

**PRATIUM PEMOGRAMAN JARINGAN**

**“Laporan Project 3”**



**Dosen Pengampu:**

Ade Kurniawan, S.Pd., M.Pd.T

**Oleh:**

Najwa Alawiyah Siregar

22346040

**PROGRAM STUDI INFORMATIKA  
DEPARTEMEN TEKNIK ELEKTRONIKA  
FAKULTAS TEKNIK  
UNIVERSITAS NEGERI PADANG  
2024**

## **KATA PENGANTAR**

Puji syukur kita panjatkan kehadiran Allah Swt. yang telah memberikan rahmat dan hidayah-Nya sehingga saya dapat menyelesaikan tugas yang berjudul “HTTP Request and API” ini tepat pada waktunya.

Adapun tujuan dari penulisan dari laporan ini adalah untuk memenuhi tugas pada mata kuliah Praktikum Pemograman Jaringan. Selain itu, laporan ini juga bertujuan untuk menambah wawasan tentang HTTP Request and API bagi para pembaca dan juga bagi penulis.

Terlebih dahulu, saya mengucapkan terima kasih kepada Bapak Ade Kurniawan, S.Pd., M.Pd.T, selaku Praktikum Pemograman Jaringan yang telah memberikan tugas ini sehingga dapat menambah pengetahuan dan wawasan sesuai dengan bidang studi yang saya tekuni ini.

Kemudian, saya menyadari bahwa tugas yang saya tulis ini masih jauh dari kata sempurna. Oleh karena itu, kritik dan saran yang membangun saya butuhkan demi kesempurnaan laporan ini.

Painan, 29 September 2024

Najwa Alawiyah Siregar

## DAFTAR ISI

<b>COVER LAPORAN TUGAS 3 .....</b>	<b>Error! Bookmark not defined.</b>
<b>KATA PENGANTAR .....</b>	<b>2</b>
<b>DAFTAR ISI .....</b>	<b>3</b>
<b>BAB I .....</b>	<b>1</b>
<b>1.1. Latar Belakang.....</b>	<b>1</b>
<b>1.2. Rumusan Masalah .....</b>	<b>2</b>
<b>1.3. Tujuan.....</b>	<b>2</b>
<b>BAB II.....</b>	<b>3</b>
<b>2.1. HTTP Request.....</b>	<b>3</b>
<b>2.2. API (Application Programming Interface) .....</b>	<b>4</b>
<b>2.3. Langkah Kerja Perintah Git.....</b>	<b>6</b>
<b>BAB III .....</b>	<b>31</b>
<b>3.1. Kesimpulan.....</b>	<b>31</b>
<b>3.2. Daftar Pustaka .....</b>	<b>31</b>

# **BAB I**

## **PENDAHULUAN**

### **1.1. Latar Belakang**

Perkembangan teknologi informasi dan komunikasi telah membawa perubahan besar dalam berbagai aspek kehidupan manusia. Salah satu yang paling signifikan adalah kemudahan akses terhadap data dan informasi melalui jaringan internet. Di balik teknologi ini, terdapat berbagai protokol dan mekanisme yang memungkinkan pengiriman dan penerimaan data antara perangkat yang terhubung di seluruh dunia. Salah satu protokol yang paling penting adalah HTTP (Hypertext Transfer Protocol), yang memungkinkan komunikasi antara klien (seperti browser atau aplikasi) dengan server untuk mengakses berbagai sumber daya di internet, seperti halaman web, gambar, atau data lainnya.

Seiring dengan berkembangnya kebutuhan untuk mengintegrasikan berbagai layanan dan data, penggunaan API (Application Programming Interface) menjadi semakin umum. API memungkinkan pengembang untuk mengakses fitur atau data dari aplikasi lain tanpa harus memahami detail implementasi dari aplikasi tersebut. Contoh nyata dari pemanfaatan API adalah aplikasi cuaca, di mana data cuaca real-time dapat diakses dari penyedia layanan cuaca seperti Weatherstack melalui API dan ditampilkan kepada pengguna dalam antarmuka aplikasi yang ramah pengguna.

Dalam konteks pengembangan aplikasi berbasis jaringan, pemahaman mengenai HTTP Request dan API sangatlah penting. HTTP Request adalah cara utama untuk mengirimkan permintaan data dari klien ke server, sedangkan API memungkinkan aplikasi untuk berkomunikasi satu sama lain dan mengintegrasikan berbagai layanan dari pihak ketiga. Pemanfaatan kedua teknologi ini menjadi sangat relevan dalam pengembangan aplikasi yang membutuhkan akses ke data eksternal, seperti aplikasi ramalan cuaca, yang memanfaatkan data dari server cuaca untuk memberikan informasi terkini kepada pengguna.

Pada laporan ini, dibahas penerapan HTTP Request dan API Weatherstack dalam pengembangan aplikasi cuaca. Dengan menggunakan API, aplikasi dapat mengambil data cuaca secara real-time berdasarkan lokasi tertentu dan menampilkannya kepada pengguna dalam bentuk yang mudah dipahami. Selain itu, dalam pengembangan perangkat lunak, pengelolaan proyek menjadi hal penting yang harus diperhatikan, terutama dalam tim pengembang. Oleh karena itu, penggunaan Git, sebuah sistem kontrol versi yang memungkinkan pengembang untuk mengelola dan melacak perubahan kode, menjadi hal yang tak terpisahkan dalam proses pengembangan aplikasi.

Penguasaan konsep-konsep tersebut menjadi sangat penting bagi mahasiswa informatika, terutama dalam menghadapi tantangan di dunia kerja yang menuntut

kemampuan dalam mengembangkan aplikasi yang fungsional dan efisien. Oleh karena itu, dalam praktikum ini, mahasiswa diajak untuk memahami lebih dalam bagaimana cara kerja HTTP Request, API, dan penggunaan Git dalam pengembangan aplikasi, serta menerapkannya dalam pembuatan aplikasi cuaca sederhana. Dengan pengalaman praktis ini, mahasiswa diharapkan mampu mengaplikasikan pengetahuan mereka dalam pengembangan proyek berbasis jaringan di masa depan.

Latar belakang ini menguraikan pentingnya pemahaman HTTP Request, API, dan pengelolaan proyek menggunakan Git, serta bagaimana topik tersebut diterapkan dalam konteks pengembangan aplikasi cuaca, yang relevan dengan dunia pengembangan aplikasi modern.

### **1.2. Rumusan Masalah**

Berdasarkan latar belakang yang telah dijelaskan, rumusan masalah dalam laporan ini adalah sebagai berikut:

1. Bagaimana cara kerja HTTP Request dalam pengambilan data dari server?
2. Bagaimana proses penggunaan API dalam mengakses data cuaca secara real-time?
3. Bagaimana langkah-langkah integrasi data dari API Weatherstack ke dalam aplikasi cuaca yang dibuat?
4. Bagaimana cara mengimplementasikan Git untuk pengelolaan proyek pemrograman jaringan?

### **1.3. Tujuan**

Tujuan dari laporan ini adalah:

1. Memahami proses HTTP Request dalam komunikasi antara klien dan server.
2. Mempelajari cara penggunaan API untuk mengakses dan menampilkan data cuaca dalam aplikasi.
3. Mengimplementasikan API Weatherstack untuk mendapatkan informasi cuaca secara real-time.
4. Menerapkan pengelolaan proyek pemrograman jaringan menggunakan Git.

## BAB II

### PEMBAHASAN

#### 2.1. HTTP Request

HTTP (Hypertext Transfer Protocol) Request adalah cara utama untuk mengirim permintaan dari klien (biasanya web browser atau aplikasi) ke server web untuk mengakses halaman web atau sumber daya lainnya.



**Gambar 1.** Ilustrasi HTTP Request Proses kerja HTTP Request dapat dijelaskan dalam beberapa langkah utama:

1. **Inisiasi Permintaan:**

- Klien (biasanya web browser) menginisiasi permintaan dengan membuat koneksi ke server web. Ini bisa dilakukan dengan menggunakan protokol TCP/IP (Transmission Control Protocol/Internet Protocol).
- Klien kemudian menentukan jenis permintaan yang ingin dilakukan, seperti GET (mengambil data), POST (mengirim data), PUT (mengganti data), DELETE (menghapus data), dll.
- Klien juga menyediakan alamat URL atau URI (Uniform Resource Identifier) yang menunjuk ke sumber daya yang diinginkan di server.

2. **Pembentukan Request:**

- Klien membuat pesan HTTP Request yang berisi informasi yang diperlukan untuk permintaan tersebut. Pesan ini terdiri dari beberapa komponen utama:
- Metode (GET, POST, dll.).
- URI (alamat sumber daya yang diminta).
- Versi protokol HTTP (misalnya, HTTP/1.1).
- Header HTTP (informasi tambahan seperti tipe konten yang diinginkan, cookie, dan lain-lain).
- Tubuh permintaan (data tambahan jika diperlukan, seperti data formulir dalam kasus POST).

3. **Kirim Request:**

Klien mengirim pesan HTTP Request ke server melalui koneksi yang telah dibuat sebelumnya. Ini melibatkan pengiriman pesan melalui protokol TCP/IP ke alamat IP server yang sesuai dengan URI yang diminta.

4. **Pengolahan Request Server:**

- Server web menerima pesan HTTP Request dari klien. - Server kemudian memproses permintaan sesuai dengan metode yang ditentukan (GET, POST, dll.) dan URI yang disediakan.
- Server dapat melakukan berbagai tugas, seperti mengambil data dari basis data, menjalankan aplikasi, atau menghasilkan halaman web dinamis.

5. **Pengiriman Respons:**

- Setelah selesai memproses permintaan, server mengirim pesan HTTP Response kembali ke klien.
- Pesan HTTP Response juga terdiri dari beberapa komponen utama:
- Kode status (misalnya, 200 OK untuk permintaan yang berhasil atau 404 Not Found jika sumber daya tidak ditemukan).
- Header HTTP (informasi tambahan seperti tipe konten respons, cookie, dan lain-lain).
- Tubuh respons (data yang diminta atau respons lainnya).

6. **Penerimaan dan Tindak Lanjut Klien:**

- Klien menerima pesan HTTP Response dari server.
- Klien kemudian menginterpretasikan respons tersebut, memproses data (jika ada) dan menampilkan halaman web atau tindakan lainnya sesuai dengan respons tersebut.
- Klien juga dapat memutuskan untuk mengirim permintaan tambahan (seperti mengambil gambar atau file tambahan) jika diperlukan untuk menampilkan halaman sepenuhnya.

7. **Penutupan Koneksi :**

Setelah selesai, koneksi antara klien dan server bisa ditutup, tergantung pada aturan dan pemrograman aplikasi.

## 2.2. API (Application Programming Interface)

API adalah sebuah set aturan dan protokol yang memungkinkan berbagai perangkat lunak atau aplikasi untuk berkomunikasi satu sama lain. API adalah cara bagi pengembang perangkat lunak untuk mengintegrasikan fungsionalitas dari satu aplikasi ke dalam aplikasi lainnya, membuat aplikasi tersebut dapat bekerja bersama dan berbagi data atau layanan. Berikut adalah beberapa konsep dasar tentang bagaimana API bekerja:

1. **Abstraksi:** API menyediakan tingkat abstraksi atau antarmuka yang didefinisikan dengan jelas untuk berinteraksi dengan suatu sistem atau layanan. Ini berarti pengembang tidak perlu tahu detail internal dari sistem atau layanan yang mereka gunakan; mereka hanya perlu tahu cara menggunakan API tersebut.
2. **Kegunaan:** API digunakan untuk berbagai tujuan, termasuk mengakses data, berkomunikasi dengan perangkat keras, mengintegrasikan layanan pihak ketiga,

dan banyak lagi. Sebagai contoh, API Google Maps memungkinkan pengembang untuk mengintegrasikan peta dan data lokasi ke dalam aplikasi mereka.

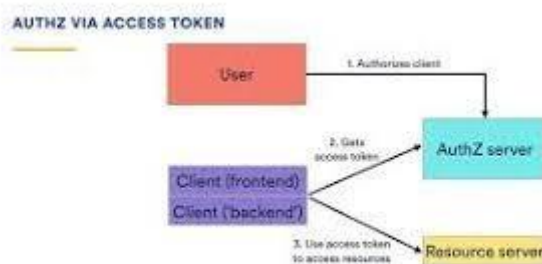
3. **Request-Response Model:** Sebagian besar API bekerja berdasarkan model permintaan-respons. Pengembang mengirimkan permintaan (request) kepada API dengan parameter yang sesuai, dan API memberikan respons (response) dengan data atau hasil yang diminta. Contoh umum adalah API REST (Representational State Transfer) yang menggunakan HTTP sebagai protokol komunikasi

Client-server architecture. API



**Gambar 2.** Ilustrasi Request-responde model pada API

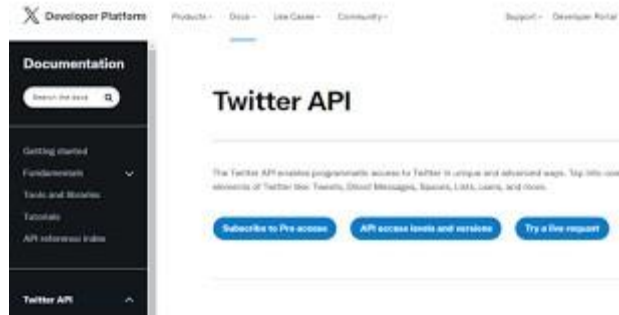
4. **Endpoint:** API memiliki berbagai endpoint, yang merupakan URL atau alamat yang spesifik untuk setiap fungsi atau aksi yang dapat diambil oleh API. Misalnya, dalam API Twitter, Anda mungkin memiliki endpoint untuk mengirim tweet, mengambil daftar pengikut, atau mencari tweet.
5. **Format Data:** Data yang dikirimkan dan diterima melalui API biasanya dalam format tertentu, seperti JSON (JavaScript Object Notation) atau XML (eXtensible Markup Language). Ini memungkinkan data untuk disusun secara terstruktur dan mudah dipahami oleh aplikasi.
6. **Autentikasi dan Otorisasi:** Untuk melindungi data dan layanan, API biasanya memiliki mekanisme autentikasi dan otorisasi. Autentikasi memverifikasi identitas pengguna atau aplikasi yang mengakses API, sementara otorisasi mengendalikan akses ke sumber daya atau layanan tertentu berdasarkan izin yang diberikan.



