1. Anda memiliki sebuah API yang mengembalikan data JSON. Bagaimana cara Anda mengambil data dari API tersebut menggunakan Golang?
    - Menggunakan package **net/http** untuk mengirim request HTTP ke endpoint API.
    - Mengambil response dari server dan pastikan request berhasil (status code 200).
    - Menggunakan package **encoding/json** untuk parsing data JSON ke dalam struktur Go yang sesuai (struct atau map).
    - Mengelola Kesalahan yang mungkin terjadi saat membuat request, menerima response, atau parsing data JSON.
2. Anda mendapat laporan bahwa ada masalah dengan performa aplikasi yang dibangun dengan Golang. Bagaimana cara Anda menemukan dan memperbaiki masalah tersebut?
    - Menggunakan monitoring tools seperti **grafana**, untuk mengumpulkan informasi, seperti latency atau konsumsi cpu & memory yang tinggi. Juga degan melihat apakah kasus ini terjadi pada semua service/user, atau hanya sebagian.
    - Periksa log, terutama untuk mencari kesalahan atau peringatan yang menyebabkan masalah
    - Optimasi kode, dengan memastikan tidak adanya pemanggilan function atau source yang berulang yang tidak perlu. Dan juga memeriksa jika ada blocking yang tidak selesai seperti menunggu channel dari goroutine
    - Optimasi query ke database. Pastikan query ke database optimal, dan index dari field memadai, terutama field yang sering dipakai untuk kondisi search. Juga dengan menggunakan cache seperti redis atau memcache untuk query yang berulang kali dipakai
3. Anda diminta untuk membangun sebuah aplikasi backend dengan Golang. Aplikasi tersebut harus dapat menerima request HTTP dan mengembalikan response berupa data JSON. Bagaimana cara Anda memulai dan mengembangkan aplikasi tersebut?
    - Gunakan package **net/http** untuk membuat endpoint HTTP. Juga gunakan package **encoding/json** untuk mengirim data JSON sebagai respons.
    - Gunakan **http.HandleFunc** untuk menetapkan handler ke endpoint tertentu. Atau gunakan 3rd party package untuk routing dan middleware, seperti **gin-gonic/gin** atau **labstack/echo**, dll
    - Tetapkan port untuk listen and server http menggunakan package net/http. Contohnya: `http.ListenAndServe(":8080", nil)`
    - Umumnya endpoint perlu akses ke database, maka perlu digunakan driver yang tepat. Juga untuk kemudahan kita bisa pakai orm seperti **go-gorm/gorm**

    - Membuat unit testing yang bisa menguji kondisi berhasil dan gagal dalam proses
    - Menjalankan aplikasi, dan mengujinya menggunakan **curl** atau **postman** dan melihat hasil yang didapat.
4. Apa saja yang perlu diperhatikan dalam proses pengembangan aplikasi tersebut?
    - Dokumentasikan kode dan API dengan baik, sehingga mudah dipahami oleh developer lain atau kita saat membuka lagi source kode.
    - Siapkan sistem monitoring seperti **grafana** untuk memantau kesehatan aplikasi. Juga pastikan logging yang baik dan mudah dibaca untuk membantu dalam diagnosis dan debugging.
    - Rencanakan arsitektur yang dapat dengan mudah ditingkatkan (scaled). Gunakan pola desain yang mendukung skalabilitas, seperti microservices.
    - Gunakan middleware untuk menambahkan fungsi-fungsi seperti autentikasi, logging, rate limiting, dan lainnya.
    - Menggunakan HTTPS untuk keamanan komunikasi.
    - Menggunakan token autentikasi seperti JWT, untuk otorisasi dan autentikasi pengguna.
    - Menggunakan validasi input, untuk mencegah kesalahan input ataupun serangan seperti Injection.
    - Tangani kesalahan dengan benar untuk menghindari pengungkapan informasi yang sensitif. Kembalikan respons yang sesuai dan log kesalahan secara aman.