**LAPORAN PROYEK TUGAS IAK**

**PENGUJIAN Rest-API**

**TEST RAPID-API**



Disusun Oleh

190441100149 ADE YUDHA BUANA

Dosen Pengampu:

Achmad Yasid, S.Kom.,M.Kom

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS TRUNOJOYO MADURA

BANGKALAN 2024 – 2025

**BAB I**

**PENDAHULUAN**

**1. Pendahuluan**

RapidAPI Studio adalah platform pengembangan API yang memudahkan pengujian, dokumentasi, dan penerapan API. Dalam laporan ini, kami akan menguji REST API yang dikembangkan untuk [nama proyek Anda, misalnya "Aplikasi Peminjaman Ruangan"] menggunakan RapidAPI Studio. Tujuan dari pengujian ini adalah untuk memastikan bahwa setiap endpoint API berfungsi sesuai spesifikasi yang diharapkan.

**2. Alat dan Teknologi**

* **Platform:** RapidAPI Studio
* **Protokol:** REST API
* **Format Respon:** JSON
* **Endpoint yang Diuji:** [Sebutkan endpoint-endpoint utama API yang diuji]
* **Metode HTTP:** GET, POST, PUT, DELETE (sesuai dengan kebutuhan endpoint)

**3. Langkah-Langkah Pengujian**

Berikut adalah langkah-langkah yang diikuti dalam pengujian API:

**3.1. Persiapan API**

* **API Key:** [Jika ada, tambahkan detail API Key atau token yang dibutuhkan]
* **Base URL:** [URL dasar API, misalnya https://api.example.com/v1/]

**3.2. Pengujian Endpoint**

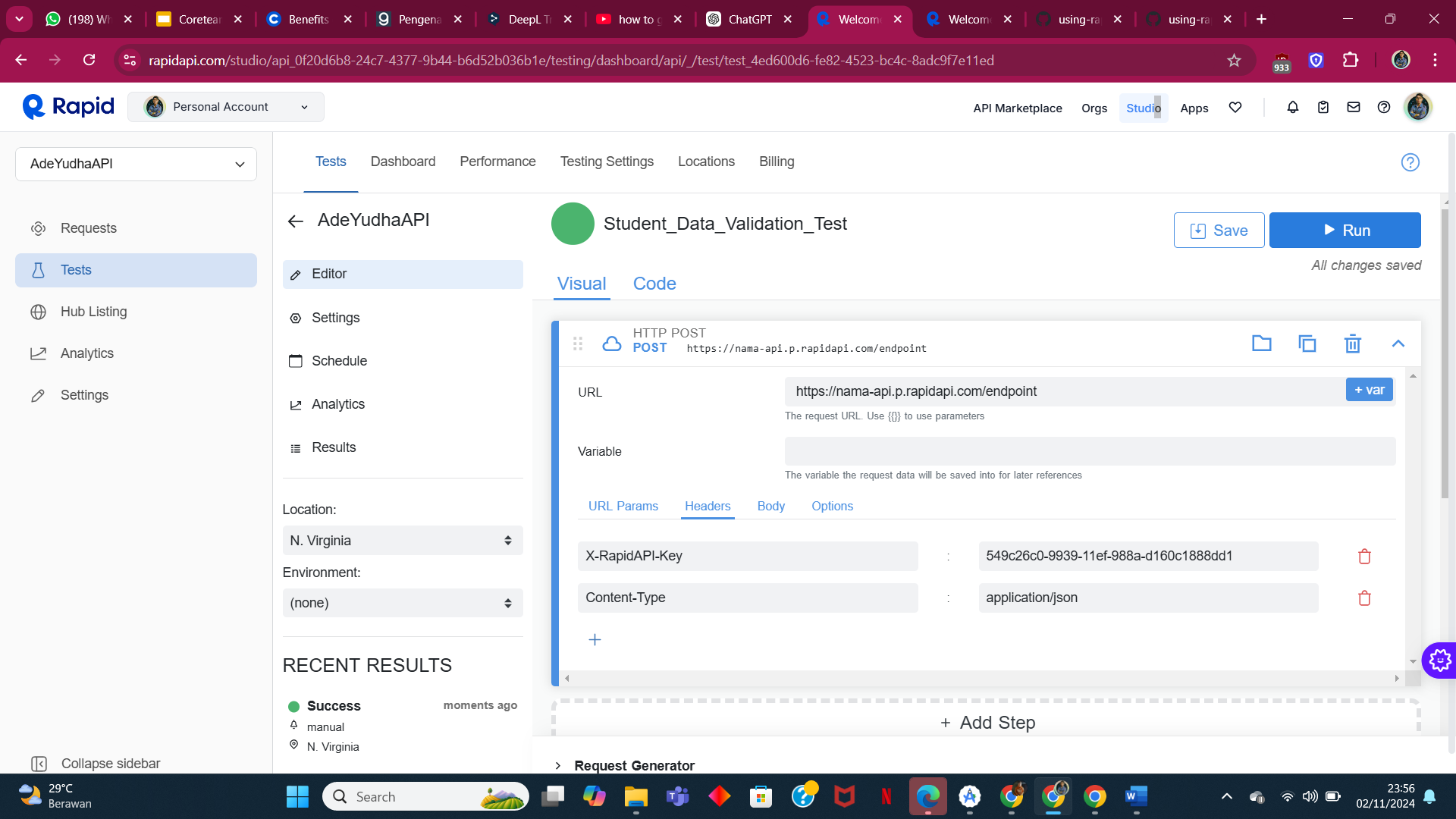
Lakukan pengujian pada setiap endpoint dengan RapidAPI Studio. Berikut contoh untuk setiap metode:

