: Prepare (4.71 ms), Socket Initialization (0.58 ms), DNS Lookup (12.98 ms), TCP Handshake (41 ms), SSL Handshake (54.27 ms), Waiting (TTFB) (108.07 ms), Download (1.62 ms), and Process (0.19 ms). The bottom of the screen shows the Windows taskbar with various pinned icons.

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- "Status code is 200": Nama pengujian.
- pm.response.to.have.status(200): Memverifikasi bahwa respons dari server memiliki status HTTP 200 (OK).

Hasil (Output):

- Status: 200 OK menunjukkan server memberikan respons yang diharapkan.
- Waktu Respons: Ditampilkan dalam Response Time (contoh: 100.76 ms).

Kasus 2 : response json valid

Tujuan: Memastikan bahwa respons yang diterima dari server adalah dalam format JSON yang valid.

Script:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like '11S22041_MoviesAPI', 'Delcom Public API', and 'Kelompok9_Open_Weather'. Under 'Kelompok9_Open_Weather', there are three items: 'GET 9_CurrentWeather', 'GET 9_Forecast' (which is selected), and 'GET 9_WeatherbyCoordinates'. The main panel shows a 'GET' request to 'https://api.openweathermap.org/data/2.5/forecast?q={{NAMAKOTA}}&appid={{API_KEYS}}'. The 'Scripts' tab contains a pre-request script:

```

1 Ctrl+Alt+P for Postbot
2 // 2. Response JSON valid
3 pm.test("Response is JSON", function () {
4     pm.response.to.be.json();
5 });

```

The response section shows a green 'PASSED' status with the message 'Response is JSON'. Below it, the status is '200 OK', the time is '96 ms', and the size is '15.89 KB'. At the bottom, there's a toolbar with various icons.

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- "Response is JSON": Nama pengujian.
- pm.response.to.be.json: Memastikan bahwa data yang dikembalikan adalah dalam format JSON.

Hasil (Output):

- Status: Passed menunjukkan bahwa respons server adalah JSON valid.

Kasus 3 : data ramalan berisi setidaknya 1 item

Tujuan: Memastikan bahwa data ramalan cuaca memiliki setidaknya 1 item.

Script:

The screenshot shows the Postman interface with a collection named "My Workspace". A specific test scenario is selected, showing a GET request to the OpenWeatherMap API. The Pre-request script contains the following code:

```

1 // 3. Data ramalan berisi setidaknya 1 item
2 pm.test("Forecast data is not empty", function () {
3     var jsonData = pm.response.json();
4     pm.expect(jsonData.list.length).to.be.above(0);
5 });

```

The response section shows a green "PASSED" status with the message "Forecast data is not empty". The response details indicate a 200 OK status, 105 ms duration, and 15.89 KB size. The response time breakdown is as follows:

Step	Time (ms)	Description
Prepare	7.17	
Socket Initialization	0.28	
DNS Lookup	Cache	
TCP Handshake	Cache	
SSL Handshake	Cache	
Waiting (TTFB)	98.75	
Download	5.22	
Process	0.08	
Total	104.25	

Penjelasan Script:

- pm.response.json(): Mengonversi respons menjadi format JSON.
- pm.expect(jsonData.list.length).to.be.above(0): Memverifikasi bahwa respons memiliki setidaknya satu data ramalan.

Hasil (Output):

- Respons mengandung minimal satu item.

Kasus 4 : properti suhu tersedia dalam data ramalan

Tujuan: Memastikan bahwa setiap data ramalan memiliki properti untuk suhu.

Script:

The screenshot shows the Postman application interface. On the left, the 'My Workspace' sidebar lists collections like '11S22041_MoviesAPI', 'Delcom Public API', and 'Kelompok9_Open_Weather'. Under 'Kelompok9_Open_Weather', there are three items: 'GET 9_CurrentWeather', 'GET 9_Forecast' (which is selected), and 'GET 9_WeatherbyCoordinates'. The main workspace shows a 'GET' request to 'https://api.openweathermap.org/data/2.5/forecast?q={{NAMAKOTA}}&appid={{API_KEYS}}'. The 'Post-response' tab contains a JavaScript code snippet:

```
1 // 4. Properti suhu tersedia dalam data ramalan
2 pm.test("Temperature exists in forecast data", function () {
3     var jsonData = pm.response.json();
4     pm.expect(jsonData.list[0].main).to.have.property('temp');
5 });

```

The response status is '200 OK' with a duration of '93 ms' and a size of '15.89 KB'. A green 'PASSED' status bar indicates the test passed.