**Gambar 3.** Ilustrasi Autentikasi dan Otorisasi API

7. **Dokumentasi:** Pengembang API menyediakan dokumentasi yang jelas untuk pengguna API. Dokumentasi ini menjelaskan bagaimana API berfungsi, endpoint yang tersedia, format permintaan dan respons, dan cara mengautentikasi.



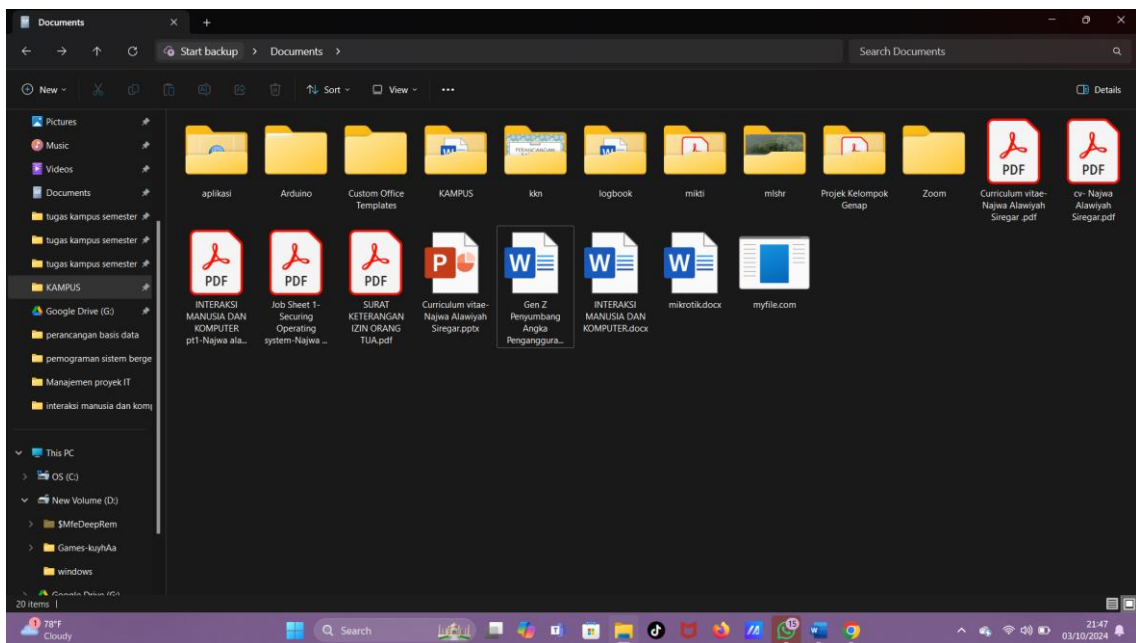


Gambar 4. Dokumentasi API Twittter melalui

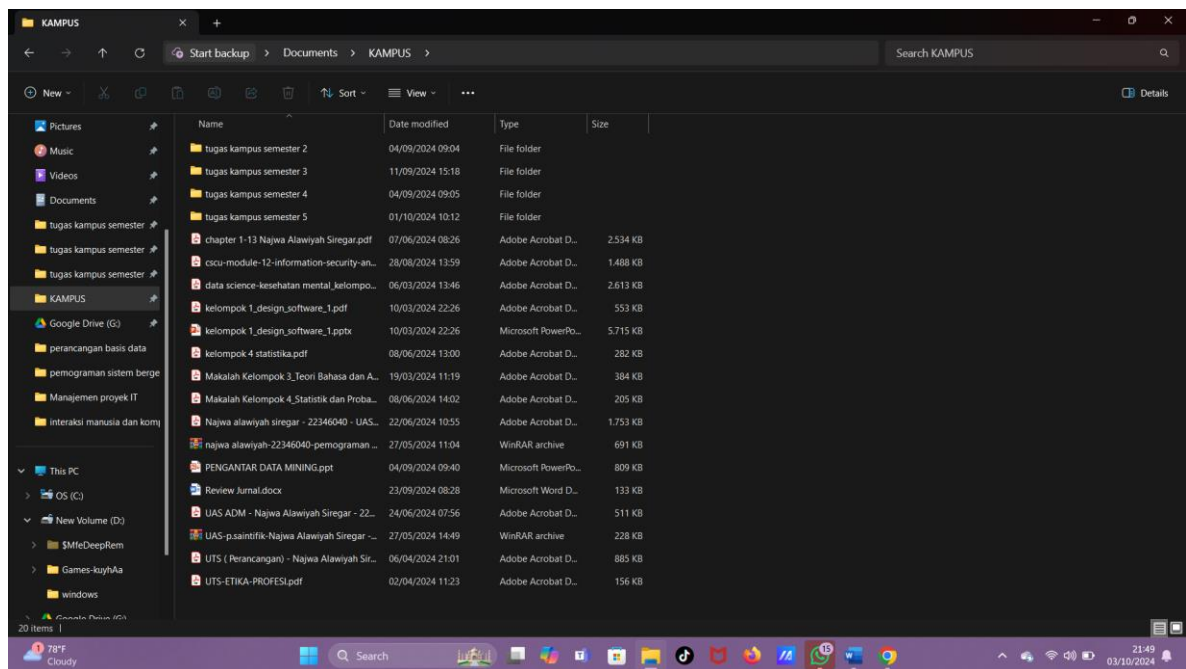
<https://developer.twitter.com/en/docs/twitter-api> Dalam dunia pengembangan perangkat lunak, API memiliki peran yang sangat penting karena memungkinkan aplikasi yang berbeda dibangun secara modular, mempercepat pengembangan, dan memungkinkan integrasi yang lebih mudah antara berbagai sistem. Pengembang dapat menggunakan API dari penyedia layanan lain untuk memperluas fungsionalitas aplikasi mereka tanpa harus membangun semuanya dari awal.

## 2.3. Langkah Kerja Perintah Git

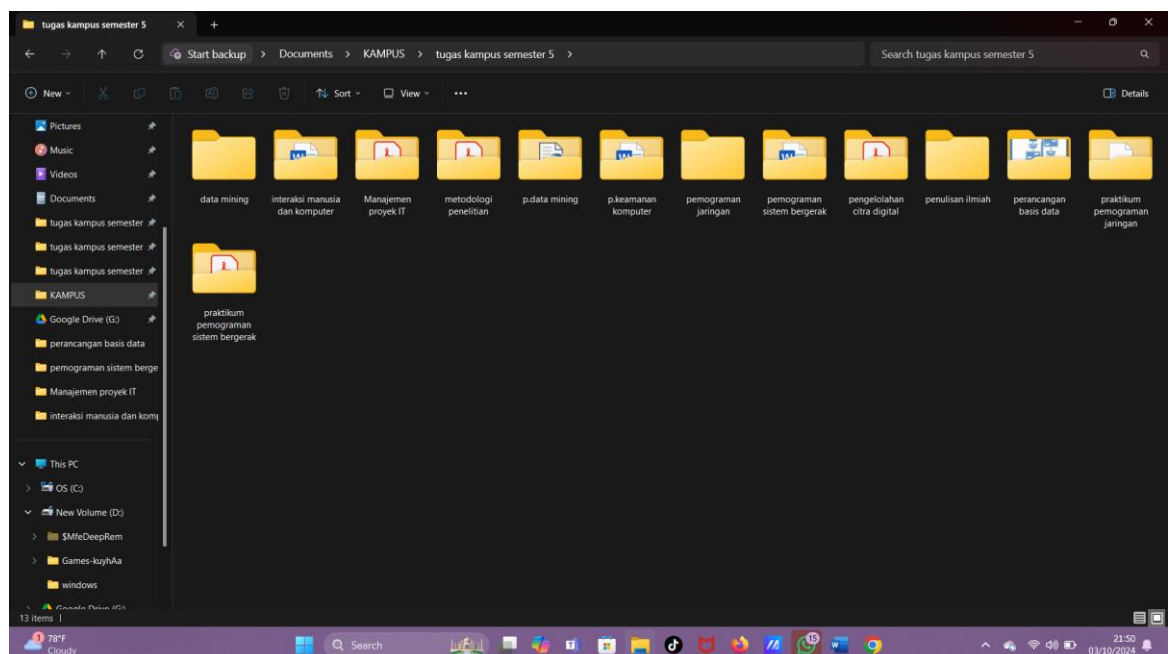
1. Pertama kita membuka file explore, setelah kita berada di file expoler kita memilih file yang tersimpan.



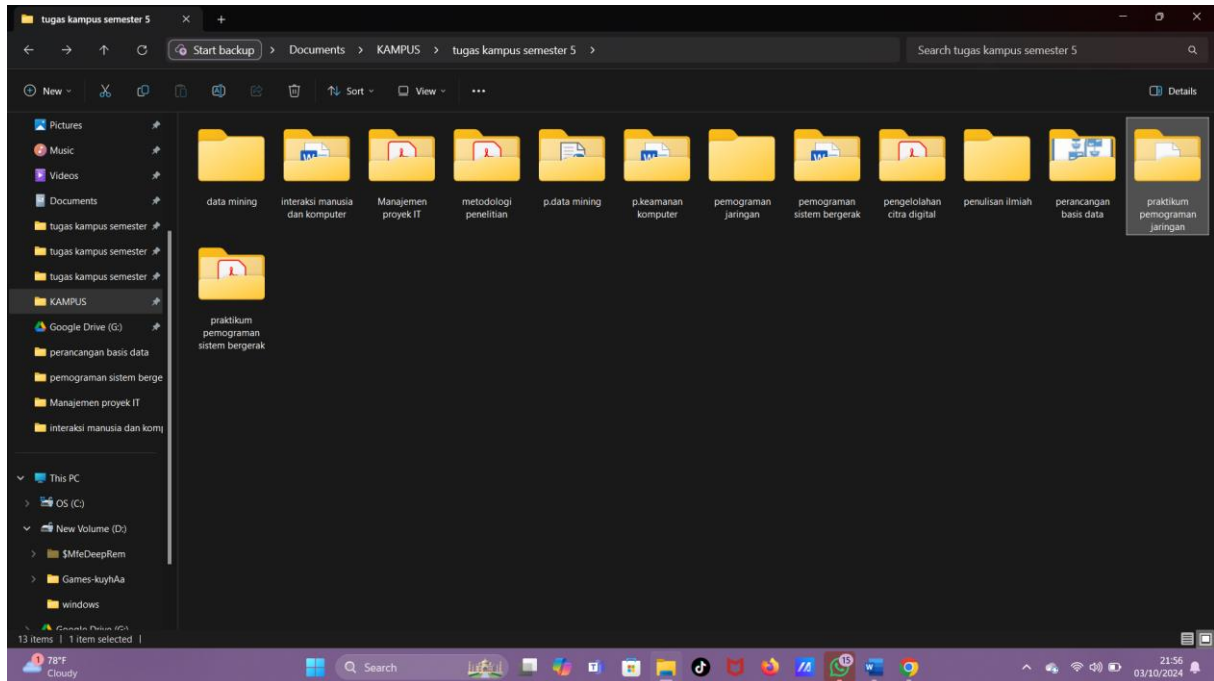
2. Selanjutnya kita membuka folder yang kita inginkan, kemudian kita pilih folder *KAMPUS*.



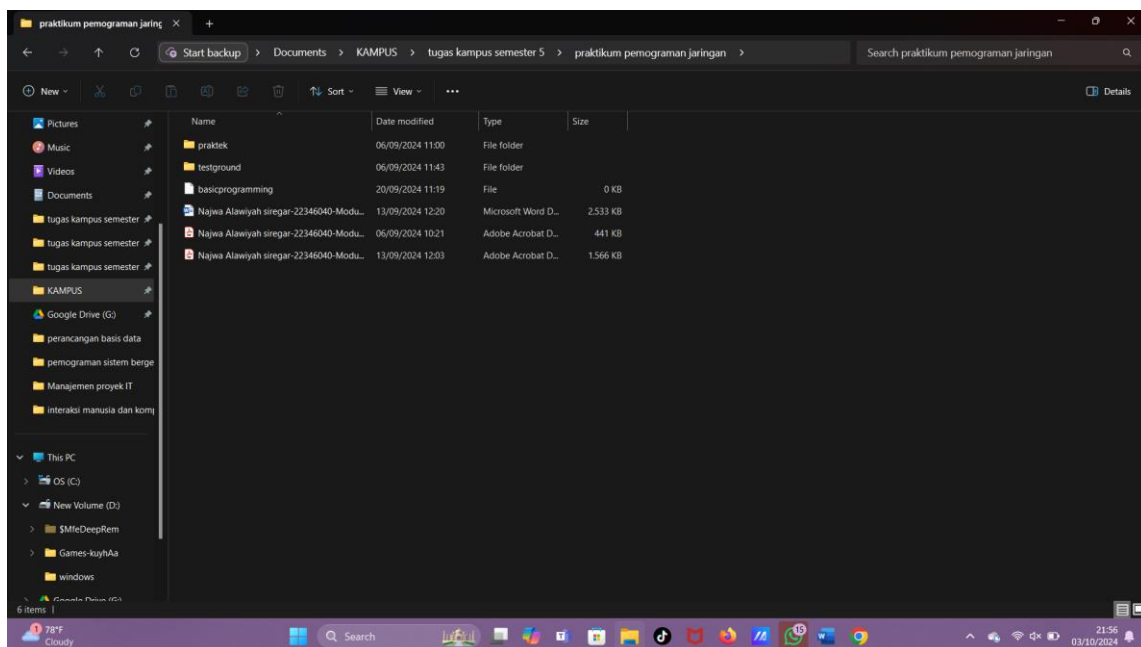
3. Kemudian setelah kita memilih folder *KAMPUS*, selanjutnya kita pilih folder yang kita inginkan, pilih *Tugas Kampus Semester 5*.