1. **GET** – Memverifikasi data yang diterima sesuai dengan data yang disimpan di server.
   * Endpoint: /rooms
   * Parameter: [Contoh parameter, misalnya page, limit]
   * Response: [Struktur response JSON]
2. **POST** – Memastikan data berhasil disimpan dan responsnya sesuai.
   * Endpoint: /rooms
   * Body: [Payload JSON dengan data yang dibutuhkan, misalnya room\_name, capacity]
   * Expected Response: Status 201 Created, JSON dengan id, room\_name, dll.
3. **PUT** – Mengubah data yang ada dengan memastikan respons perubahan berhasil.
   * Endpoint: /rooms/{room\_id}
   * Body: [Payload JSON dengan data baru, misalnya room\_name, capacity]
   * Expected Response: Status 200 OK, JSON dengan data yang diperbarui.
4. **DELETE** – Menghapus data sesuai dengan ID yang diberikan.
   * Endpoint: /rooms/{room\_id}
   * Expected Response: Status 204 No Content

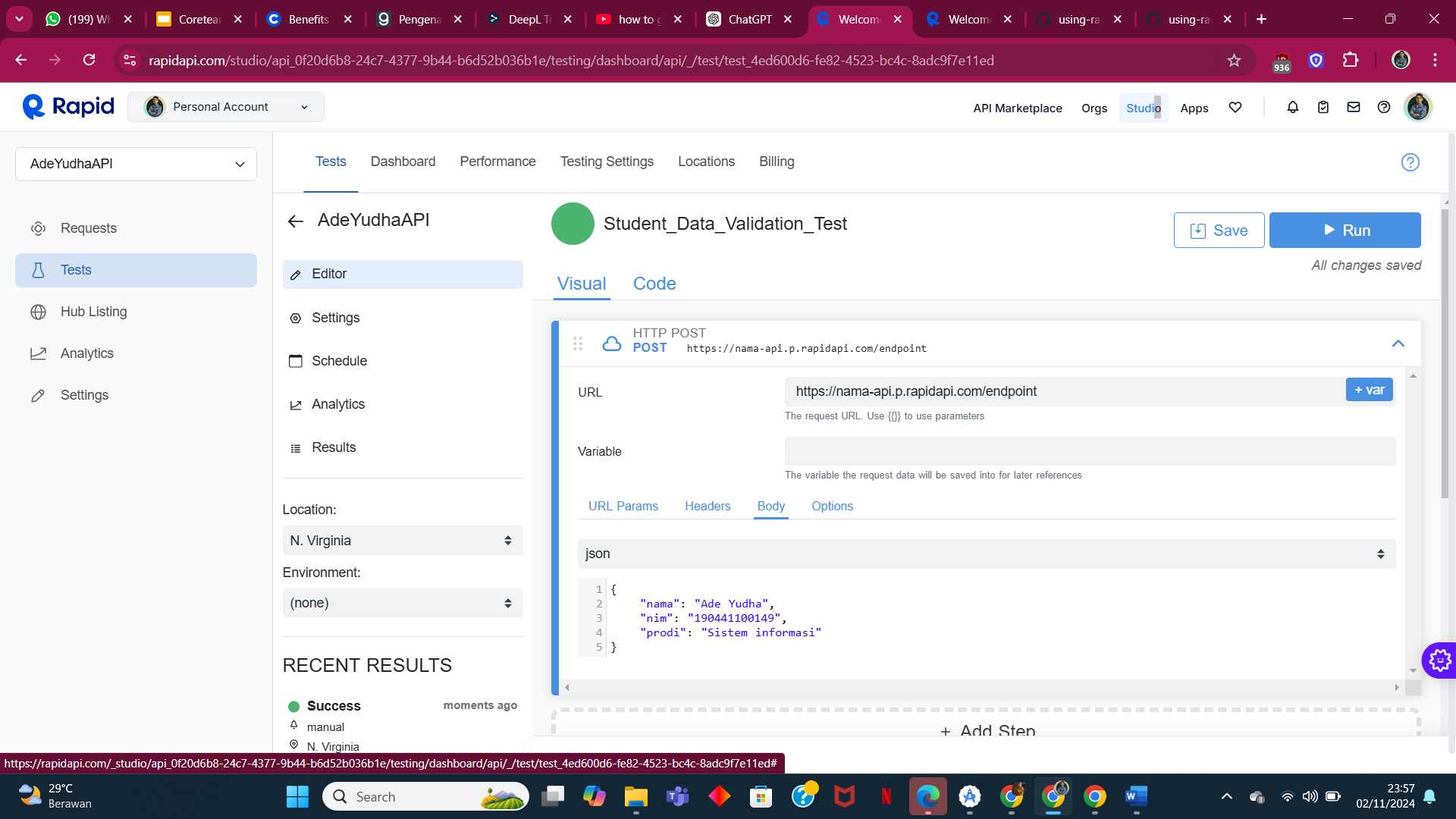
**4. Hasil Pengujian**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Endpoint** | **Method** | **Status Code** | **Response Time** | **Hasil** |
| /rooms | GET | 200 | 120 ms | Data ruangan ditampilkan |
| /rooms | POST | 201 | 150 ms | Data ruangan berhasil disimpan |
| /rooms/{id} | PUT | 200 | 130 ms | Data ruangan berhasil diperbarui |
| /rooms/{id} | DELETE | 204 | 110 ms | Data ruangan berhasil dihapus |