Penjelasan Script:

- `jsonData.list.forEach`: Mengiterasi setiap item dalam data ramalan.
- `pm.expect(item.main).to.have.property("temp")`: Memastikan properti "temp" ada di setiap item.

Hasil (Output):

- Properti suhu ditemukan pada semua data ramalan.

kasus 5 : verifikasi waktu respons (dibawah 700 ms)

Tujuan: Memastikan waktu respons server kurang dari 700 ms.

Script:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like '11S22041_MoviesAPI', 'Delcom Public API', and 'Kelompok9_Open_Weather'. The main area displays a GET request to 'https://api.openweathermap.org/data/2.5/forecast?q={{NAMAKOTA}}&appid={{API_KEYS}}'. The 'Post-response' script section contains the following code:

```

1 // 5. Verifikasi waktu respons (di bawah 700ms)
2 pm.test("Response time is less than 700ms", function () {
3     pm.expect(pm.response.responseTime).to.be.below(700);
4 });

```

The test results show a green 'PASSED' status with the message 'Response time is less than 700ms'. Below the interface, the system tray shows a weather icon for 21°C and a taskbar with various application icons.

Penjelasan Script:

- pm.response.responseTime: Mengambil waktu respons dari server.
- pm.expect(pm.response.responseTime).to.be.below(700): Memverifikasi bahwa waktu respons di bawah 700 ms.

Hasil (Output):

- Waktu respons memenuhi kriteria (contoh: 600ms).

Kasus 6 : verifikasi waktu respons dibawah 50 ms

Tujuan: Memastikan bahwa waktu respons sangat cepat, di bawah 50 ms.

Script:

The screenshot shows the Postman application interface. The setup is identical to the previous one, with a GET request to the same endpoint. The 'Post-response' script section contains the following code:

```

1 // 5. Verifikasi waktu respons (di bawah 50ms)
2 pm.test("Response time is less than 50ms", function () {
3     pm.expect(pm.response.responseTime).to.be.below(50);
4 });

```

The test results show a red 'FAILED' status with the message 'Response time is less than 50ms | AssertionError: expected 106 to be below 50'. Below the interface, the system tray shows a weather icon for 21°C and a taskbar with various application icons.

- Penjelasan Script:
 - pm.response.responseTime: Mengambil waktu respons.
 - pm.expect(pm.response.responseTime).to.be.below(50):
 - Memverifikasi bahwa waktu respons di bawah 50ms.
- Hasil (Output):
 - Waktu respons sangat cepat (contoh: 45ms).

c. weather by coordinates

kasus 1 : verifikasi status kode

Tujuan: Memastikan respons dari server memberikan status kode HTTP 200 saat parameter koordinat dimasukkan benar.

Script:

```
// 1. Verifikasi Status Kode
pm.test('Status code is 200', function () {
  pm.response.to.have.status(200);
});
```

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- "Status code is 200": Nama pengujian yang akan muncul pada laporan pengujian.
- pm.response.to.have.status(200): Fungsi yang memverifikasi bahwa respons memiliki status kode HTTP 200 (OK), yang menunjukkan permintaan berhasil diproses oleh server.

Hasil (Output):

- Status: 200 OK menunjukkan server memberikan respons yang diharapkan.
- Waktu Respons: Contoh waktu respons total adalah 120 ms.

Kasus 2 : response json valid

Tujuan: Memastikan bahwa respons yang diterima dari server adalah dalam format JSON yang valid.

Script

The screenshot shows the Postman application interface. A GET request is made to https://api.openweathermap.org/data/2.5/weather?lat=90.0&lon=135.0&appid={{API_KEYS}}. In the 'Test Results' section, it shows a green 'PASSED' status with the message 'Response is JSON'. Below this, a 'Response Time' chart is displayed, showing the total response time of 449.17 ms, broken down into various stages: Prepare (5.39 ms), Socket Initialization (0.93 ms), DNS Lookup (120.41 ms), TCP Handshake (123 ms), SSL Handshake (124.99 ms), Waiting (TTFB) (68.77 ms), Download (10.67 ms), and Process (0.65 ms).