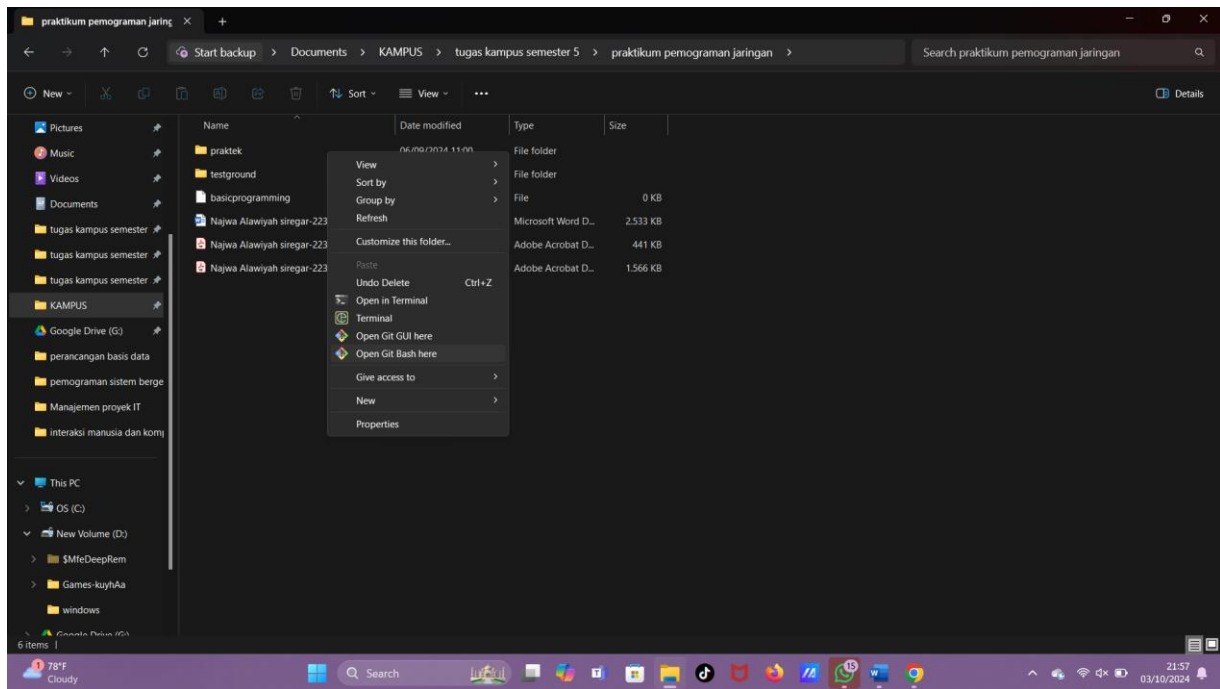
4. Kemudian setelah kita memilih folder Tugas Kampus *Semester 5*, selanjutnya kita pilih folder yang kita inginkan, pilih *Pratikum Pemograman Jaringan*.



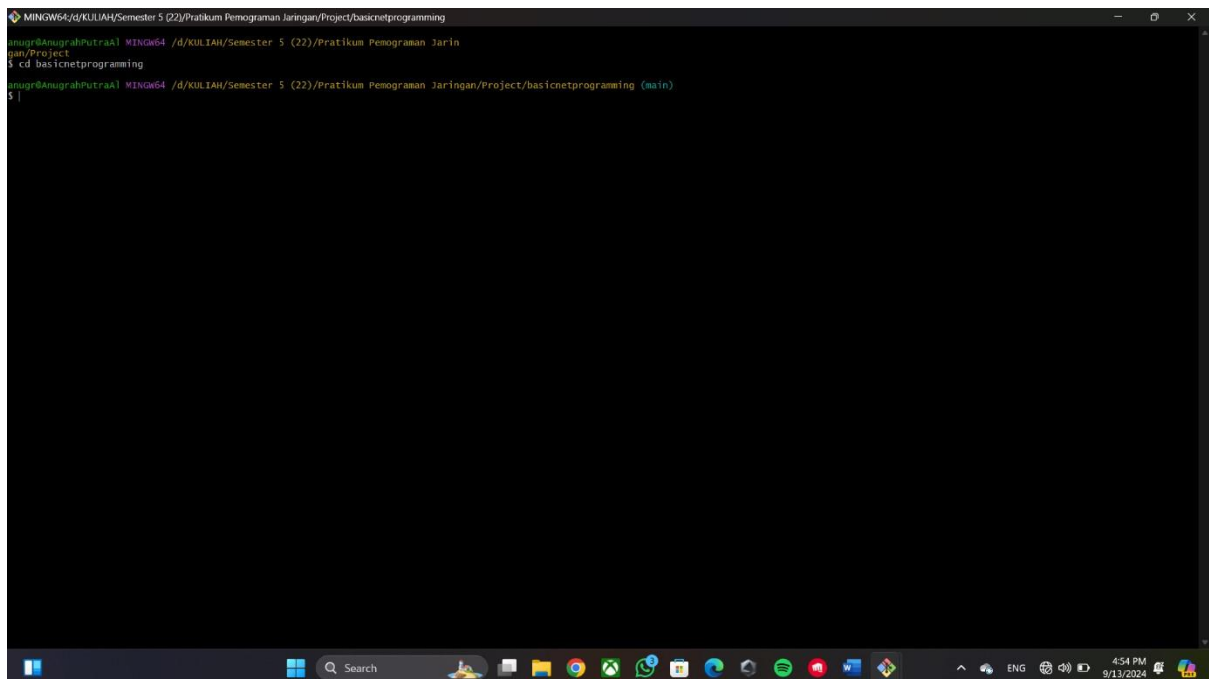
5. Setelah sudah membuka folder yang kita inginkan, lalu kita buat folder baru yang bernama *Praktek*



6. Kemudian setelah kita membuat folder baru, lalu kita pilih folder baru yang telah kita buat, dan click kanan kemudian pilih *Show more options*, kemudian kita di arahkan ke menu selanjutnya, lalu click *Open Git Bash Here*.



7. Setelah kita mencloning link yang tadi, kemudian kita mengetikan `cd basicnetprogramming` yang bertujuan untuk membuka *basicnetprogramming*.



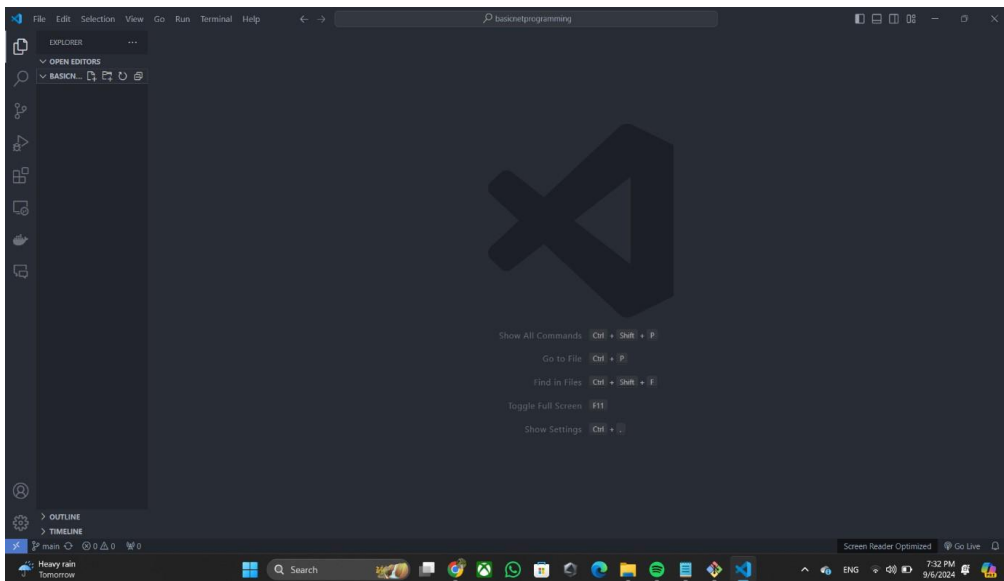
8. Kemudian kita membuat syntax `cd ..` yang fungsi nya untuk keluar dari folder kita sebelumnya.

```
MINGW64/d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jaringan/Project
anugraAnugraPutraA1 MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jari
n/Project
$ cd basicnetprogramming
anugraAnugraPutraA1 MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jaringan/Project/basicnetprogramming (main)
$ cd ..
anugraAnugraPutraA1 MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jaringan/Project
$
```

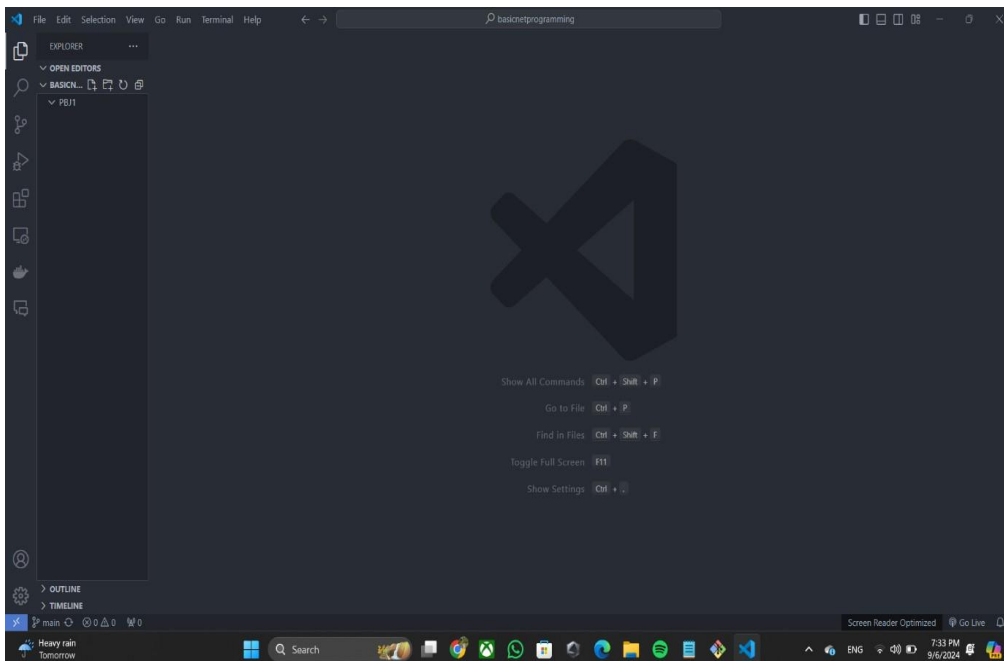
9. Setelah kita membuat syntax sebelumnya, kita lanjutkan untuk membuat syntax baru yaitu *code basicnetprogramming*, yang bertujuan untuk menampilkan code di *visual studio code*.

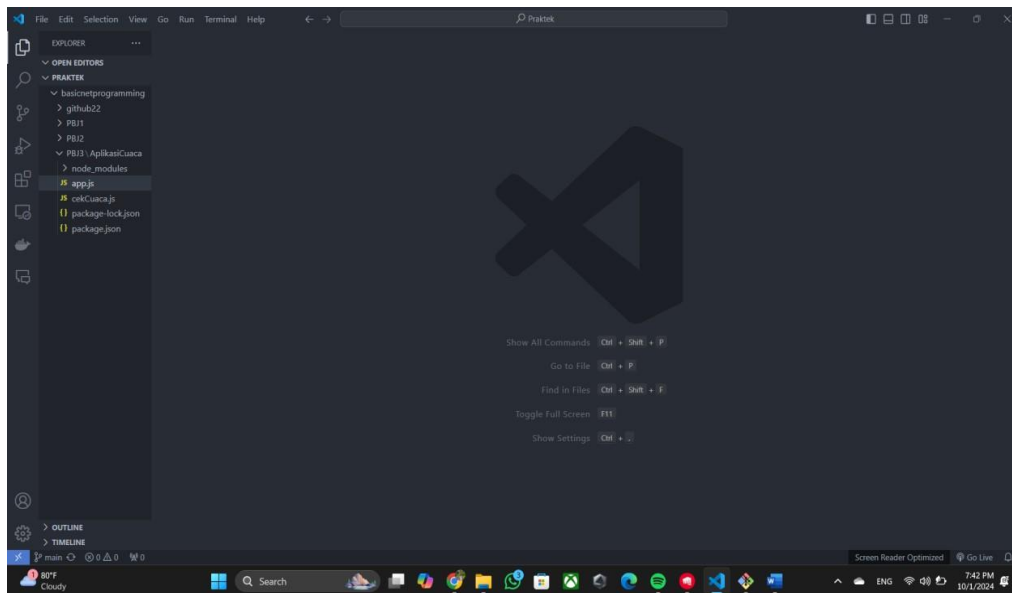
```
MINGW64/d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jaringan/Project
anugraAnugraPutraA1 MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jari
n/Project
$ cd basicnetprogramming
anugraAnugraPutraA1 MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jaringan/Project/basicnetprogramming (main)
$ cd ..
anugraAnugraPutraA1 MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jaringan/Project
$ code basicnetprogramming
anugraAnugraPutraA1 MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jaringan/Project
$
```

10. Tampilan setelah kita melakukan perintah sebelumnya.



- Langkah selanjutnya kita membuat folder di dalam folder yang bernama *PBJ3*. Kemudian setelah membuat folder dalam folder yang telah kita buat yaitu Aplikasi Cuaca, langkah berikutnya kita membuat suatu file baru yang bernama (**app.js** dan **CekCuac.js**).



