Date:
M T W T F S S

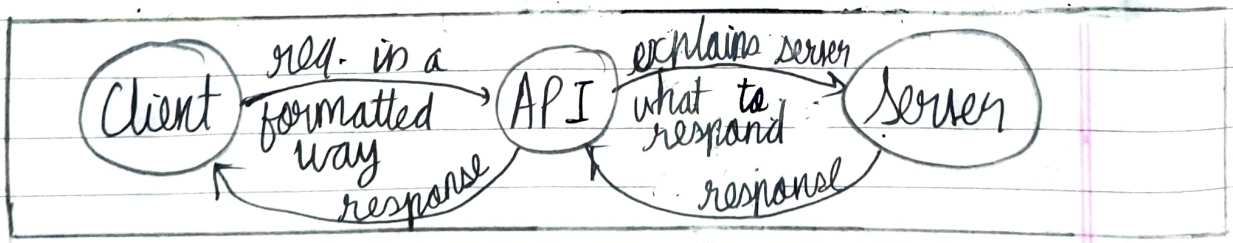
# API Design

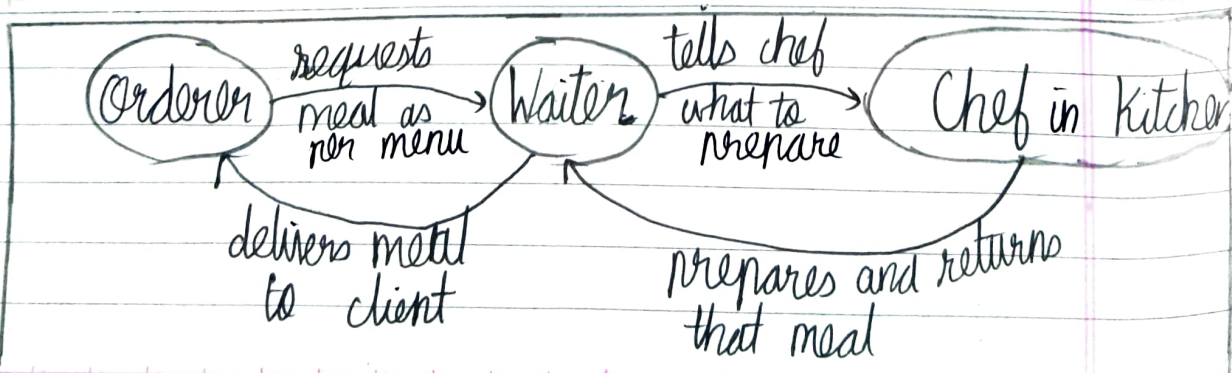
IMP: A "sibling" of sys. design and NOT a "subset"

## Importance of API design:

API → Applica^n Program Interface

- A contract provided by 1 piece of software to another piece of software
- Consists of structured req., res.
  One piece of software says "gimme this info formatted in this way and I'll return this f^n / data / whatever reqd. res. is."
- Analogy: Ordering food in a restaurant



API is at the core of a service's backend.

Also certain software products like Stripe (payment transac" authentica" and conduc" service) have the API as the core/main service (enablin payments)

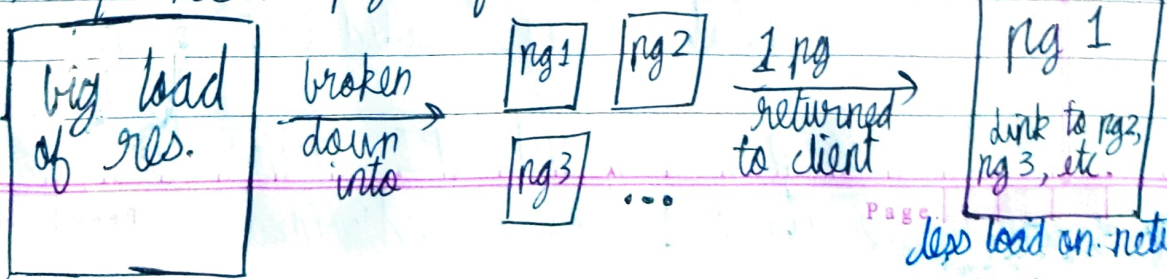
## API Design Importance

1. A lotta customers or services, gonna use your API / depend on your API
2. Any decision in its designing (eg:- name of param. in an endpt.) has huge impacts on users
3. When a new API is developed, its gotta go thru a riggorous design & review process to make it easy to use and update