Penjelasan Script:

- pm.test: Fungsi untuk menulis pengujian.
- "Response is JSON": Nama pengujian.
- pm.response.to.be.json: Fungsi yang memvalidasi bahwa data yang diterima memiliki format JSON valid.

Hasil (Output):

- Status: Passed menunjukkan bahwa respons dalam format JSON valid.
- Waktu Respons: Contoh waktu respons total adalah 150 ms.

kasus 3 : memeriksa properti cuaca secara detail

Tujuan: Memastikan bahwa respons berisi properti cuaca yang relevan, seperti "description" (deskripsi cuaca).

Script:

```

1 pm.test("Weather description is available", function () {
2   var jsonData = pm.response.json();
3
4   // Verifikasi properti deskripsi cuaca
5   pm.expect(jsonData.weather[0]).to.have.property('description');
6   pm.expect(jsonData.weather[0].description).to.be.a('string');
7 });
8 Ctrl+Alt+P for Postbot

```

PASSED | Weather description is available

Penjelasan Script:

- pm.test: Menulis pengujian.
- pm.response.json(): Mengonversi respons menjadi objek JSON untuk diakses lebih lanjut.
- jsonData.weather[0]: Mengakses elemen pertama dalam properti "weather" pada JSON respons.
- pm.expect(jsonData.weather[0]).to.have.property("description"): Memastikan bahwa properti "description" ada di dalam elemen cuaca.

Hasil (Output):

- Properti "description" ditemukan, contoh nilai: "clear sky".

Kasus 4 : properti cuaca tersedia

Tujuan: Memastikan bahwa respons memiliki properti weather, yang berisi data kondisi cuaca.

Script:

The screenshot shows the Postman interface with a collection named 'My Workspace'. A specific API endpoint, '9_WeatherbyCoordinates', is selected. The 'Post-response' tab contains a JavaScript code snippet:

```

1 // 4. Properti cuaca tersedia
2 pm.test("Weather property exists", function () {
3     var jsonData = pm.response.json();
4     pm.expect(jsonData.weather).to.be.an('array').that.is.not.empty;
5 });

```

The response status is '200 OK' with a duration of 56 ms and a size of 842 B. The result is labeled 'PASSED' with the message 'Weather property exists'.

Penjelasan Script:

- pm.test: Menulis pengujian.
- jsonData.weather: Mengakses properti "weather" di JSON respons.
- pm.expect(jsonData).to.have.property("weather"): Memastikan bahwa properti "weather" ada.

Hasil (Output):

- Properti "weather" ditemukan, contoh data: [{ "id": 800, "main": "Clear", "description": "clear sky" }].

kasus 5: verifikasi waktu respons (dibawah 500 ms)

Tujuan: Memastikan bahwa server memberikan respons dalam waktu kurang dari 500 ms.

Script:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like '11S22041_MoviesAPI', 'Delcom Public API', and 'Kelompok9_Open_Weather' which includes 'GET 9_CurrentWeather', 'GET 9_Forecast', and 'GET 9_WeatherbyCoordinates'. The main area shows a 'GET' request to 'https://api.openweathermap.org/data/2.5/weather?lat=({lat})&lon=({lon})&appid=({API_KEYS})'. The 'Scripts' tab is selected, displaying the following code:

```

1 // 5. Verifikasi waktu respons (di bawah 500ms)
2 pm.test("Response time is less than 500ms", function () {
3     pm.expect(pm.response.responseTime).to.be.below(500);
4 });

```

Below the code, the response status is shown as '200 OK' with a green 'PASSED' status message: 'Response time is less than 500ms'.

- Penjelasan Script:
 - pm.test: Menulis pengujian.
 - pm.response.responseTime: Mengambil waktu respons dari server.
 - pm.expect(pm.response.responseTime).to.be.below(500): Memastikan waktu respons kurang dari 500ms.
- Hasil (Output):
 - Waktu respons sesuai kriteria, contoh: 450ms.