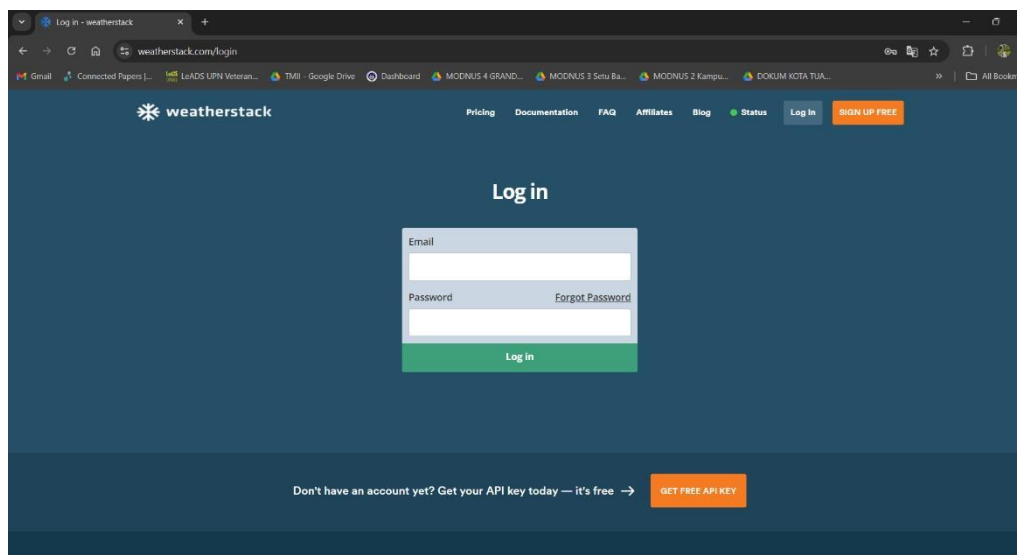


## 12. Latihan

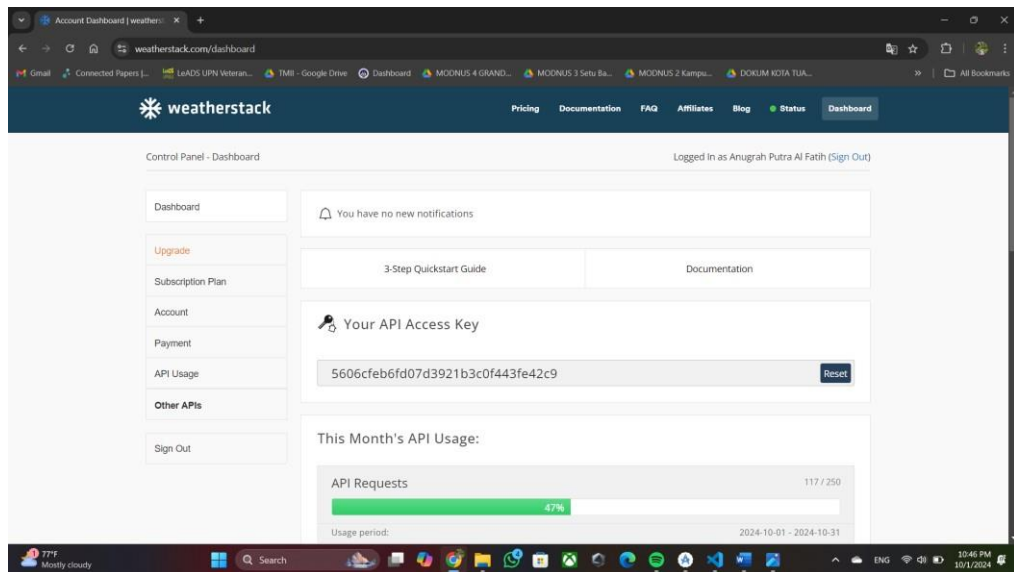
### A. HTTP Request & API

#### a. Mendapatkan API AccessKey

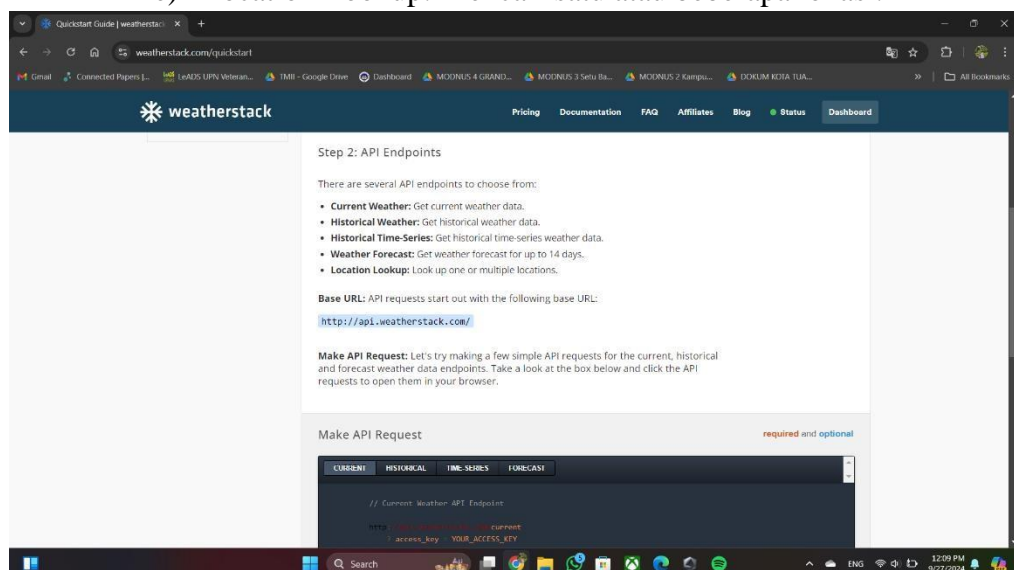
1. Buatlah akun di <https://weatherstack.com/>. Bagian company details dapat dikosongkan.



2. Setelah itu silakan login dan anda akan masuk ke dashboard seperti gambar dibawah ini. Silakan perhatikan kode pada bagian **“Your API Secret key”** karena anda akan menggunakannya didalam program.

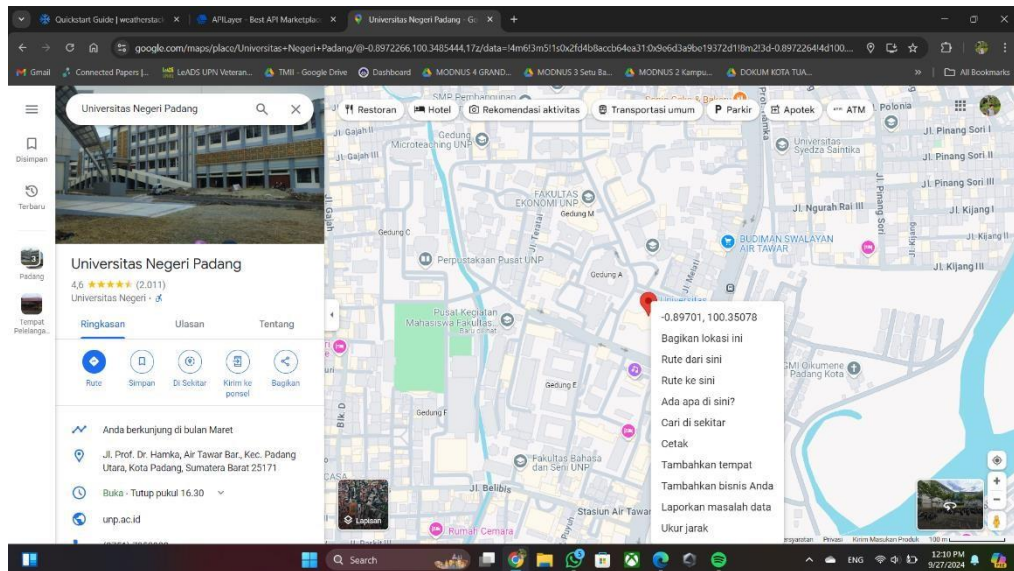


3. Klik 3-Step Quickstart Guide dan kemudian perhatikan base URL <http://api.weatherstack.com/> pada **Step 2: API Endpoints**. URL ini akan digunakan sebagai base URL untuk melakukan request ke API Weatherstack. Weatherstack menyediakan beberapa API Endpoints untuk mendapatkan data terkait cuaca diantaranya:
  - a) Current Weather: Mendapatkan data cuaca terbaru
  - b) Historical Weather: Mendapatkan data historis cuaca
  - c) Historical Time-Series: Mendapatkan data historis cuaca dalam rangkaian waktu tertentu.
  - d) Weather Forecast: Mendapatkan data ramalan cuaca hingga 14 hari.
  - e) Location Lookup: Mencari satu atau beberapa lokasi.

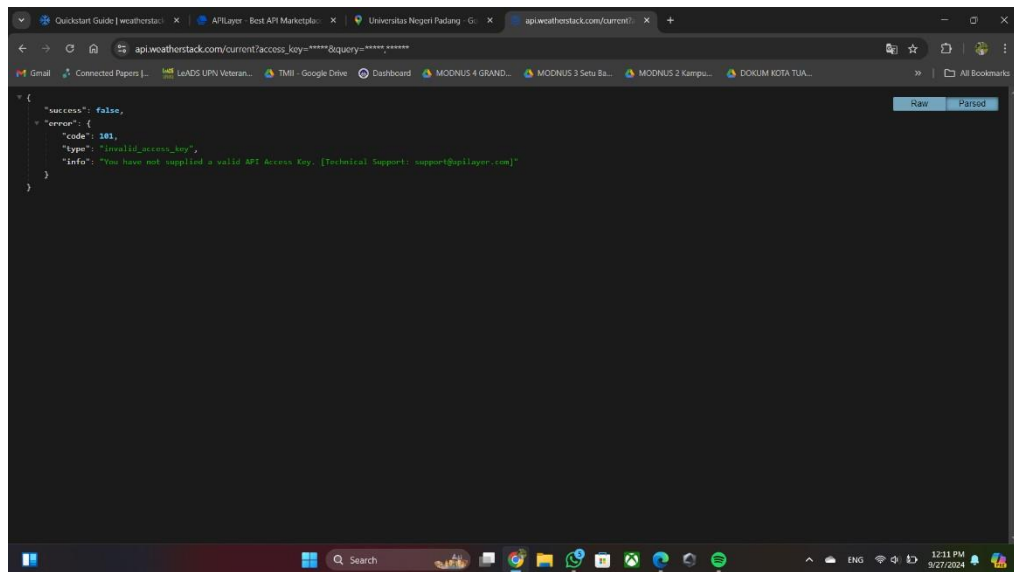




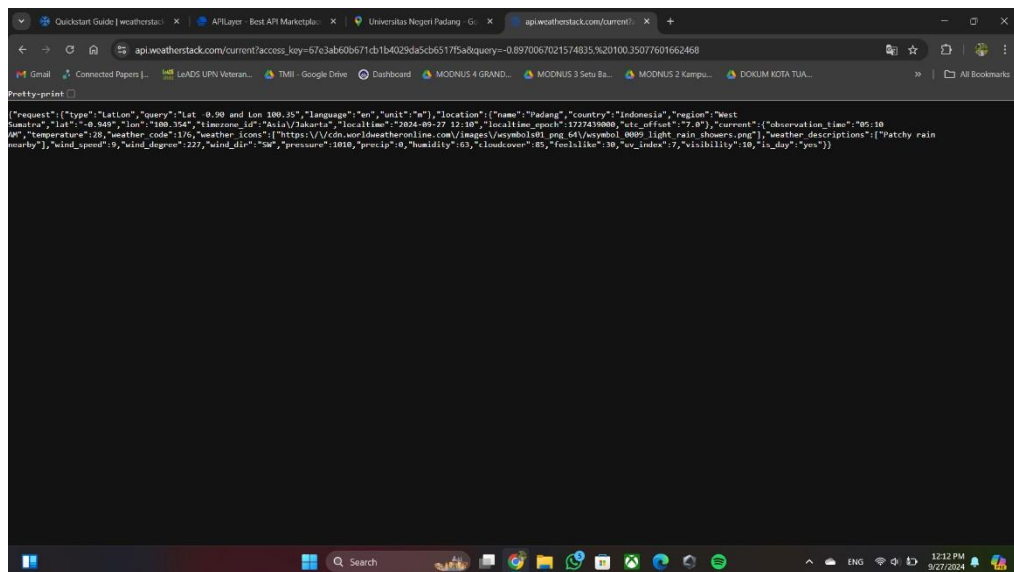
4. Bukalah google map dan carilah **Universitas Negeri Padang**. Klik kana dibagian map dan akan tampil informasi terkait latitude dan longitude. Informasi ini juga terdapat pada url google maps. Latitude dan longitude pada URL dan pada maps bisa saja berbeda tergantung dimana anda memposisikan klik kanan anda pada maps



