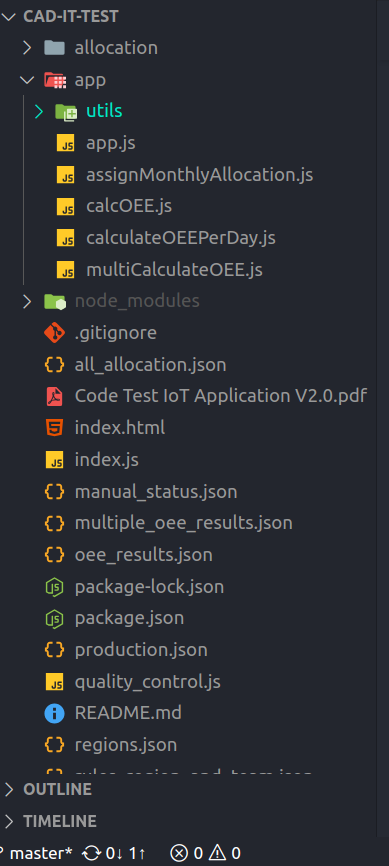
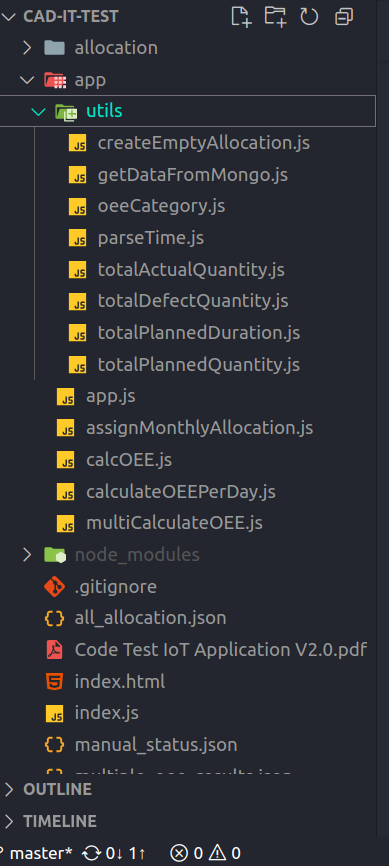
REVISI

NOTE:

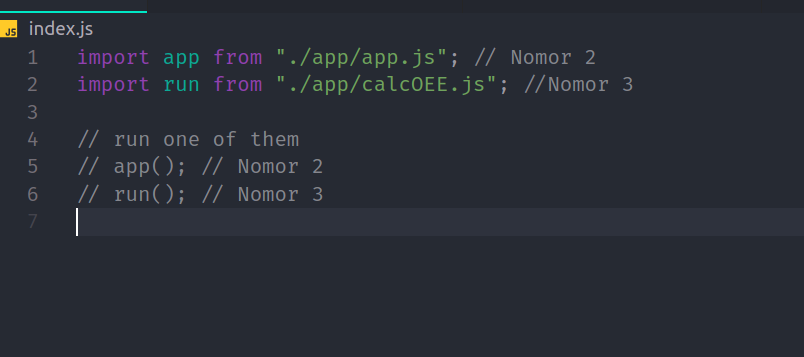
Dokumentasi program, penjelasan mengenai fungsi fungsi helper dan sebagainya, saya jelaskan lebih detail pada file README di github saya. Silahkan kunjungi link berikut : <https://github.com/rmdhnz/cad-it-test>

Struktur Folder





file utama yang melakukan bootstrapping : [index.js](http://index.js)

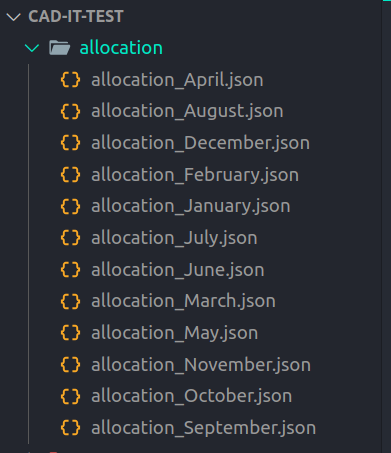


1. Downtime Aggregation

Because I just need to answer 2 questions, I will not explain about it, but I Have solved in my github.

1. Regional Arrangements

Hasil dari penyusunan tiap bulannya disimpan dalam bentuk file JSON pada sebuah folder bernama “allocation”



dan juga saya buat yang merangkum keseluruhan hasil dari alokasi, tersimpan pada file “all\_allocation.json”

Pada soal ini, database yang digunakan adalah regions.json, teams.json dan juga rules\_region\_and\_team.json.

File yang menjalankan algoritma ini adalah “[app.js](http://app.js)” pada folder app.

Algoritma yang digunakan :

* Inisialisasi Region dan Kuota

Masing-masing region (misalnya Jakarta = 2 orang, Papua = 14 orang) disiapkan dengan kapasitas kosong.