## Some salient API design features

### Pagina":

When a network req potentially warrants a really large res., the relevant API might be designed to return only a single or couple page(s) of that res. (i.e. a ltd nor" of the res. accompanied by the identifier / token for client to req. next pg. if desired.




less load on network

Date:
M T W T F S S

Pagination is often used to list endpts.

eg:- Endpt. to list videos on the YouTube trendin page

i) Cannot expect just 1 endpt. to return a huge list of vids.

ii) Reason - It wouldn't perform very well on mobile devices due to lower network speeds (unoptimal)

Also, unoptimal cuz most users only scroll thru 1st 10 or 20 vids.

iii) API is designed to respond with only the 1st few vids. of that list. i.e. API res. is paginated.

CRUD oprns.

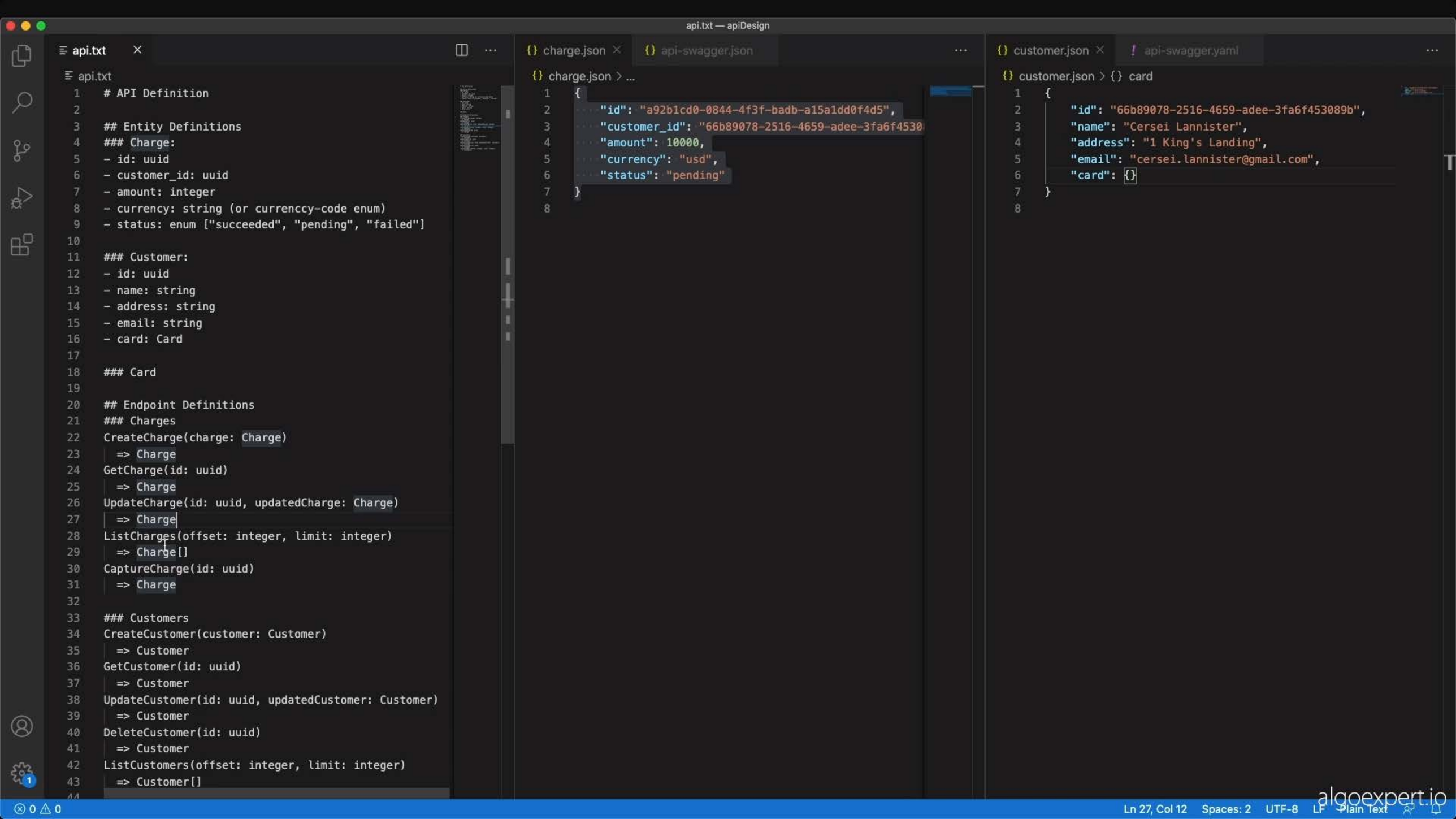
C - Create
R - Read
U - Update
D - Delete

} 4 oprns. servin as a bedrock of a f^n in sys and ∴ find themselves at the core of a lotta APIs.