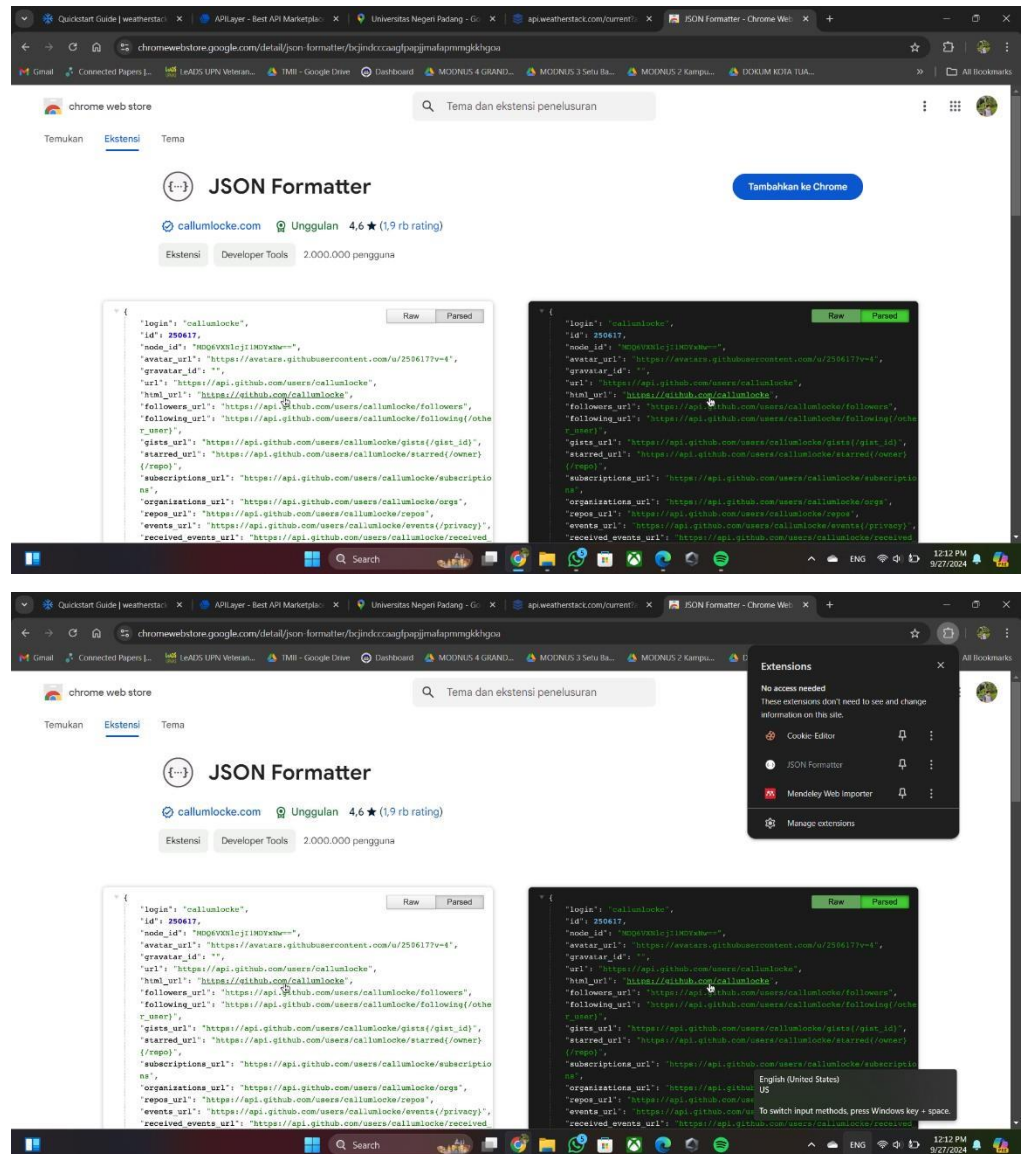
5. Ketikan URL berikut di tab baru browser anda [http://api.weatherstack.com/current?access\\_key=\\*\\*\\*\\*\\*&query=\\*\\*\\*\\*\\*](http://api.weatherstack.com/current?access_key=*****&query=*****), \*\*\*\*\*Gantilah tandan bintang (\*) berwarna merah dengan kode access API key anda, warna hijau dengan latitude dan warna coklat dengan longitude. Pastikan tidak ada angka atau karakter yang tertinggal termasuk tanda minus (-) jika ada. Tekan enter dan akan tampil seperti gambar dibawah ini

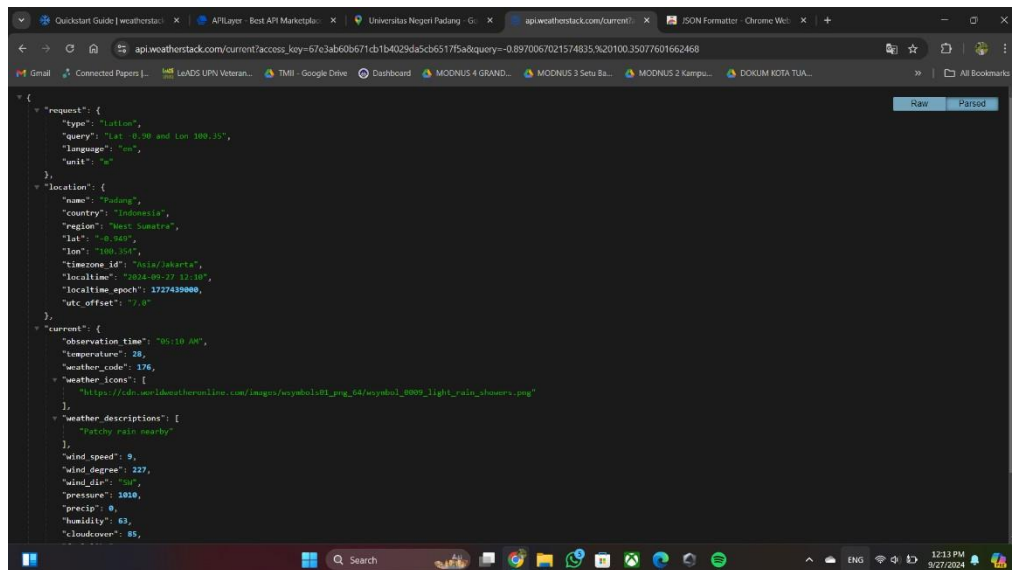


6. Tampilan diatas menggunakan format JSON. Agar tampilan menjadi lebih rapi, bukalah <https://chrome.google.com/webstore/category/extensions> kemudian cari JSON Formatters dan install ekstensi tersebut



7. Tampilan file JSON dari API Weatherstack akan menjadi seperti ini



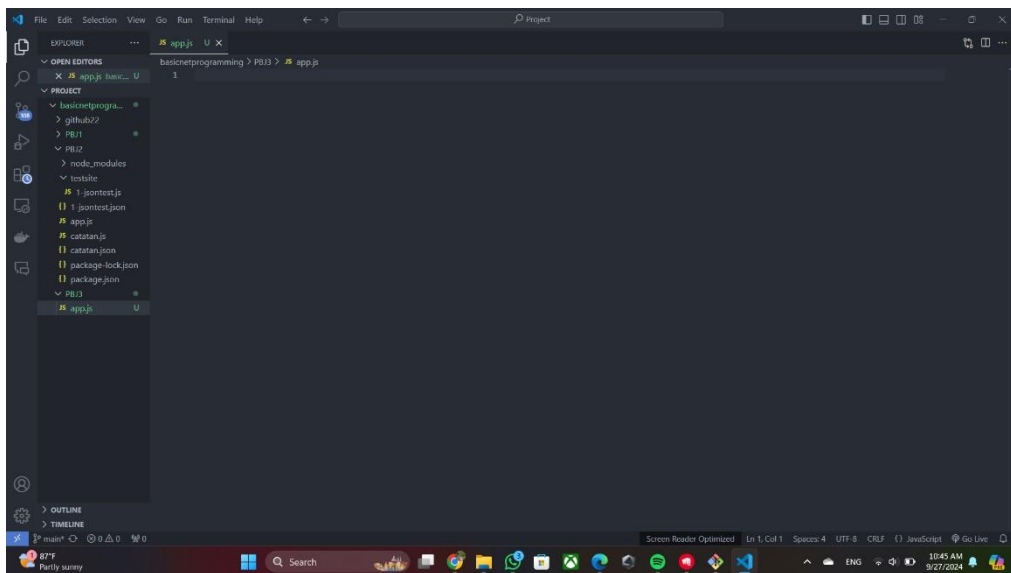
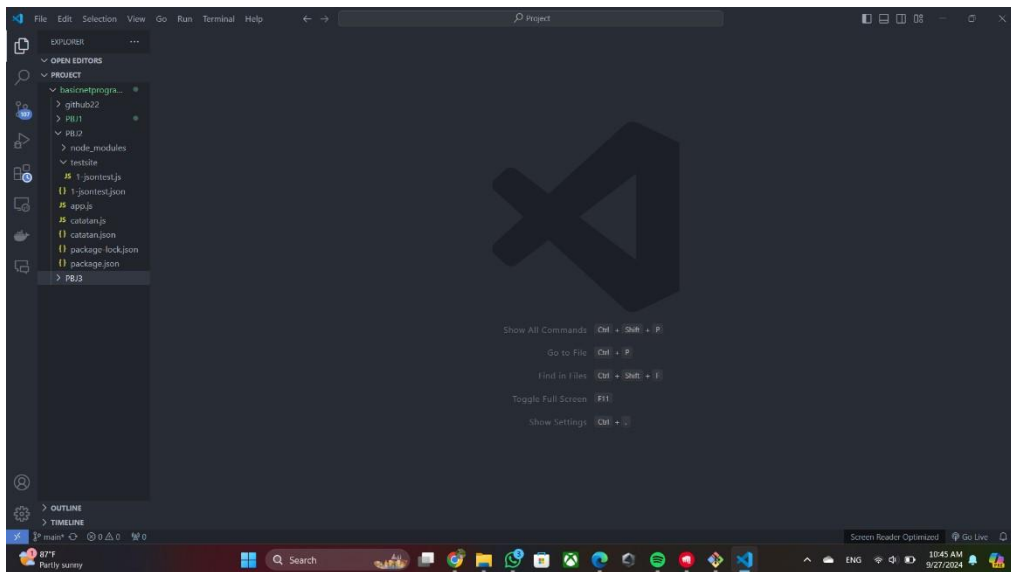


8. Jika anda menggunakan browser **Firefox**, maka **tidak perlu** menginstall extension atau add-on untuk tampilan seperti ini.

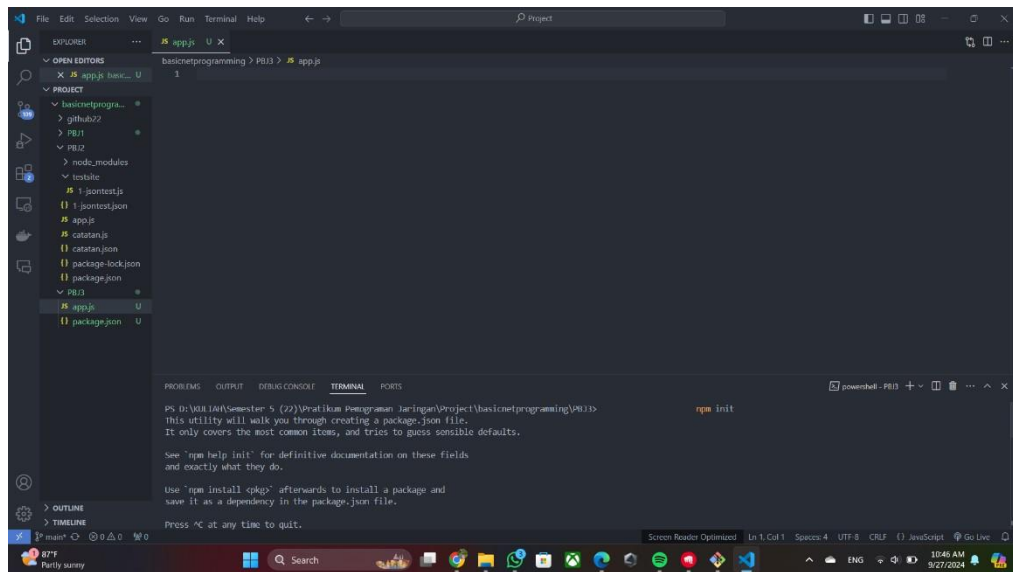
9. Anda dapat menemukan informasi lengkap terkait cuaca di Universitas Negeri Padang melalui data yang ditampilkan seperti informasi lokasi, cuaca terkini, kapan cuaca di observasi, temperature, deskripsi dan lainnya. **PENTING!** Anda akan mengakses data ini melalui program

#### b. HTTP Request

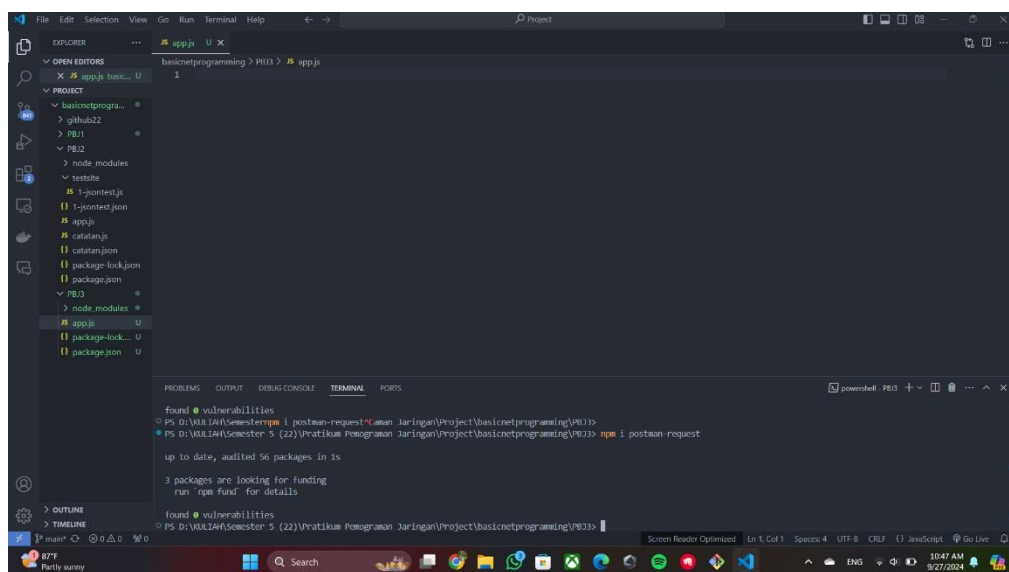
1. Buatlah folder project baru pada visual studio code dengan nama **aplikasiCuaca**, lalu buatlah file javascript baru dalam folder tersebut dengan nama **app.js**.