* Menempatkan Tim Tetap (Fixed Teams)

Tim seperti *Accountant* selalu di *Central Papua*, atau *Country Manager dan HRD* selalu di *Jakarta*. Tim-tim ini langsung dimasukkan ke region-nya dan kapasitasnya dikurangi sesuai banyak tim yang masuk ke region tersebut.

* Menempatkan Tim yang Harus Bersama (Grouped Teams)

Misalnya, *PM 1*, *Lead Mining 1*, dan *Lead Mining 2* harus berada di region yang sama. Algoritma mencari region yang masih cukup kapasitasnya, lalu menempatkan mereka bersama-sama ke region tersebut.

* Menempatkan Tim Lain yang Belum Dialokasikan

Tim-tim yang tidak punya aturan khusus akan dimasukkan secara acak menggunakan fungsi random ke region yang masih memiliki sisa kuota.

* Pengulangan Tiap Bulan  
  Proses ini dilakukan untuk setiap bulan (Januari–Desember), dan hasilnya disimpan dalam file JSON per bulan, seperti allocation\_January.json, allocation\_February.json, dst.
* Mencatat Sisa Kuota

Di setiap hasil alokasi bulanan, sistem mencatat juga berapa sisa kuota yang tidak terpakai di tiap region.

* Melakukan check

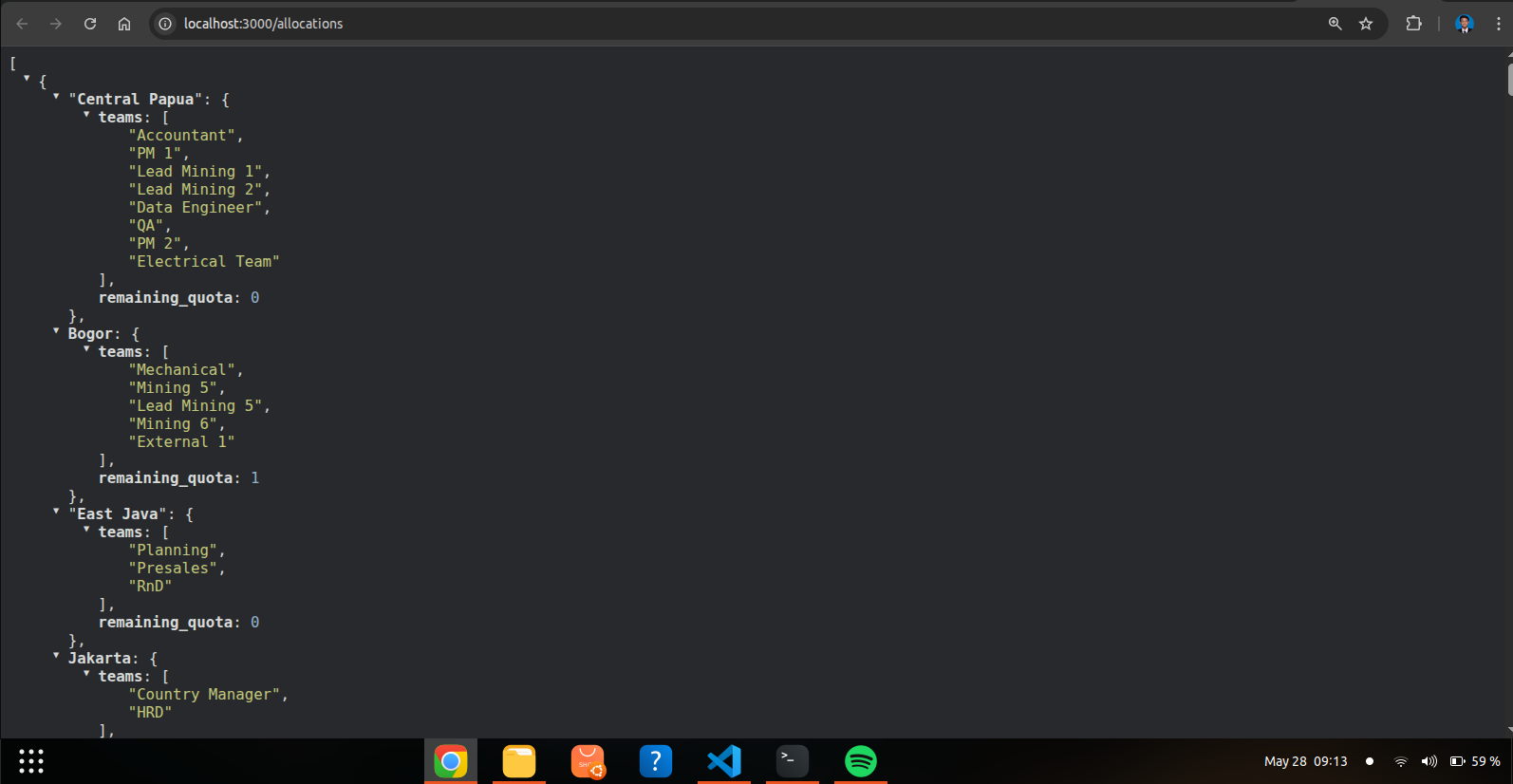
Setelah semua tim sudah dialokasikan, dilakukan pengecekan apakah ada tim yang belum dialokasikan, dengan meninjau isi dari variabel “teamSizeLeft”. apabila bernilai kosong artinya semua tim sudah dialokasikan, karena setiap 1 tim sudah dialokasikan, fungsi tersebut melakukan delete data untuk tim tersebut