PRO-TIPS:

(1) Read up APIs of some popular services. eg:- GitHub, Google Cloud IOT

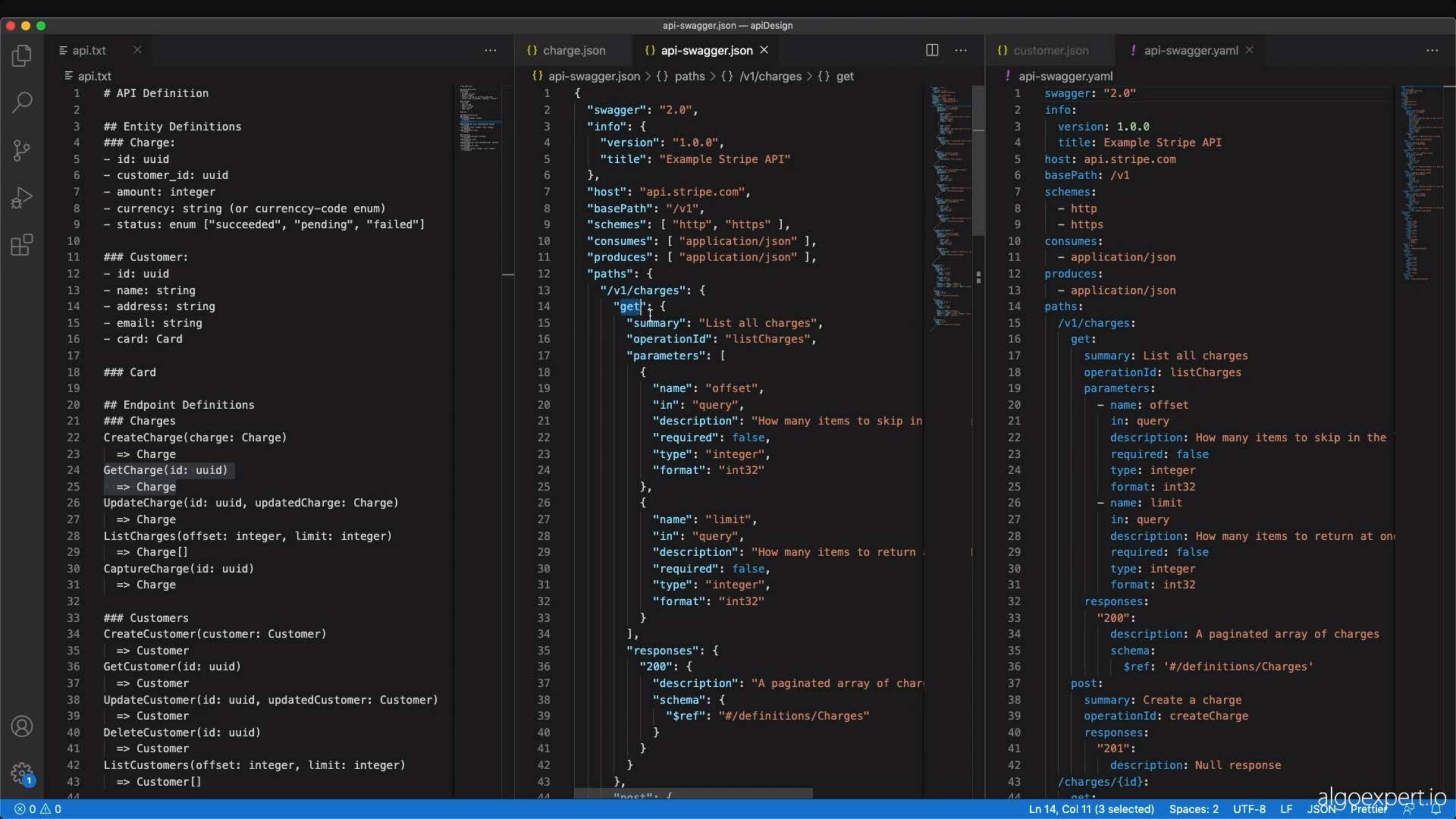
(2) Try to figure out API design for a service, then see its API docs to understand how close you are.




```
# API Definition

## Entity Definitions
### Charge:
- id: uuid
- customer_id: uuid
- amount: integer
- currency: string (or currenccy-code enum)
- status: enum ["succeeded", "pending", "failed"]

### Customer:
- id: uuid
- name: string
- address: string
- email: string
- card: Card

### Card

## Endpoint Definitions
### Charges
CreateCharge(charge: Charge)
  => Charge
GetCharge(id: uuid)
  => Charge
UpdateCharge(id: uuid, updatedCharge: Charge)
  => Charge
ListCharges(offset: integer, limit: integer)
  => Charge[]
CaptureCharge(id: uuid)
  => Charge

### Customers
CreateCustomer(customer: Customer)
  => Customer
GetCustomer(id: uuid)
  => Customer
UpdateCustomer(id: uuid, updatedCustomer: Customer)
  => Customer
DeleteCustomer(id: uuid)
  => Customer
ListCustomers(offset: integer, limit: integer)
  => Customer[]
```

```
{
    "id": "a92b1cd0-0844-4f3f-badb-a15a1dd0f4d5",
    "customer_id": "66b89078-2516-4659-adee-3fa6f4530
    "amount": 10000,
    "currency": "usd",
    "status": "pending"
}
```

```
{
    "id": "66b89078-2516-4659-adee-3fa6f453089b",
    "name": "Cersei Lannister",
    "address": "1 King's Landing",
    "email": "cersei.lannister@gmail.com",
    "card": {}
}
```