2. Akses terminal melalui visual studio code, pastikan anda berada pada direktori folder yang baru anda buat. Lalu install **npm** dengan perintah **npm init** (sama seperti pada modul 2)



3. Setelah itu, silakan install module **postman-request** <https://www.npmjs.com/package/postman-request> dengan perintah **npm i postman-request**.



4. Ketikkanlah kode berikut di file app.js. Kemudian Gantilah tanda bintang (\*) seperti yang anda lakukan pada Langkah A no.5. Lalu jalankan program tersebut melalui terminal dan pahami apa yang ditampilkan.

```
1 const request = require('postman-request')
2 const url = 'http://api.weatherstack.com/current?access key=6703ab68671cb1b4029d5cb6517f5a&query=0.8886443736412998, 100.42317399108163'
3 request({ url: url }, (error, response) => {
4   console.log(response)
5   // console.log(response.body)
6   const data = JSON.parse(response.body)
7   console.log(data)
8   // console.log(data.current)
9   // console.log(data.current.temperature)
10 })
```

- Jadikan kode pada baris ke-4 menjadi komentar dan kemudian **uncomment** baris ke-5 dan ke-6. Jalankan dan pahami apa yang ditampilkan

```
1 const request = require('postman-request')
2 const url = 'http://api.weatherstack.com/current?access key=6703ab68671cb1b4029d5cb6517f5a&query=0.8886443736412998, 100.42317399108163'
3 request({ url: url }, (error, response) => {
4   // console.log(response)
5   const data = JSON.parse(response.body)
6   console.log(data)
7   // console.log(data.current)
8   // console.log(data.current.temperature)
9 })
```

```
PS D:\MULIA\Semester 5 (22)\Pratikum Pemrograman Jaringan\Project\basiconetprogramming\PB3> node app.js
{
  request: {
    type: 'latlon',
    query: 'lat=0.89 and lon=100.42',
    language: 'en',
    unit: 'm'
  },
  location: {
    name: 'Padang',
```

- Jadikan kode pada baris ke-6 sebagai komentar, lalu **Uncomment** lagi kode pada baris ke 7 dan pahami apa yang ditampilkan



```
const request = require('https');
const url = 'https://api.weatherstack.com/current?access_key=67e3ab6b671c1b1b4829duScb6517f5a&query=0.8886443736412998, 100.42317399108163';
request({ url: url }, (error, response) => {
  //console.log(response)
  const data = JSON.parse(response.body)
  //console.log(data)
  console.log(data.current)
  // console.log(data.current.temperature)
})
```

```
PS D:\VULIAH\Semester 5 (22)Vratrikum Pomograman Jaringan\Project\basinetprogramming\VB3> node app.js
{
  observation_time: '03:50 AM',
  temperature: 27,
  weather_code: 113,
  weather_icons: [
    'https://cdn.weatheronline.com/images/xyxh01s01_png_64/xyxh01_0801_sunny.png'
  ],
  weather_descriptions: [ 'Sunny' ],
}
```

- Jadikan kode pada baris ke-7 sebagai komentar, lalu **Uncomment** lagi kode pada baris ke-8 dan pahami apa yang ditampilkan.

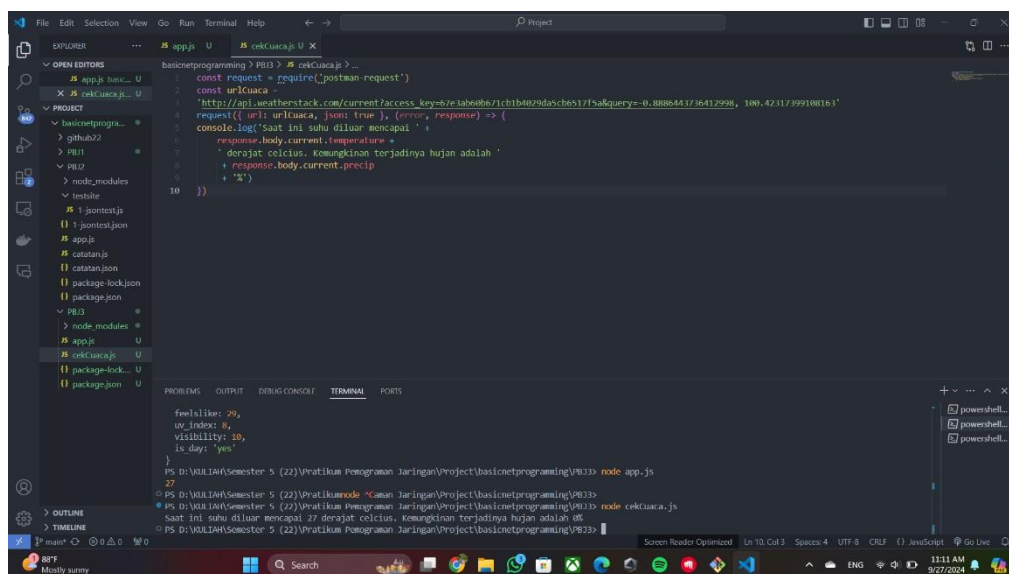
```
const request = require('http');
const url = 'http://api.weatherstack.com/current?access_key=67e3ab6b671c1b1b4829duScb6517f5a&query=0.8886443736412998, 100.42317399108163';
request({ url: url }, (error, response) => {
  //console.log(response)
  const data = JSON.parse(response.body)
  //console.log(data)
  console.log(data.current)
})
```

```
PS D:\VULIAH\Semester 5 (22)Vratrikum Pomograman Jaringan\Project\basinetprogramming\VB3> node app.js
{
  cloudcover: 22,
  feellike: 29,
  uv_index: 8,
  visibility: 10,
  is_day: 'yes',
}
```

- Perlu diperhatikan pada const url bahwa **base url** untuk akses API menggunakan HTTP bukan HTTPS. Untuk menggunakan versi HTTPS maka dikenakan biaya. Pastikan ketika anda mengetikannya di browser juga menggunakan HTTP.



9. Perhatikan struktur file API dalam format JSON pada saat anda mengaksesnya melalui browser
10. Terlihat bahwa data tersebut memiliki hirarki yaitu request, location, current dimana didalamnya memiliki sub-hirarki atau data lagi. Kode pada no.4 baris ke-8 menunjukkan bahwa anda telah mengakses data -> current -> temperature. Hal ini berarti anda juga dapat mengakses data lainnya. Sementara itu **response.body** menunjukkan bahwa anda mengakses keseluruhan data yang ditampilkan.
11. Buatlah file baru dengan nama **cekCuaca.js** dan ketikkan kode berikut ini. Kemudian jangan lupa mengganti tanda bintang dengan data anda masing-masing. Jalankan program tersebut dan pahami apa yang ditampilkan.



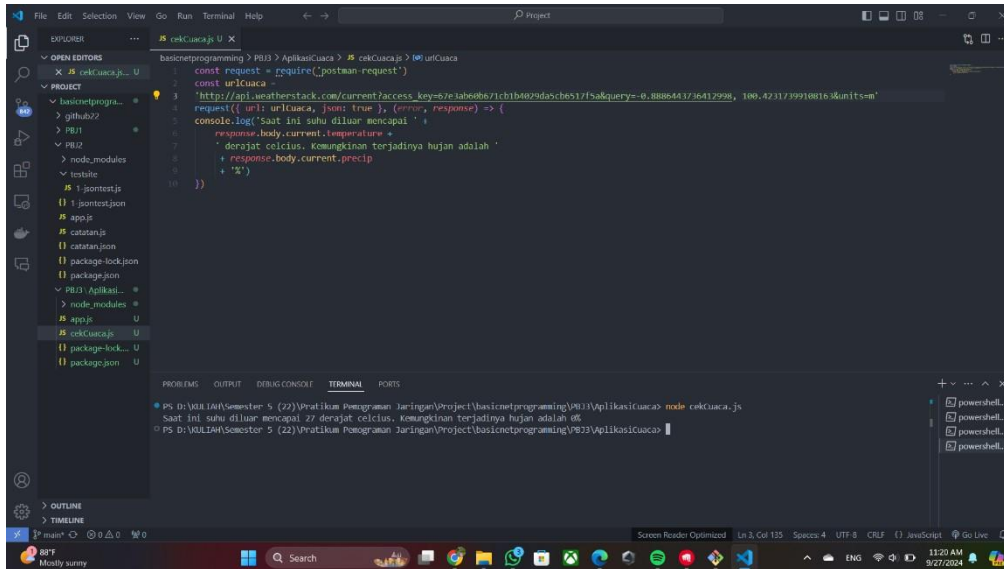
```
const request = require('postman-request')
const urlCuaca = 'http://api.weatherstack.com/current?access_key=7e1a6e0b671c1b402d4d5e517f5akquery=-0.8886441736412998, 106.42317399108151'
request({ url: urlCuaca, json: true }, (error, response) => {
  console.log('Saat ini suhu diluar mencapai ' +
    response.body.current.temperature +
    ' derajat celcius. Kemungkinan terjadinya hujan adalah ' +
    response.body.current.precip +
    '%')
})
```

```
feelLike: 20,
uv_index: 8,
visibility: 10,
is_day: 'yes'
}
```

```
PS D:\VULIAH\Semester 5 (22)\Pratikum Pemrograman Jaringan\Project\basinetprogramming\VB33> node app.js
27
PS D:\VULIAH\Semester 5 (22)\Pratikum Pemrograman Jaringan\Project\basinetprogramming\VB33>
PS D:\VULIAH\Semester 5 (22)\Pratikum Pemrograman Jaringan\Project\basinetprogramming\VB33> node cekCuaca.js
Saat ini suhu diluar mencapai 27 derajat celcius. Kemungkinan terjadinya hujan adalah 0%
PS D:\VULIAH\Semester 5 (22)\Pratikum Pemrograman Jaringan\Project\basinetprogramming\VB33>
```

12. Bukalah dokumentasi Weatherstack melalui link berikut ini <https://weatherstack.com/documentation> dan perhatikan bahwa Weatherstack memiliki beberapa opsi. Silakan klik **unit parameter**. Anda dapat melihat bahwa akan muncul units yang terkait dengan beberapa parameter suhu Celcius, Fahrenheit dan lainnya. Anda dapat menambahkan unit parameter ini keakhir pemanggilan API anda

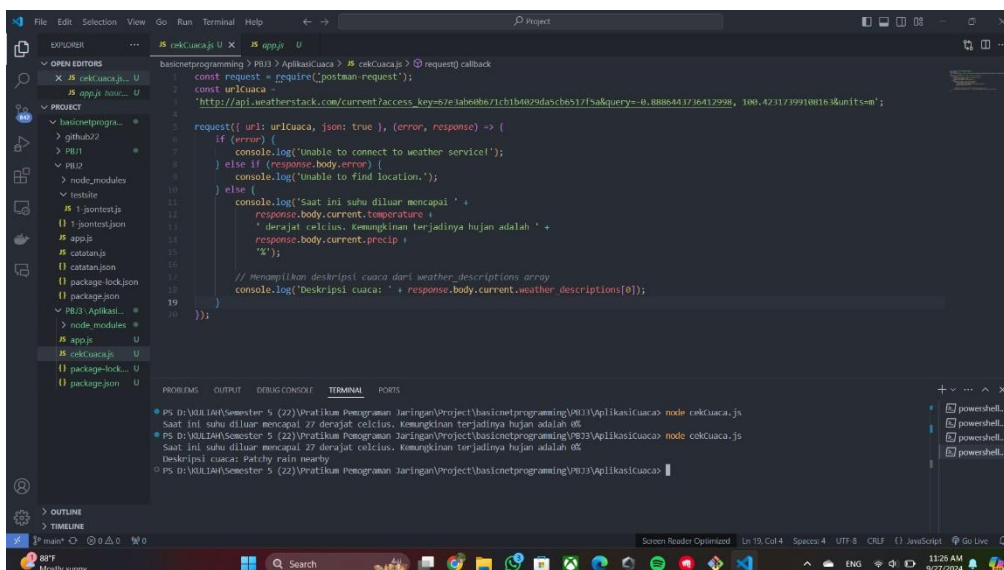
seperti berikut. Selanjutnya gantilah huruf **m** dengan parameter lainnya yang tersedia pada dokumentasi weatherstack. Silakan jelajahi dokumentasi tersebut anda akan menemukan juga parameter language (bahasa) dan lainnya.



```
const request = require('postman-request');
const urlCuaca = 'http://api.weatherstack.com/current?access_key=67e1ab0b671cb1b4029d45cb517f5a&query=-0.8886441736412998,100.4231739910816&units=m';
request({url: urlCuaca, json: true }, (error, response) => {
  console.log('Saat ini suhu diluar mencapai ' +
    response.body.current.temperature +
    ' derajat celsius. Kemungkinan terjadinya hujan adalah ' +
    response.body.current.precip +
    '%');
});
```