Lalu, apabila semua tim sudah teralokasi, maka seharusnya tiap bulan remaining\_quota yang tersedia tidak lebih dari 2.

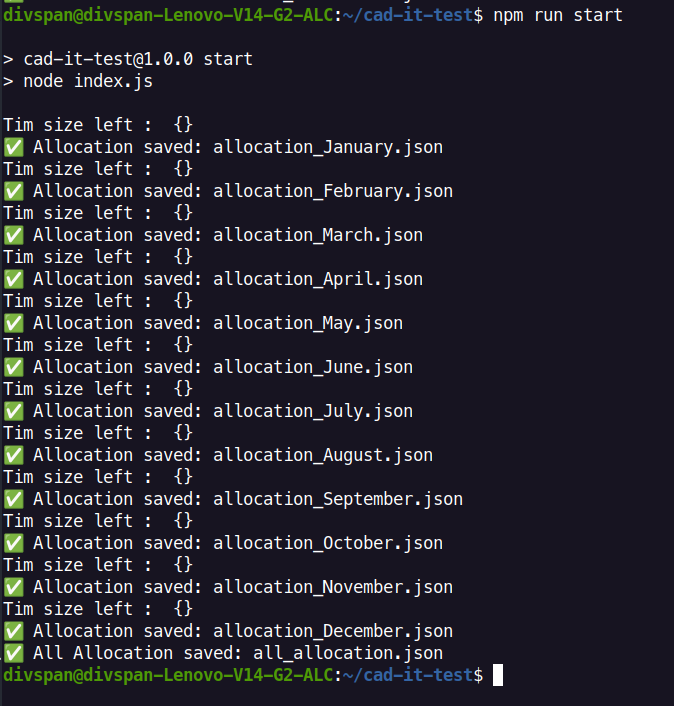
Selanjutnya dilakukan pengecekan juga apakah kombinasi suatu region bulan ini sama atau tidak dengan bulan sebelumnya, apabila sama, maka akan throw error dan algoritma pengalokasian diulang lagi dari awal.

Hasilnya tersimpan untuk masing masing bulan pada folder allocation, dan keseluruhan pada file all\_allocation.json .

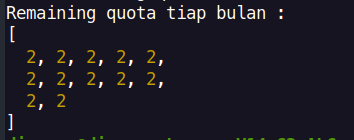
berikut ini merupakan tampilan REST API Endpoint nya



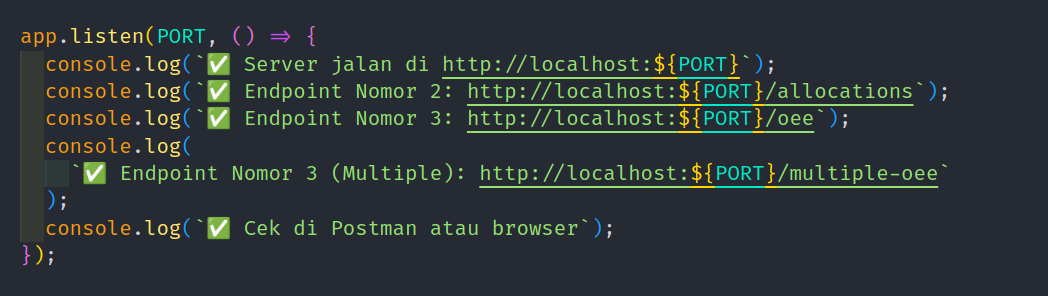
dan ini untuk terminal



jalankan node quality\_control.js untuk melihat hasil ini:







1. OEE Calculation

Pada soal ini, database yang digunakan adalah status.json dan production.json. Alur pengerjaannya ialah mencari Availability, Performance, dan juga Quality dan menghitung OEE. Hasil akhir OEE akan diklasifikasikan sesuai dengan tabel yang tersedia.

