1. Desain Back Backend
    a. Stack Backend

        Berikut adalah *tech stack backend* untuk pengembangan aplikasi antar makanan:

        - Node Js

          Node Js digunakan sebagai *runtime environment*. Alasan penggunaan Node js adalah dengan menggunakan Node js memiliki fitur *built-in http server library* yang menjadikannya mampu menjadi sebuah *web server* tanpa bantuan *software* lainnya seperti apache dan Nginx. Selain itu dengan Node js akan membuat *restful* menjadi ringan dan efisian karena menerapkan konsep *non-blocking* (*Asyncronous*).
        - Express Js

          Express.js adalah sebuah framework aplikasi berbasis *web* yang menggunakan *core* pemrograman Node.js dengan komponen modul Http dan Connect. *Framework* ini dibuat untuk berjalan dalam mesin V8 Chrome, membuatnya berjalan beriringan dengan Node sehingga dapat melakukan kerja dengan sangat cepat.
        - MongoDB

          mongoDB merupakan salah satu database noSQL, yaitu sebuah konsep penyimpanan data non-relational. Model data nya yang berbasis dokumen membuat penggunanya untuk tidak perlu merancang struktur tabel seperti pada SQL.nSebagai basis data NoSQL, MongoDB memilki penyimpanan data yang lebih besar serta *low-cost.*

    b. Desain pattern

        Desain pattern yang digunakan untuk pengembangan back-end aplikasi antar makanan adalah *Model View Controller* (MVC).

        1. Model

           Model merupakan bagian dari program yang bertugas untuk mengatur, memanipulasi serta mengorganisasikan data yang ada pada database. Pada aplikasi ini akan ada tiga model yaitu *user, product*, pesanan.

            - model *user* akan mengatur atau memanage data terkait dengan data user yaitu seperti nama user, email, password, kapan data ditambahkan, dan kapan data diupdate. Model ini nantinya akan dipanggil pada *controller* terkait dengan user seperti ketika proses authentication
            - model *product* untuk mengatur atau memanage data terkait dengan data product yang dijual. data product tersebut seperti nama produk, harga, gambar, kapan product tersebut ditambahkan dan kapan product diupdate.
            - model pesanan digunakan untuk mengatur dan memanage data terkait dengan data pesanan user. pada model ini nantinya akan terdapat *object* yang menyimpan data *user* sebagai pemesan makanan, dan *product* apa yang dipesan oleh *user*.

        2. View

           Bagian yang bertugas untuk menampilkan informasi dalam bentuk *Graphical User Interface* (GUI). Karena pada bagian ini yang akan dibuat adalah *backend* berupa restful api, maka view digunakan untuk menampilkan template *response*.

3. Controller

   Pada *controller* ini akan terdapat logika pemrosesan dari setiap *endpoint* yang telah dibuat. *Controller* menghubungkan serta mengatur model dan *view* agar dapat saling terhubung. Berikut adalah rancangan dari *controller* yang dibuat untuk aplikasi antar makanan :

   - Auth (*controller* untuk menangani proses terkait dengan proses *auth* seperti *register login forget password.*
   - Product (*controller* untuk menangani managemen product seperti CRUD *product*)..
   - Pesanan (*controller* yang berisi logika dari setiap *endpoint* terkait dengan pemesanan *product*)

2. Keamanan dalam pengiriman data

   c. JSON Web Token (JWT)

      JSON Web Token (JWT) adalah standar terbuka (RFC 7519) yang mendefinisikan cara yang ringkas dan mandiri untuk mentransmisikan informasi antar pihak secara aman sebagai objek JSON. JWT token akan dibuat setiap kali *user* berhasil *login*. Tujuan dari JWT itu adalah untuk keamanan dengan cara memastikan bahwa ketika akan melakukan CRUD *product user* harus *login* terlebih dahulu. Dengan menggunakan JWT proses authentikasi untuk mengakses sebuah *endpoint* dapat disederhanakan. Selain itu, penyimpanan dari JWT yang memanfaatkan *local storage* atau *cookies* sebagai media penyimpanan informasi *user* tersebut dapat memangkas beban kerja server.

   d. CORS

      CORS adalah fitur keamanan yang memberi tahu *browser* terkait sebuah *request* dari aplikasi *web* domain lain atau origin lain tersebut diperbolehkan atau tidak untuk mengakses ke RESTful api yang telah dibuat. CORS ini akan digunakan dengan menginstall *library* dari NPM.

4. Menurut pendapat saya terdapat beberapa yang salah pada desain *endpoint* dan terdapat kekurangan pada *endpoint* yang seharusnya dibuat. Berikut adalah penjelasannya:

   a. Pada *endpoint create product*, dokumentasi *endpoint* yang dibuat seharusnya menurut saya pada bagian *request* bentuknya adalah *form* data. Hal tersebut dikarenakan pada saat tambah data *product* seharusnya akan dilakukan *upload* file *image* produk. Begitu pula pada *endpoint update product*. seharusnya *request* menggunakan *form* data.

   b. Pada setiap *endpoint*, dokumentasi dari *response* setiap *endpoint* hanya berupa *response* dengan status 200. Seharusnya akan lebih baik lagi kalo dibuat dokumentasi untuk status *response* yang lainnya seperti ketika gagal dengan status 400.

   c. Pada *endpoint update*, dokumentasi hanya menggambarkan bahwa yang diupdate pada *request* hanya mengirimkan *field name product.*

   d. Pada data *product* mungkin lebih baik lagi kalo *field* nya ditambah informasi *product* seperti *stock product*.

   e. Terkait dengan *endpoint,* apabila memang tujuan nya untuk membuat aplikasi antar-makanan, tentu saja *endpoint* yang tertulis di dokumentasi tersebut hanya menggambarkan sebagian *endpoint* saja seperti belum terdapat *endpoint* yang digunakan untuk mengangani pemesanan makanan.