### c. Latihan 1 – API Access WeatherStack

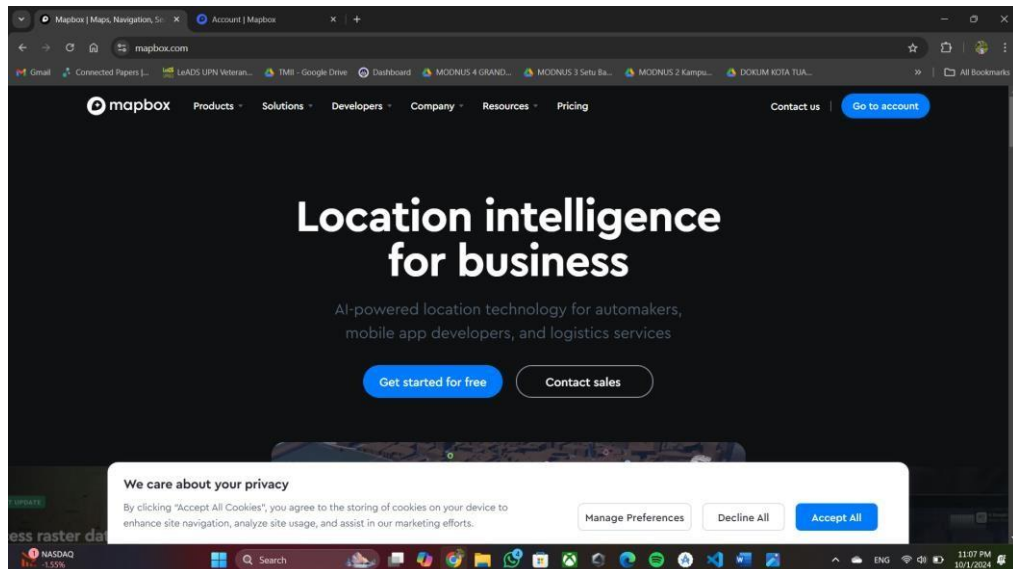
1. Pada kode yang ada pada Langkah b no.13, tambahkan kode agar Anda bisa mengakses `weather_descriptions` (deskripsi cuaca).Keemudian tampilkan teks berikut sesuai dengan keinginan anda. Perlu diperhatikan bahwa data `weather_descriptions` menggunakan format array sehingga anda perlu menambah `[]` di akhir pemanggilan data. Misal: seandainya `temperature` dalam format array maka akan menjadi `response.body.current.temperature[0]`



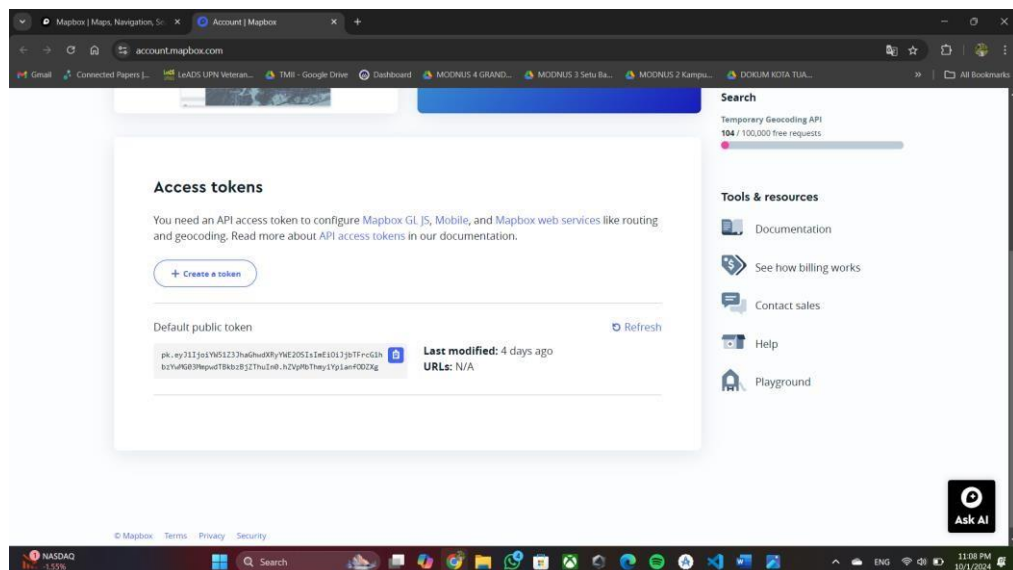
```
const request = require('postman-request');
const urlCuaca = 'http://api.weatherstack.com/current?access_key=67e1ab0b671cb1b4029d45cb517f5a&query=-0.8886441736412998,100.4231739910816&units=m';
request({url: urlCuaca, json: true }, (error, response) => {
  if (error) {
    console.log('Unable to connect to weather service!');
  } else if (response.body.error) {
    console.log('Unable to find location.');
```

d. Latihan 2 – API Mapbox

1. Silakan buat akun <https://www.mapbox.com/>. Setelah itu login dan cek menu tokens.



2. Bukalah dokumentasi API mapbox melalui link berikut ini <https://docs.mapbox.com/api/search/geocoding/> Carilah **Forward Geocoding** (menu dibagian kanan)

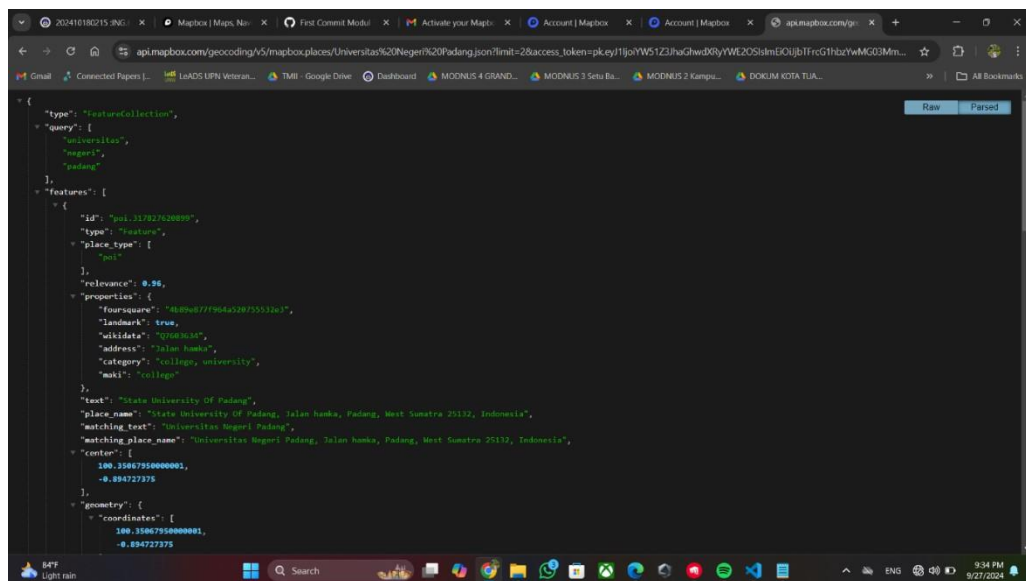


3. Pahami dokumentasinya dan lihat bagaimana cara pemanggilan API nya. Perhatikan data yang ada pada tabel dan juga contoh pada **example request** .

4. Berikut adalah salah satu example request yang mereka sediakan <https://api.mapbox.com/geocoding/v5/mapbox.places/Washington.js>

o  
n?limit=2&access\_token=**pk.eyJ1IjoicHJvc2thOTkiLCJhIjoieY2xsbd**  
**k**  
**YWppMDBlYzNyYXcwNmM3p6OSJ9.9S7iRjDXb1JqUbpKK94**  
**9**

**ew**. Gantilah teks warna **hijau** dengan kota atau tempat yang ingin anda cari, misal: **Padang** atau **Universitas Negeri Padang**. Lalu, ganti teks warna **merah** dengan token anda. Cobalah akses di browser dan lihat data yang ditampilkan .



5. Lalu cobalah lakukan **request data** dengan menambahkan kode berikut ke file app.js, Selanjutnya gantilah teks warna **hijau** dengan lokasi atau tempat yang ingin anda cari dan teks warna **merah** dengan token API Mapbox Anda .Limit=1 digunakan untuk membatasi pencarian data agar hanya menampilkan 1 pencarian. Cobalah ganti angka 1 menjadi 2 atau 3, lalu gantilah array [0] pada features dengan angka 1 atau 2. Lakukan untuk kedua variable latitude dan longitude.

Kemudian perhatikanlah bahwa tampilan latitude dan longitude pada terminal telah berganti

```

1  const axios = require('axios');
2
3  // URL API OpenWeatherMap
4  const baseUrl = 'https://api.openweathermap.org/data/2.5/';
5
6  // Fungsi untuk mendapatkan lokasi saat ini
7  async function getLocation() {
8    const geocodeURL = 'https://api.mapbox.com/geocoding/v5/mapbox.places/universitas Negeri Padang.json?access_token=pk-eyJ11J0lY6S123hGhUdx8yYkE2051';
9
10   request({url: geocodeURL, json: true }, (error, response) => {
11     if (error) {
12       return console.log('unable to connect to geolocation service!');
13     } else if (response.body.features.length === 0) {
14       return console.log('no location found. Try another search.');
```

e. Latihan 3- Memanngildat API

1. Lanjutkanlah program pada **bagian d**. Lakukanlah pemanggilan data sebagaimana yang telah anda pelajari pada saat mengakses API weatherstack untuk menampilkan query, place\_name dan place\_type.

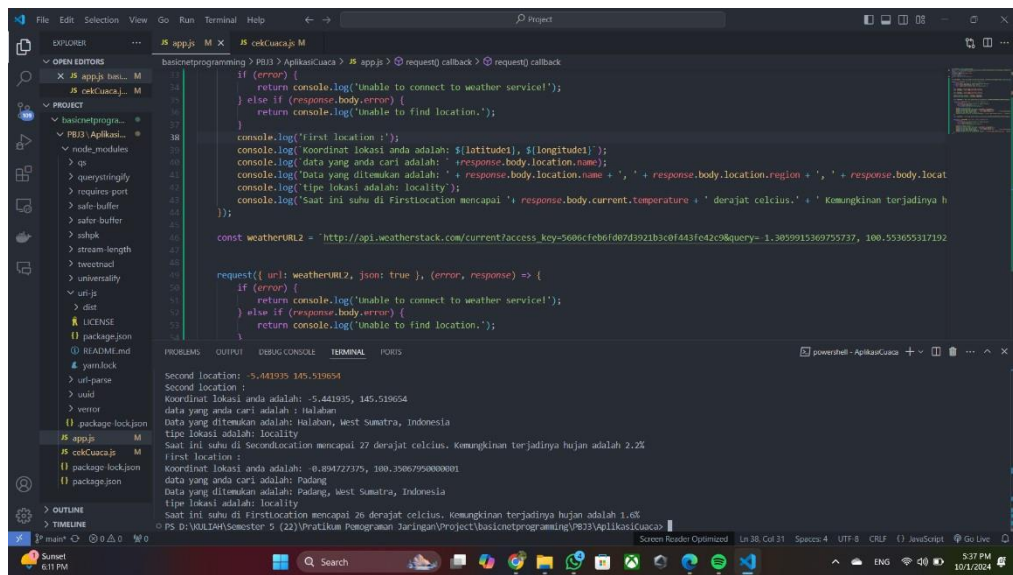
Contoh tampilannya adalah sebagai berikut

The image shows a VS Code editor window with a file named 'app.js' open. The code is a JavaScript script that uses the 'request' library to fetch weather data from 'api.weatherstack.com'. It logs the first and second locations and their weather conditions to the console. The terminal output shows the results of running 'node app.js', displaying the first and second locations and their weather conditions. The interface includes a sidebar with file explorer, search, and source control, and a bottom status bar with various icons and information.

2. Lakukanlah pemanggilan data dari API weatherstack dan mapbox API hingga menampilkan output seperti berikut. **Penting!** Data lokasi bisa berbeda tergantung lokasi yang anda cari. Pastikan data lokasi yang



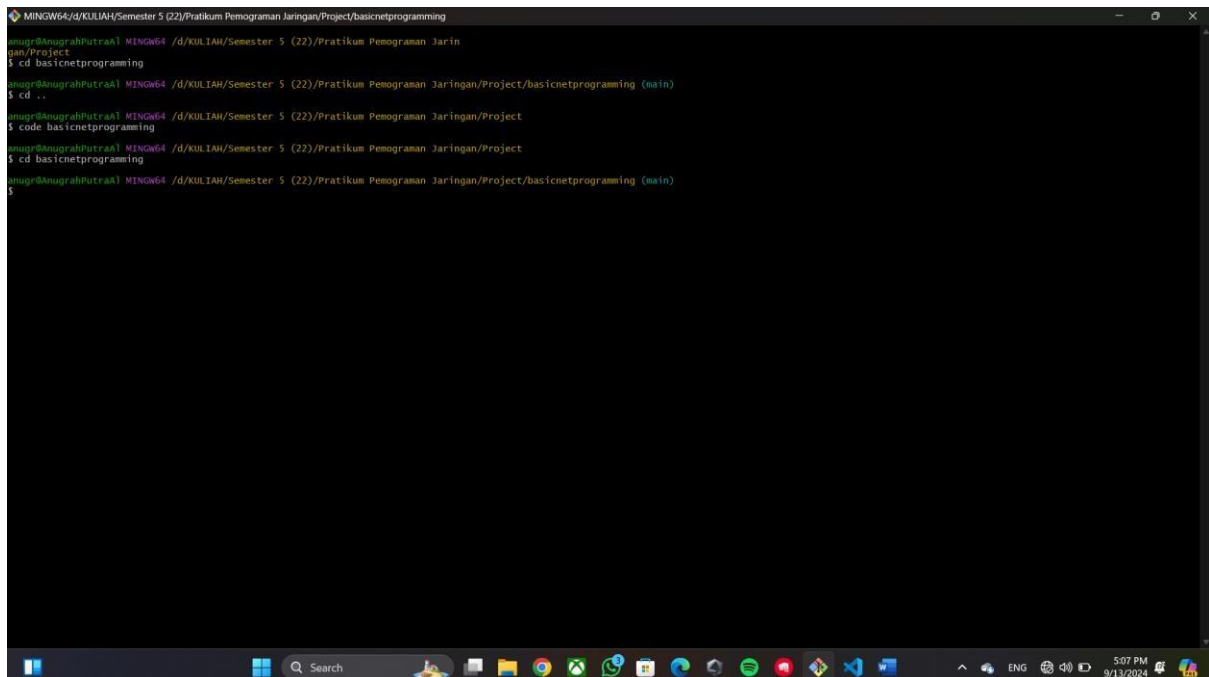
anda cari dengan API mapbox sesuai dengan koordinat yang anda berikan pada API weatherstack



```
basicnetprogramming > PB3 > AplikasiCuaca > JS app.js > request() callback > request() callback
14 | if (error) {
15 |   return console.log('unable to connect to weather service!');
16 | } else if (response.body.error) {
17 |   return console.log('unable to find location.');
```

```
Second location: -5.441935, 145.516054
Second location :
Koordinat lokasi anda adalah: -5.441935, 145.516054
data yang anda cari adalah: Halaban
Data yang ditemukan adalah: Halaban, West Sumatra, Indonesia
tipe lokasi adalah: Locality
Saat ini suhu di FirstLocation mencapai 27 derajat celcius. Kemungkinan terjadinya hujan adalah 2.2%
```