Algoritma yang digunakan :

1. Load data dari file js
2. DIlakukan Grouping

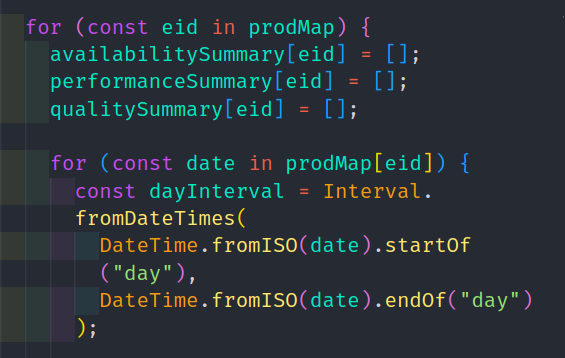
Data produksi dan status digrup berdasarkan:

* 1. equipment\_id
  2. tanggal (date)

1. DIlakukan iterasi untuk tiap equipment id dan tanggal

kita hanya perlu menghitung A,P, dan Q karena semua variabel yang dibutuhkan untuk menghitung A,P,Q sudah saya sediakan fungsinya di folder utils, kita hanya perlu melakukan import dan callback function

Setelah didapat A,P,Q lanjut dihitung OEE dan nilai dari OEE berkisar 0 sampai dengan 1.



pada potongan script berikut, nested loop tersebut berarti kita menghitung dulu untuk A,P,Q untuk semua tanggal di 1 equipment.

Kemudian terdapat juga bagian split status untuk tengah malam

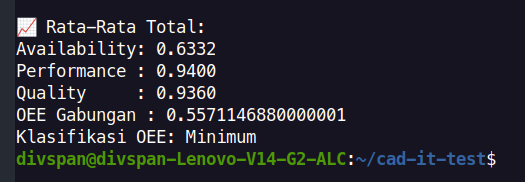


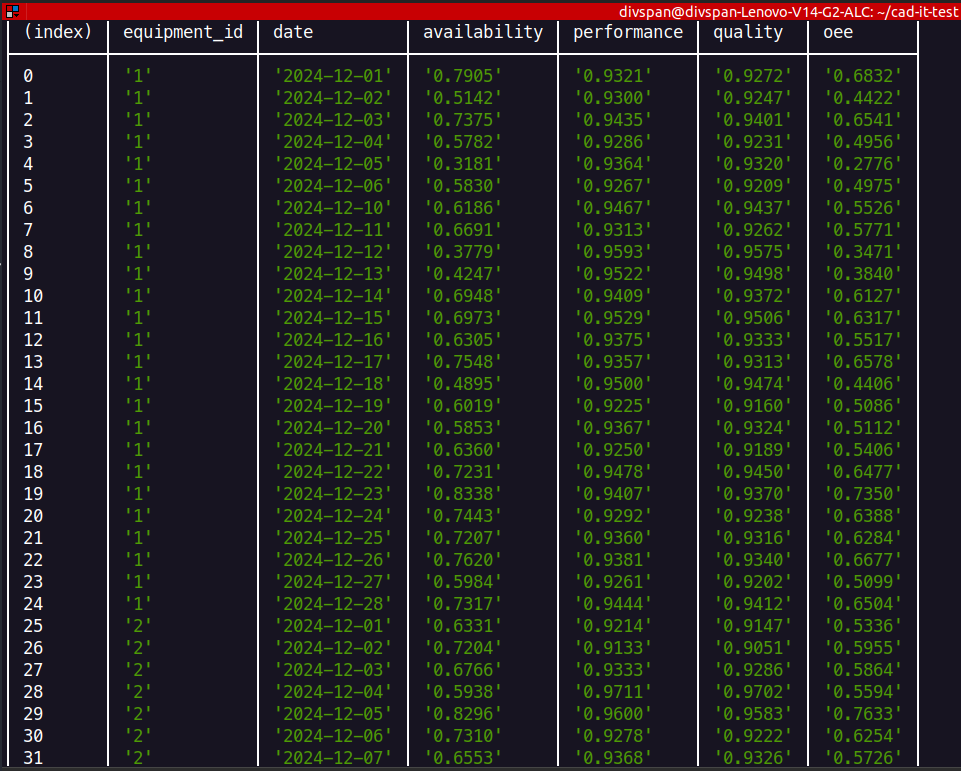
Jika status tidak melewati tengah malam (tanggal mulai dan selesai sama), langsung dimasukkan tanpa dipecah.

Jika status melewati tengah malam, gunakan loop while untuk memecahnya:

* currentStart: penanda awal periode.
* endOfDay: pukul 23:59:59 hari itu.
* currentEnd: akhir segmen saat ini (sampai end, atau hanya sampai endOfDay).
  + - Push segmen waktu ke splitStatuses
* Pindahkan titik awal ke hari berikutnya (menit pertama setelah currentEnd).
* Fungsi mengembalikan dalam bentuk json

Berikut tampilan pada terminal :





dan ini pada REST API Endpoint :