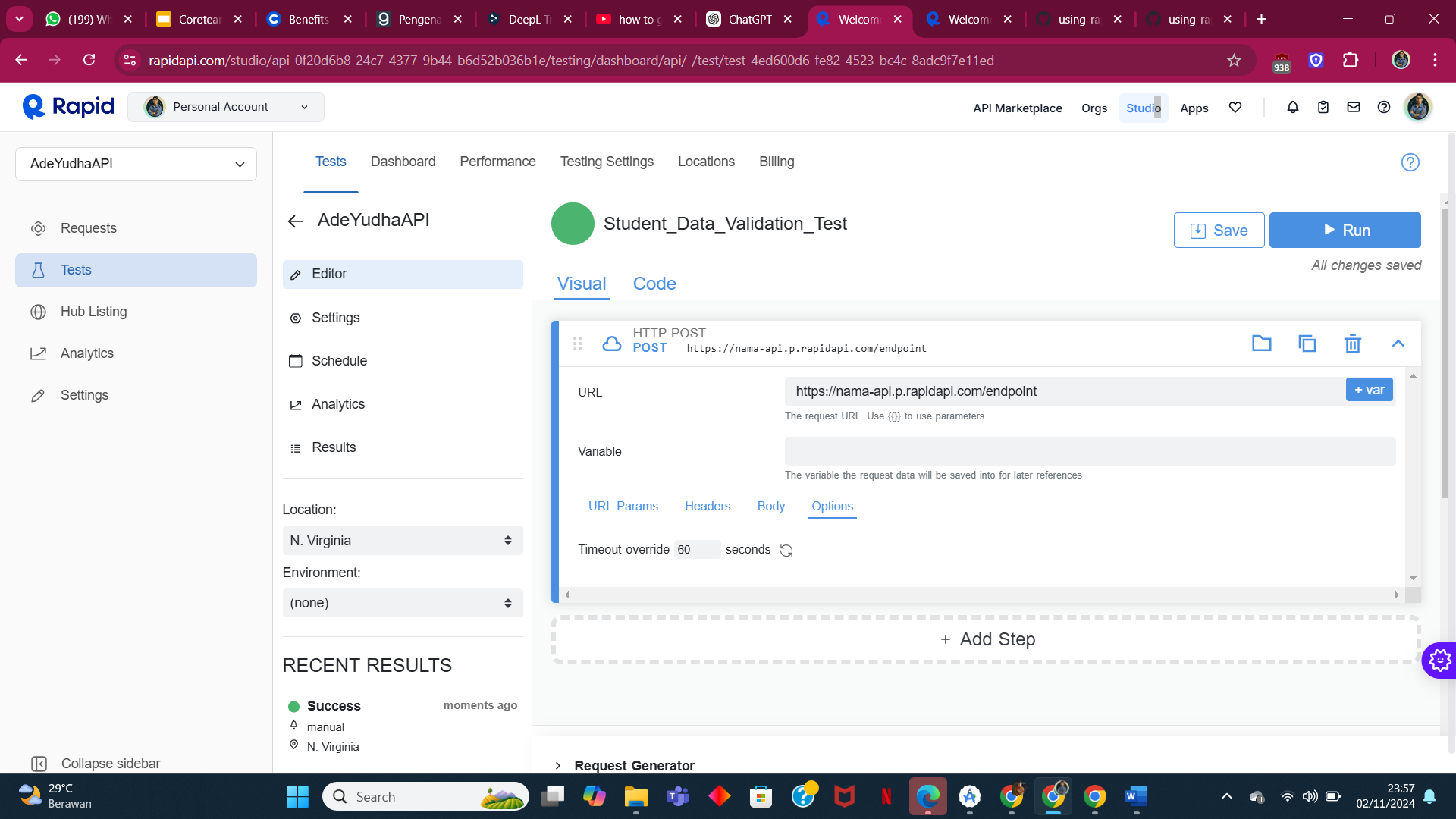
Header



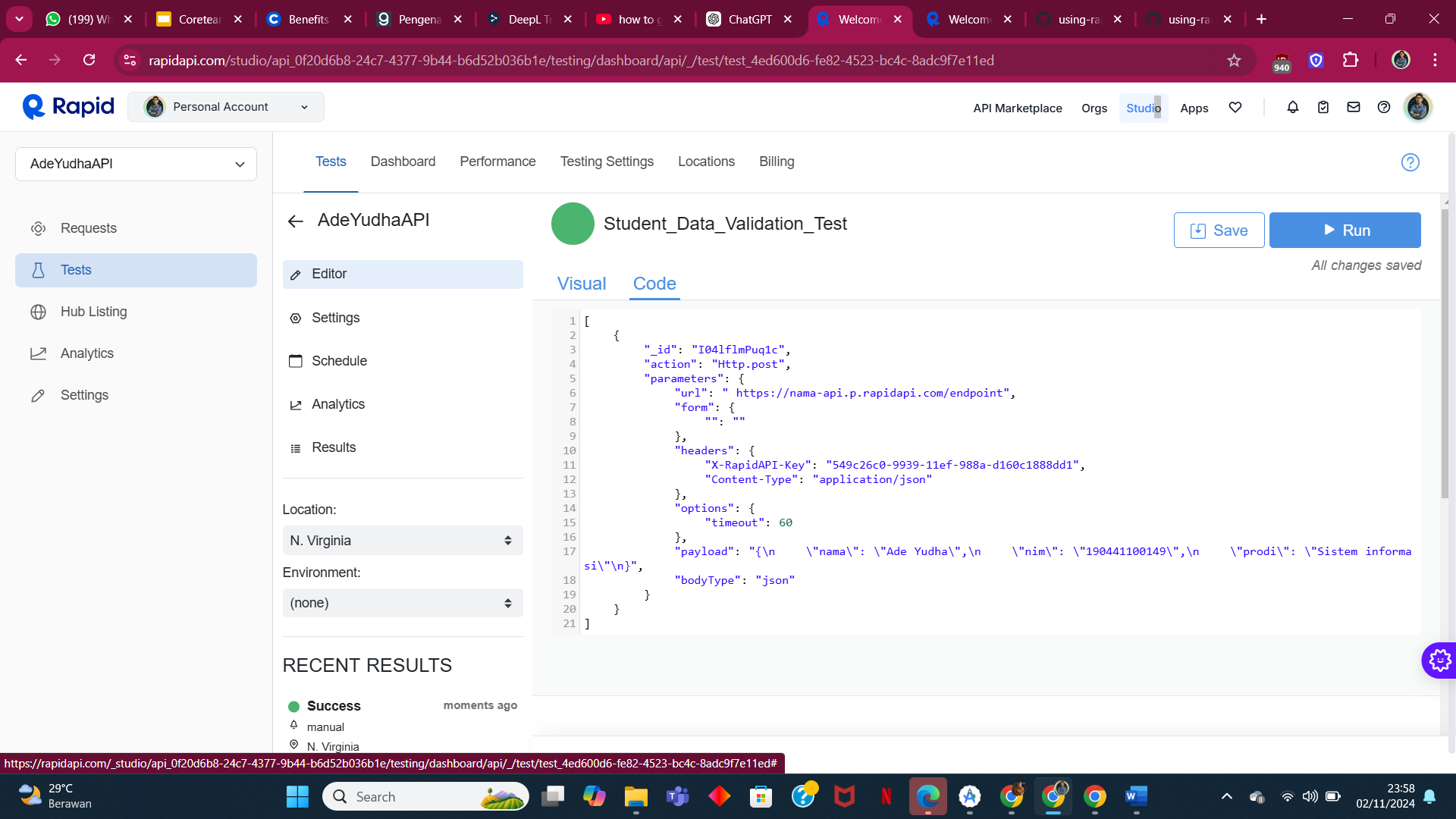
Body