13. Setelah seluruhnya selesai, kita kembali lagi ke *Git* lalu kita menuliskan syntax lagi dengan *cd basicnetprogramming*.



```
MINGW64/d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jaringan/Project/basicnetprogramming
msgr@AnugrahPutraA1 MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jari
$ cd basicnetprogramming
msgr@AnugrahPutraA1 MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jari
$ cd ..
msgr@AnugrahPutraA1 MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jari
$ code basicnetprogramming
msgr@AnugrahPutraA1 MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jari
$ cd basicnetprogramming
msgr@AnugrahPutraA1 MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemograman Jari
$
```

14. Langkah berikutnya, kita menambahkan file yang kita buat tadi menulis code atau syntax dengan *git add PB3*.



```
MINGW64/d/KULIAH/Semester 5 (22)/Pratikum Pemrograman Jaringan/Praktek/basicnetprogramming
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/parse.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/regex.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/rng-browser.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/rng.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/sha1-browser.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/sha1.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/stringify.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uid.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidParse.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidStringify.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidValidate.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidVersion.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidv1.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidv2.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidv3.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidv4.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidv5.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidv-bin.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/v1.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/v2.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/v3.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/v35.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/v4.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/v5.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/validate.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/version.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/package.json
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/wrapper.mjs
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/.npmignore
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/CHANGELOG.md
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/CONTRIBUTING.md
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/LICENSE
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/README.md
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/lib/verror.js
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/package.json
create mode 100644 PB33/AplikasiCuaca/package-lock.json
create mode 100644 PB33/AplikasiCuaca/package.json

ming@AnugrahPutra:~$ MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemrograman Jaringan/Praktek/basicnetprogramming (main)
$ git push
Enumerating objects: 844, done.
Counting objects: 100% (844/844), done.
Delta compression using up to 16 threads
Compressing objects: 100% (811/811), done.
Writing objects: 100% (843/843), 1.59 MiB | 682.00 KiB/s, done.
Total 843 (delta 102), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (102/102), done.
to https://github.com/anugrahhhh/basicnetprogramming.git
8930c06..c29013b main -> main

ming@AnugrahPutra:~$ MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemrograman Jaringan/Praktek/basicnetprogramming (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

ming@AnugrahPutra:~$ MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemrograman Jaringan/Praktek/basicnetprogramming (main)
$
```

17. Lalu setelah melihat statusnya, selanjutnya menuliskan syntax terbaru yaitu *git push* digunakan untuk mengirim (mem-push) commit yang telah dibuat secara lokal ke repository jarak jauh (remote repository), seperti GitHub, GitLab, atau Bitbucket. Dengan git push, perubahan yang telah disimpan di repository lokal akan disinkronkan dan ditambahkan ke repository remote agar dapat diakses oleh orang lain atau perangkat lain.

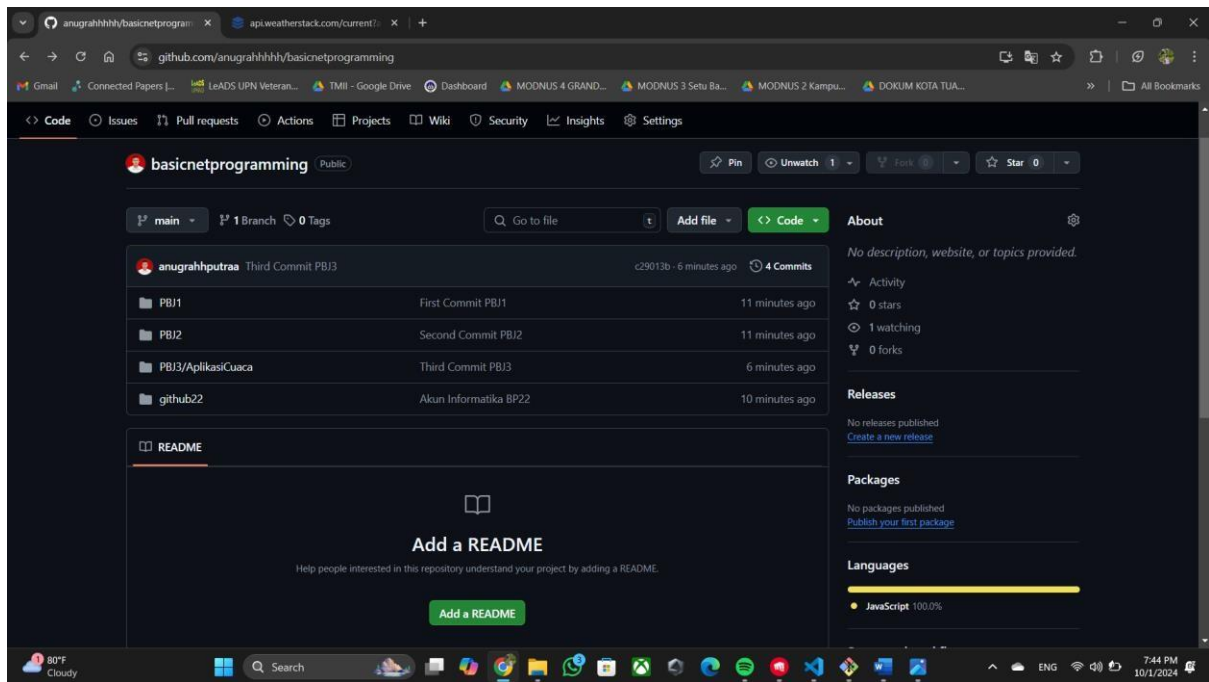
```
MINGW64/d/KULIAH/Semester 5 (22)/Pratikum Pemrograman Jaringan/Praktek/basicnetprogramming
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/esm-node/v5.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/esm-node/validate.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/esm-node/version.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/index.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/md5-browser.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/md5.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/m1.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/parse.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/regex.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/rng-browser.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/rng.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/sha1-browser.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/sha1.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/stringify.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uid.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidParse.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidStringify.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidValidate.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidVersion.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidv1.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidv2.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidv3.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidv4.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidv5.min.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/uid/uidv-bin.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/v1.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/v2.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/v3.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/v35.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/v4.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/v5.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/validate.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/dist/version.js
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/package.json
create mode 100644 PB33/AplikasiCuaca/node_modules/uid/wrapper.mjs
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/.npmignore
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/CHANGELOG.md
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/CONTRIBUTING.md
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/LICENSE
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/README.md
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/lib/verror.js
create mode 100644 PB33/AplikasiCuaca/node_modules/verror/package.json
create mode 100644 PB33/AplikasiCuaca/package-lock.json
create mode 100644 PB33/AplikasiCuaca/package.json

ming@AnugrahPutra:~$ MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemrograman Jaringan/Praktek/basicnetprogramming (main)
$ git push
Enumerating objects: 844, done.
Counting objects: 100% (844/844), done.
Delta compression using up to 16 threads
Compressing objects: 100% (811/811), done.
Writing objects: 100% (843/843), 1.59 MiB | 682.00 KiB/s, done.
Total 843 (delta 102), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (102/102), done.
to https://github.com/anugrahhhh/basicnetprogramming.git
8930c06..c29013b main -> main

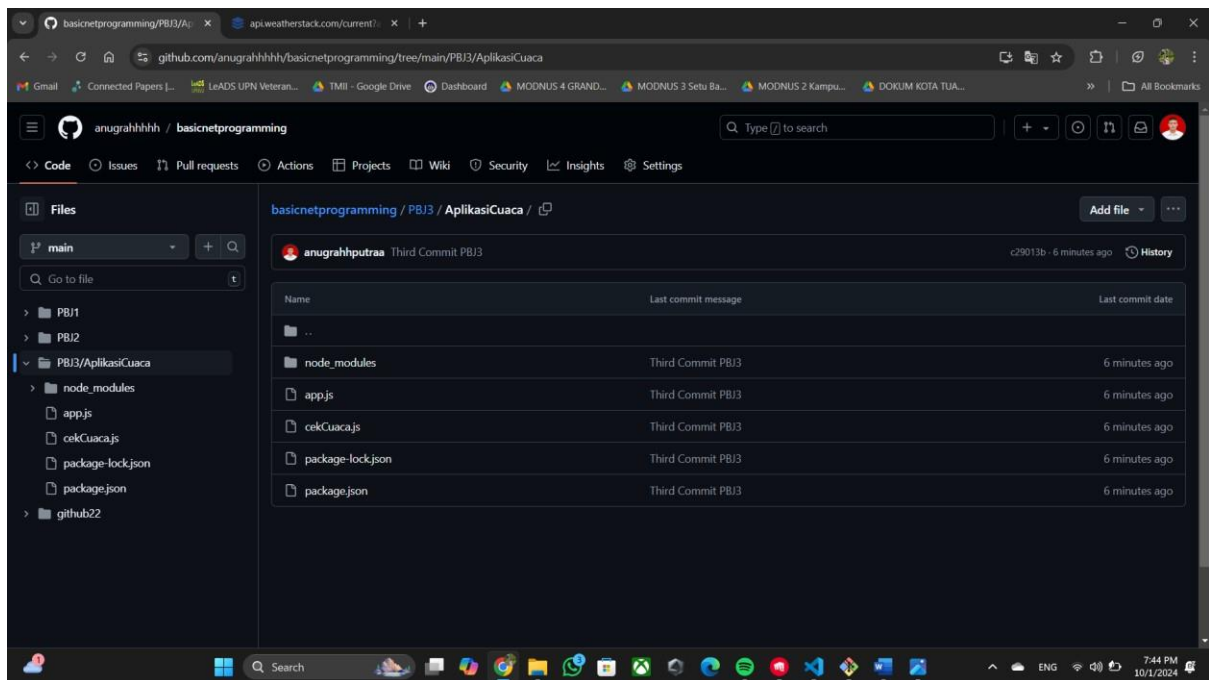
ming@AnugrahPutra:~$ MINGW64 /d/KULIAH/Semester 5 (22)/Pratikum Pemrograman Jaringan/Praktek/basicnetprogramming (main)
$
```

18. Setelah berhasil kita lalu check, ke akun github.





19. Setelah kita membuka folder nya di dalam github tersebut,kita bisa melihat folder yang telah kita buat.



## **BAB III**

### **PENUTUP**

#### **3.1. Kesimpulan**

Berdasarkan pelaksanaan praktikum ini, dapat disimpulkan bahwa pemahaman tentang HTTP Request, API, dan Git merupakan komponen penting dalam pengembangan aplikasi berbasis jaringan. HTTP Request berperan sebagai mekanisme pengiriman permintaan data dari klien ke server, sedangkan API menyediakan antarmuka yang memungkinkan aplikasi untuk mengakses data dari layanan pihak ketiga, seperti pada kasus API Weatherstack yang digunakan untuk mengambil data cuaca secara real-time.

Implementasi API dalam pembuatan aplikasi cuaca membuktikan bahwa API dapat memudahkan pengembang dalam memperoleh dan mengintegrasikan data eksternal ke dalam aplikasi, tanpa perlu memahami detail teknis dari layanan penyedia data tersebut. Penggunaan API juga meningkatkan efisiensi pengembangan aplikasi dengan memanfaatkan data yang selalu diperbarui secara otomatis oleh pihak ketiga.

Selain itu, penggunaan Git dalam pengelolaan proyek sangat membantu dalam memantau dan mengatur perubahan pada kode, terutama dalam proyek kolaboratif. Sistem kontrol versi ini memungkinkan pengembang untuk bekerja lebih terstruktur dan memastikan setiap perubahan pada proyek dapat dilacak dengan baik.

Secara keseluruhan, praktikum ini memberikan wawasan mendalam mengenai pentingnya pemanfaatan teknologi jaringan dalam pengembangan aplikasi modern, khususnya dalam hal akses data melalui HTTP Request dan API, serta pengelolaan proyek yang baik menggunakan Git. Dengan pemahaman ini, mahasiswa diharapkan dapat menerapkan teknik-teknik yang dipelajari dalam pengembangan aplikasi jaringan di masa depan.

#### **3.2. Daftar Pustaka**

1. Casciaro. (2014, December). Node.js Design Patterns (3rd ed.). Packt Publishing.
2. Pasquali, S., & Faaborg, K. (2017, December 29). Mastering Node.js: Build Robust and Scalable Real-Time Server-Side Web Applications Efficiently.
3. Guides | Node.js. (n.d.). Node.js. <https://nodejs.org/en/docs/guides>
4. npm Docs. (n.d.). Npm Docs. <https://docs.npmjs.com/>
5. Leka, M. (2022, June 8). Exploring the JavaScript Ecosystem: Popular Tools, Frameworks, and Libraries. Medium. <https://mirzaleka.medium.com/exploring-javascript-ecosystem-popular-tools-frameworks-libraries-7901703ec88>