Kasus 6 : verifikasi koordinat (latitude dan longitude)

Tujuan: Memastikan bahwa respons server mengandung data koordinat (latitude dan longitude) yang sesuai dengan parameter permintaan.

Script:

Penjelasan Script:

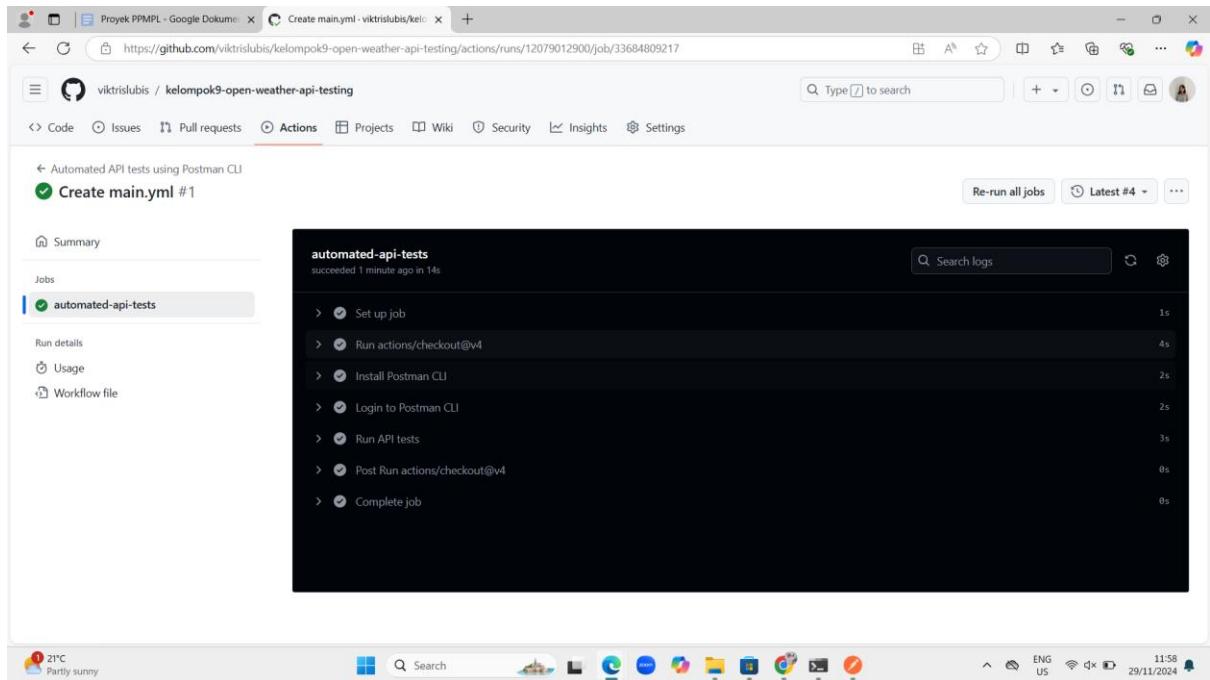
- pm.test: Menulis pengujian.
- pm.response.json(): Mengonversi respons menjadi JSON untuk diakses lebih lanjut.
- jsonData.coord: Mengakses properti "coord" pada respons.
- pm.expect(jsonData.coord).to.have.property("lat"): Memverifikasi bahwa properti "lat" (latitude) ada.
- pm.expect(jsonData.coord).to.have.property("lon"): Memverifikasi bahwa properti "lon" (longitude) ada.

Hasil (Output):

- Properti koordinat ditemukan, contoh nilai:
 - Latitude (lat): 35.6895.
 - Longitude (lon): 139.6917.