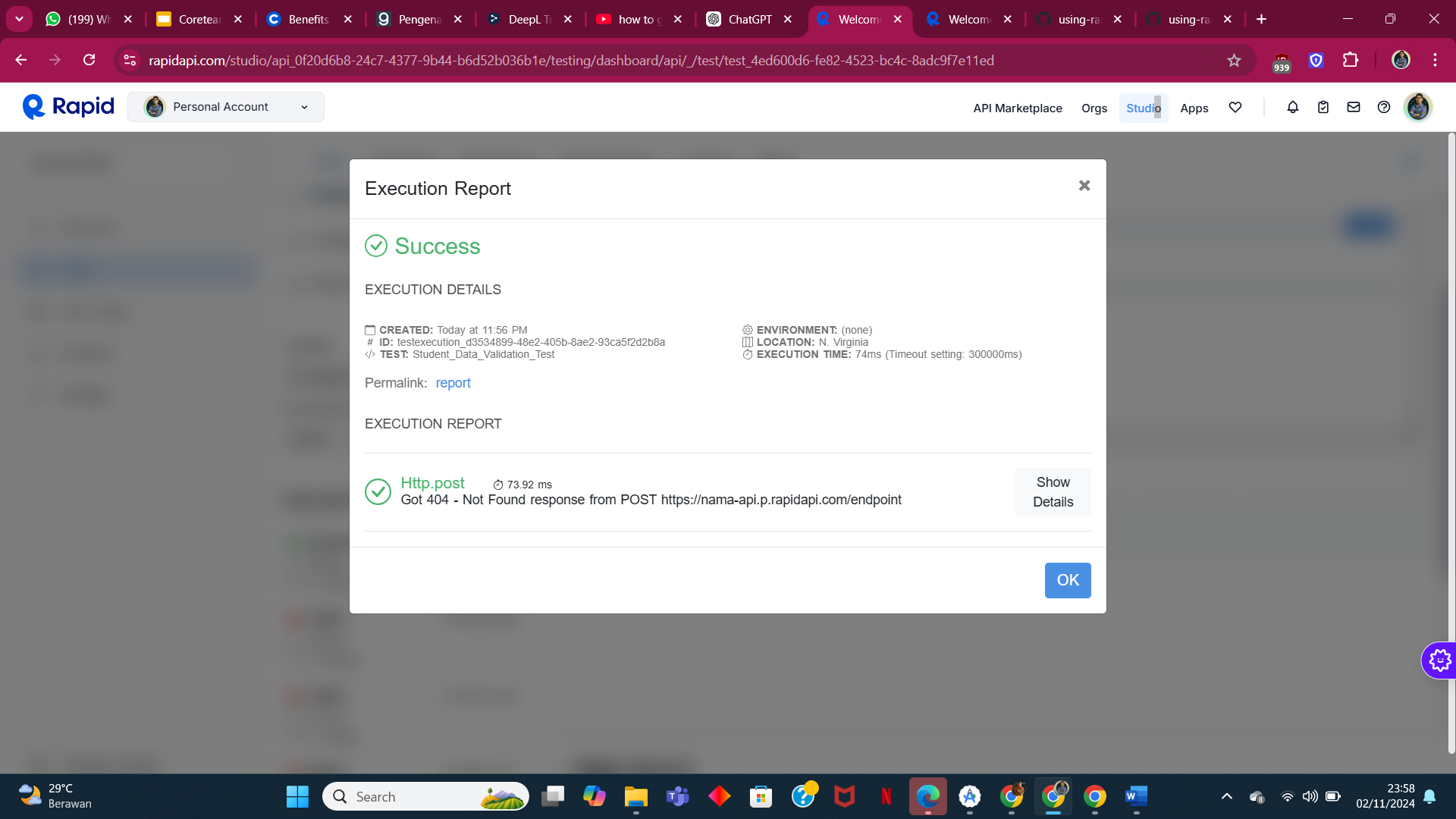
Options



SourceCode



View Report



Ujicoba API Chat GTP-O4 sendiri di Visual Studio Code

Source Code:

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>API Request Example</title>

