Document Title	**Technical Test Software Engineer**		
Document No	TEST/CETTA-TECH/001	**Document Date**	1 August 2024
Document Version	1.0	**Summary of Change**	Initial Technical Test for Software Engineer
Created By	Philipus Silaen *VP System Development*	**Acknowledge By**	Sanjaya I.Mayluddin *Chief Technology Officer*

TECHNICAL TEST

Bagian 1: API CRUD untuk Data Lingkungan

1. **Deskripsi Tugas:**
   - Buatlah sebuah API CRUD (Create, Read, Update, Delete) untuk mengelola data lingkungan seperti **sensor pencemaran udara**. Jika menggunakan data real akan mendapatkan nilai tambah (Dataset dapat ditemukan di platform data publik seperti Kaggle, UCI Machine Learning Repository, atau sumber data lingkungan lainnya).
   - Gunakan salah satu dari framework berikut: Laravel, Lumen, Express, atau ElysiaJS. (Jika menggunakan ElysiaJS, Anda akan mendapatkan nilai lebih)
   - Database yang digunakan adalah PostgreSQL.
2. **Spesifikasi API:**
   - Create: Endpoint untuk menambahkan data sensor lingkungan baru.
   - Read: Endpoint untuk mengambil data sensor lingkungan berdasarkan ID dan juga endpoint untuk mengambil data secara keseluruhan.
   - Update: Endpoint untuk memperbarui data sensor lingkungan yang ada.
   - Delete: Endpoint untuk menghapus data sensor lingkungan berdasarkan ID.
   - Additional Requirements:
     - Pastikan untuk menangani validasi data dan error handling dengan baik.
     - Implementasikan autentikasi dan otorisasi (misalnya, menggunakan token API).
     - Sertakan dokumentasi pada repository Anda yang menjelaskan cara menggunakan API dan cara mengatur lingkungan pengembangan.
3. **Pengiriman:**
   - Silahkan unggah kode Anda ke GitHub atau platform repositori kode lainnya.

- Kirimkan link repositori beserta dokumentasi yang diperlukan.

. **Bagian 2: Tampilan dashboard sederhana**

- Buat tampilan dashboard sederhana menggunakan framework laravel atau svelte menggunakan api yang sudah ada di bagian 1.
- **Anda wajib hosting live website tersebut sendiri atau menggunakan layanan hosting aplikasi gratis seperti Heroku, Firebase, Vercel, dll.**

Bagian 3 Logical & Analytical Test

Untuk menjawab bagian ini, Anda dapat menggunakan bahasa pemrograman yang Anda inginkan. Jawaban Anda akan dievaluasi berdasarkan:

- Ketepatan solusi Anda
- Kebersihan, ekspresi, dan kejelasan kode serta logika Anda
- Kesederhanaan dan efisiensi kode

Harap diingat bahwa ketepatan solusi bukanlah satu-satunya kriteria evaluasi, terutama bagi kandidat yang berpengalaman. Faktor lain, seperti memahami pertanyaan dengan benar mempertimbangkan kasus-kasus khusus, dan menghindari logika atau struktur data yang berlebihan, juga akan diperhitungkan.

Pertanyaan 1: Pengolahan Token untuk Data Kualitas Udara

Anda sedang bekerja dengan sistem yang menerima bacaan kualitas udara dari berbagai sensor. Setiap bacaan diwakili sebagai token string dalam format area:parameter:nilai, di mana ":" digunakan sebagai pemisah antara token.

Tulis program yang menerima token string dan melakukan operasi berikut:

- ingest(string): Menerima sebuah string dan menyimpannya dalam koleksi.
- appearance(prefix): Menerima string sebagai input dan mengembalikan nilai yang dinormalisasi antara 0 dan 1, yang mewakili persentase token yang disimpan yang dimulai dengan prefix yang diberikan.

Contoh:

```
ingest('area1:pm25:55')
```

```
ingest('area1:pm10:35')
ingest('area1:pm25:45')
ingest('area1:pm25:60')
ingest('area2:pm10:70')

appearance('area1')
# > 0.8
appearance('area1:pm25')
# > 0.6
appearance('area2')
# > 0.2
appearance('area1:pm25:45')
# > 0.2


ingest('area3:pm25:55')
ingest('area3:pm10:35')
ingest('area3:pm25:45')
ingest('area3:pm25:60')
ingest('area3:pm10:70')

appearance('area3:pm25')
# > 0.6
appearance('area4')
# > 0
```

Jelaskan space and time complexity dari solusi Anda.

Pertanyaan 2: Desain Cache untuk Data Sensor

Desain dan implementasikan struktur data untuk sistem cache yang digunakan dalam menyimpan dan mengambil data sensor. Cache harus mendukung operasi berikut:

- get(key): Mengembalikan nilai yang terkait dengan key yang ditentukan jika ada dalam cache, jika tidak, mengembalikan **-1**.
- put(key, value, weight): Mengaitkan nilai dengan key dalam cache, di mana nilai tersebut dapat diambil nanti dengan get(key).

Cache memiliki kapasitas tetap, dan ketika kapasitas ini tercapai, key dengan skor terendah harus dihapus. Skor dihitung menggunakan rumus:

```
weight / [ln(current_time - last_accessed_time + 1) + 1]
```

Implementasikan cache dengan tujuan mengoptimalkan kompleksitas waktu dari get(key). Usahakan agar kompleksitas waktu rata-rata dari get(key) adalah konstan.
Di akhir jawaban Anda, berikan dan jelaskan kompleksitas komputasi dari **get(key)** dan **put(...)** dalam notasi **Big O**.

Pertanyaan 3:

Pertimbangkan program berikut yang menghitung deret matematika terkait pengukuran kualitas udara:

```
function recur(n, cur) {
    if (!cur) {
        cur = 0;
    }
    if (n < 2) {
        throw new Error('Input tidak valid');
    }
    if (n === 2) {
        return 1 / n + cur;
    }
    return recur(n - 1, cur + 1 / (n * (n -1)));
}
```

Tulis program yang melakukan perhitungan yang sama seperti fungsi recur, tetapi tanpa menggunakan rekursi.

Notes:

Jika menggunakan svelte dengan tailwind.css untuk dashboard menjadi nilai tambah.

(Hasil test bagian 1 dan 2 silahkan upload ke github dan kirim linknya, dokumentasi pada repository menjadi nilai tambahan. Untuk hasil test bagian 3 kirim dalam bentuk tautan gist di gist.github.com, dengan 3 file bernama answer[1-3].md (atau ekstensi yang benar untuk bahasa pemrograman yang dipilih, misalnya, answer1.c / answer2.md / answer3.py). Baca baik-baik kembali requirementnya, kemampuan anda dalam membaca requirement juga dinilai disini).

Deadline: 5 hari sesudah tes ini diberikan

Hasil test dapat dikirimkan ke :

people@zicare.id
cc: philipus.silaen@cetta.tech

DOCUMENT CHANGES

NO	CHANGE BY	POSITION	CHANGE DATE	VERSION
1	Philipus Silaen	*VP System Development*	1 August 2024	1.0