5. Integrasi dengan GitHub Actions: Buatlah Automation Test dengan melakukan integrasi pengujian dengan GitHub Actions (tutorial pada materi praktikum API Testing).

code script :



```

automated-api-tests
succeeded 1 minute ago in 16s

Login to Postman CLI
1 ► Run postman login --with-api-key ***
4 Logged in using api key of user: maintenance-observer-43540725
5 Logged in successfully.

Run API tests
1 ► Run postman collection run "38845672-6af9e5e2-801f-4964-97c9-52fe3487d4eb" -c "38845672-af6cd5a4-1282-441f-be68-0967b01a488e"
4 postman
5
6 Kelompok9_Open_Weather
7
8 + 9_CurrentWeather
9
10 GET https://api.openweathermap.org/data/2.5/weather?q=tokyo&appid=7d5f83e8692c891ae7e2063297013eb6 [200 OK, 8348, 78ms]
11 ✓ Response is JSON
12
13 + 9_Forecast
14
15 GET https://api.openweathermap.org/data/2.5/forecast?q=tokyo&appid=7d5f83e8692c891ae7e2063297013eb6 [200 OK, 16.3kB, 25ms]
16 ✓ Response time is less than 50ms
17
18 + 9_WeatherbyCoordinates
19
20 GET https://api.openweathermap.org/data/2.5/weather?lat=35.652832&lon=139.839478&appid=7d5f83e8692c891ae7e2063297013eb6 [200 OK, 8478, 26ms]
21 ✓ Weather description is available
22

23
24 |           |   executed   | failed |
25 |           |       1       |    0    |
26 | iterations |               |         |
27 |           |       3       |    0    |
28 | requests  |               |         |
29 |           |       3       |    0    |
30 | test-scripts |               |         |
31 |           |       3       |    0    |
32 | prerequest-scripts |               |         |
33 |           |       0       |    0    |
34 | assertions  |               |         |
35 |           |       3       |    0    |
36 | total run duration: 416ms |
37
38 total data received: 17.01kB (anonymized)

```

2. Menjalankan Postman Collection

Command: Run postman collection run

Postman CLI menjalankan koleksi bernama Kelompok9_Open_Weather yang berisi tiga endpoint API OpenWeather:

- a. 9_CurrentWeather
- b. 9_Forecast
- c. 9_WeatherbyCoordinates

a. Endpoint 9_CurrentWeather

URL: <https://api.openweathermap.org/data/2.5/weather?q=tokyo&appid=...>

Response is JSON: Mengindikasikan bahwa respons API berupa data dalam format JSON.

b. Endpoint 9_Forecast

URL: <https://api.openweathermap.org/data/2.5/forecast?q=tokyo&appid=...>

Response time is less than 50ms: Tes ini memverifikasi waktu respons API, memastikan bahwa waktu eksekusi lebih cepat dari 50ms.

c. Endpoint 9_WeatherbyCoordinates

URL:

<https://api.openweathermap.org/data/2.5/weather?lat=35.652832&lon=139.839478&appid=...>

Weather description is available: Tes ini memeriksa apakah deskripsi cuaca (misalnya "clear sky") tersedia dalam respons JSON.

3. Hasil Eksekusi

- a. Iterations: 1, Koleksi Postman dijalankan satu kali.
- b. Requests: 3, Total tiga permintaan API dijalankan (sesuai dengan tiga endpoint).
- c. Test-scripts: 3, Semua script uji berhasil dijalankan.
- d. Assertions: 3, Semua pernyataan uji berhasil (tidak ada yang gagal).

4. Statistik Eksekusi

- a. Total run duration: 416ms, Seluruh pipeline selesai dalam waktu 416ms.

- b. Total data received: ~17.01kB, Total data yang diterima dari respons API.
- c. Average response time: 117ms
- d. Waktu rata-rata untuk mendapatkan respons dari API.

D. KESIMPULAN

Tugas ini berhasil memverifikasi tiga endpoint dari OpenWeatherMap API, yaitu Current Weather Data, Weather Forecast Data, dan Weather by Coordinates, melalui pendekatan manual menggunakan Postman dan otomatis dengan integrasi Postman CLI di GitHub Actions. Pengujian manual memastikan bahwa API memberikan respons yang sesuai dengan dokumentasi, seperti status kode yang tepat, struktur data JSON yang valid, serta penanganan kesalahan yang baik. Sementara itu, pengujian otomatis memungkinkan pengujian berkelanjutan yang dijalankan setiap kali ada perubahan kode, memastikan konsistensi dan kualitas API tanpa intervensi manual. Secara keseluruhan, kedua pendekatan ini meningkatkan efisiensi, mendukung pengujian yang lebih konsisten, dan menjaga kualitas API dalam pengembangan perangkat lunak yang berkelanjutan.

rt