    <link rel="stylesheet" href="css/style.css">

</head>

<body>

    <div class="container">

        <h1>Giraldo Nainggolan AI </h1>

        <div class="input-container">

            <label for="userMessage">Your Message:</label>

            <textarea id="userMessage" rows="4" placeholder="Type your message here..."></textarea>

            <button id="sendBtn">Send Message</button>

        </div>

        <div class="output-container">

            <h2>Response:</h2>

            <pre id="responseOutput">Jangan berharap banyak, 250 karakter aja 😁!</pre>

        </div>

    </div>

    <script src="js/main.js"></script>

</body>

</html>

**style.css**

\* {

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

body {

    font-family: Arial, sans-serif;

    background-color: #f0f0f0;

    display: flex;

    justify-content: center;

    align-items: center;

    height: 100vh;

}

.container {

    background-color: #fff;

    padding: 20px;

    border-radius: 8px;

    box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.1);

    width: 400px;

    text-align: center;

}

h1 {

    font-size: 24px;

    color: #333;

    margin-bottom: 20px;

}

.input-container {

    margin-bottom: 20px;

}

textarea {

    width: 100%;

    padding: 10px;

    font-size: 16px;

    border: 1px solid #ccc;

    border-radius: 4px;

    resize: none;

}

button {

    margin-top: 10px;

    padding: 10px 20px;

    font-size: 16px;

    background-color: #4CAF50;

    color: white;

    border: none;

    border-radius: 4px;

    cursor: pointer;

}

button:hover {

    background-color: #45a049;

}

.output-container {

    margin-top: 20px;

    text-align: left;

}

pre {

    background-color: #f8f8f8;

    padding: 10px;

    border-radius: 4px;

    border: 1px solid #ddd;

    max-height: 150px;

    overflow-y: auto;

    white-space: pre-wrap;

    word-wrap: break-word;

}

**main.js**

const url = 'https://cheapest-gpt-4-turbo-gpt-4-vision-chatgpt-openai-ai-api.p.rapidapi.com/v1/chat/completions';

const options = {

    method: 'POST',

    headers: {

        'x-rapidapi-key': '9173484eeamshca65f65e83c008ap137197jsnc76d37759a45',

        'x-rapidapi-host': 'cheapest-gpt-4-turbo-gpt-4-vision-chatgpt-openai-ai-api.p.rapidapi.com',

        'Content-Type': 'application/json'

    }

};

document.getElementById("sendBtn").addEventListener("click", async () => {

    const userMessage = document.getElementById("userMessage").value;

*// Set up body content for API request*

    const bodyContent = JSON.stringify({

        messages: [

            {

                role: 'user',

                content: userMessage

            }

        ],

        model: 'gpt-4o',

        max\_tokens: 100,

        temperature: 0.9

    });

    try {

        const response = await fetch(url, {

            ...options,

            body: bodyContent

        });

        if (!response.ok) {

            throw new Error(`Error ${response.status}: ${response.statusText}`);

        }

        const result = await response.json();

        document.getElementById("responseOutput").textContent = result.choices[0].message.content;