api-swagger.json — apiDesign
api.txt
charge.json
api-swagger.json
customer.json
api-swagger.yaml
api-swagger.json > { } paths > { } /v1/charges > { } get
# API Definition
## Entity Definitions
### Charge:
- id: uuid
- customer_id: uuid
- amount: integer
- currency: string (or currenccy-code enum)
- status: enum ["succeeded", "pending", "failed"]
### Customer:
- id: uuid
- name: string
- address: string
- email: string
- card: Card
### Card
## Endpoint Definitions
### Charges
CreateCharge(charge: Charge)
=> Charge
GetCharge(id: uuid)
=> Charge
UpdateCharge(id: uuid, updatedCharge: Charge)
=> Charge
ListCharges(offset: integer, limit: integer)
=> Charge[]
CaptureCharge(id: uuid)
=> Charge
### Customers
CreateCustomer(customer: Customer)
=> Customer
GetCustomer(id: uuid)
=> Customer
UpdateCustomer(id: uuid, updatedCustomer: Customer)
=> Customer
DeleteCustomer(id: uuid)
=> Customer
ListCustomers(offset: integer, limit: integer)
=> Customer[]
{
"swagger": "2.0",
"info": {
"version": "1.0.0",
"title": "Example Stripe API"
},
"host": "api.stripe.com",
"basePath": "/v1",
"schemes": [ "http", "https" ],
"consumes": [ "application/json" ],
"produces": [ "application/json" ],
"paths": {
"/v1/charges": {
"get": {
"summary": "List all charges",
"operationId": "listCharges",
"parameters": [
{
"name": "offset",
"in": "query",
"description": "How many items to skip in
"required": false,
"type": "integer",
"format": "int32"
},
{
"name": "limit",
"in": "query",
"description": "How many items to return
"required": false,
"type": "integer",
"format": "int32"
}
],
"responses": {
"200": {
"description": "A paginated array of char
"schema": {
"$ref": "#/definitions/Charges"
}
}
}
},
swagger: "2.0"
info:
version: 1.0.0
title: Example Stripe API
host: api.stripe.com
basePath: /v1
schemes:
- http
- https
consumes:
- application/json
produces:
- application/json
paths:
/v1/charges:
get:
summary: List all charges
operationId: listCharges
parameters:
- name: offset
in: query
description: How many items to skip in the
required: false
type: integer
format: int32
- name: limit
in: query
description: How many items to return at on
required: false
type: integer
format: int32
responses:
"200":
description: A paginated array of charges
schema:
$ref: '#/definitions/Charges'
post:
summary: Create a charge
operationId: createCharge
responses:
"201":
description: Null response
/charges/{id}:
Ln 14, Col 11 (3 selected)
Spaces: 2
UTF-8
LF
JSON
Prettier
algoexpert.io