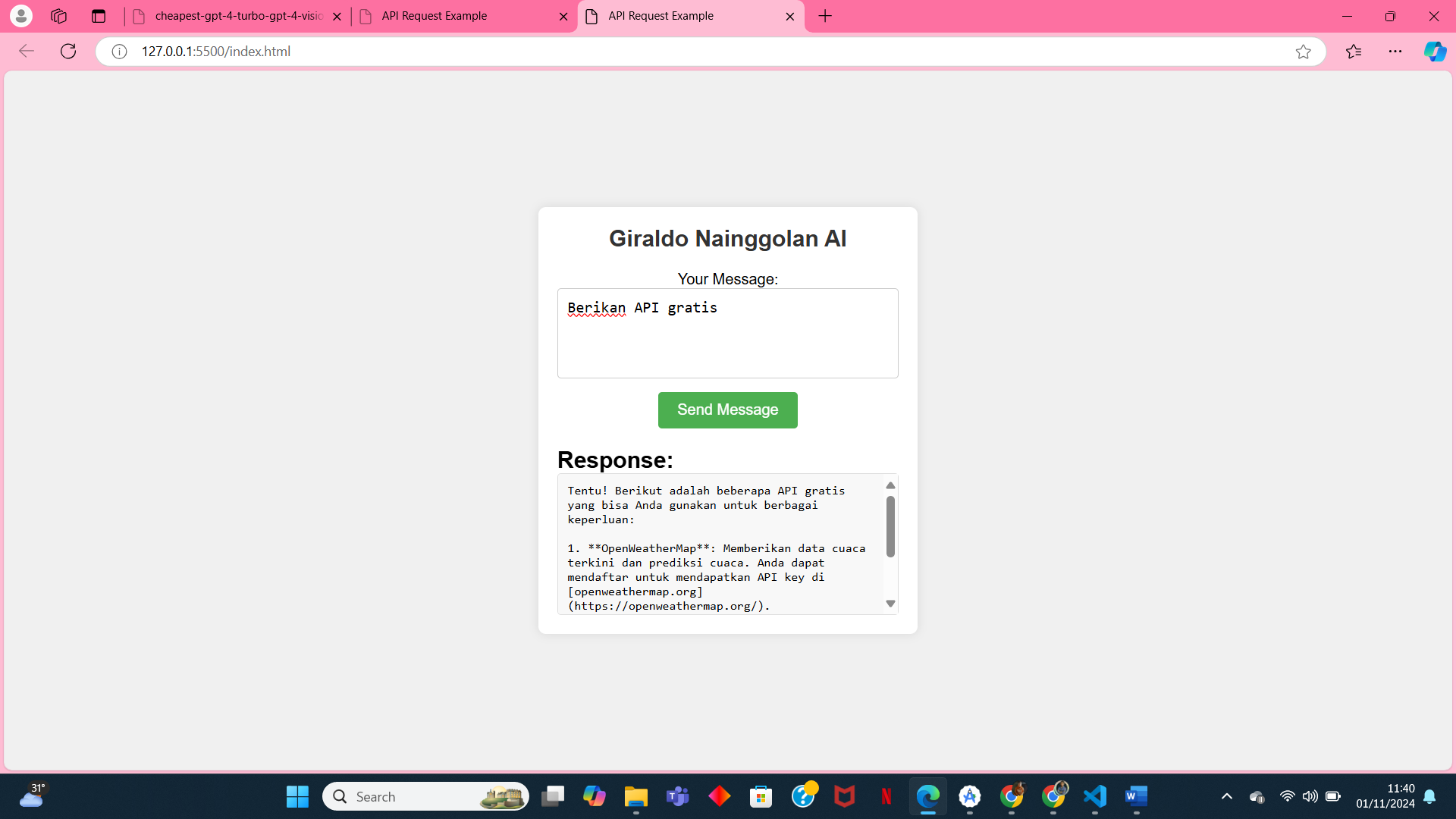
    } catch (error) {

        document.getElementById("responseOutput").textContent = `Error: ${error.message}`;

    }

});

Hasilnya:



**5. Analisis Hasil**

Berdasarkan pengujian di atas:

* Semua endpoint berfungsi sesuai spesifikasi.
* Waktu respons cukup cepat, rata-rata di bawah 200 ms.
* Tidak ada kesalahan dalam payload atau validasi data.

**6. Kesimpulan**

Pengujian REST API menggunakan RapidAPI Studio berhasil. API berfungsi dengan baik pada setiap endpoint yang diuji. Seluruh fitur berhasil diakses sesuai spesifikasi tanpa adanya error atau masalah performa yang signifikan.

**7. Saran Pengembangan**

* Tambahkan fitur autentikasi jika belum diterapkan.
* Pertimbangkan untuk menambahkan caching di endpoint GET untuk meningkatkan performa pada beban tinggi.