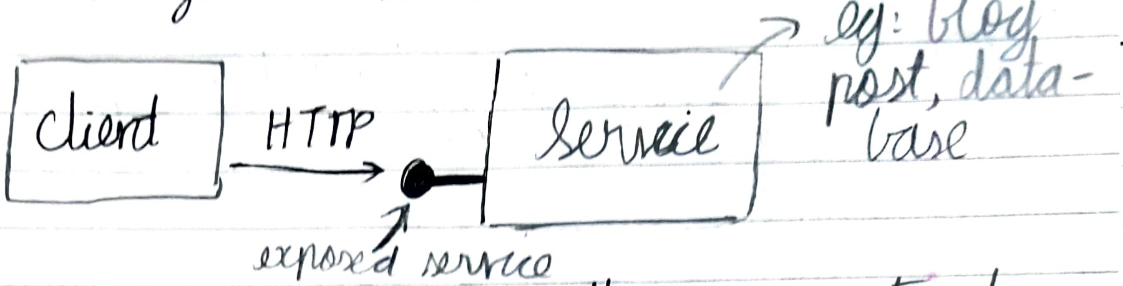
Analogy for API :

$\longrightarrow$ Client / orderer — API / Waiter — Server / Kitchen

HTTP: foundaⁿ of internet comm. everytime you load webpage in browser, it's makin HTTP req. to server

REST API → (App. Prog. Interface)

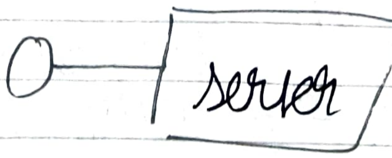
1) App itself is the ~~client~~ client or the front-end part
2) Under the hood, it needs to talk to the server or backend to get or save data.



3) This communicaⁿ happens using the HTTP protocol (the same protocol that powers our web)
4) Most apps these days follow this client-server architecture
5) On the server, we expose a bunch of services that are accessible via the HTTP protocol.
6) The client can then call the services by sending HTTP req.

uses HTTP and std like JSON so can be used by any Prog. lang. ← REST → Representational State Transfer
- This is basically a conven" for building these HTTP services
- relies on a stateless, client-server protocol, mostly HTTP
- We use simple HTTP protocol princips to provide support to

CRUD ops {
- Create data
- Read data
- Update data
- Delete data



Ex :- We have a company called VIDLY for rentin out movies. We have a client app to manage our customer list.
On the server we should expose a service and a service like this :

(1) $\underbrace{\text{http}}_{\text{use https to exchange data on secure channel}}\text{://}\underbrace{\text{vidly.com}}_{\text{domain}}\text{/}\underbrace{\text{api}}_{\text{not compulsory but a lotta companies}}\text{/customers}$

follow this conven^n of includin api in address to expose their restful service

can be after domain (like here) or a sub-domain like api.vidly.com

(2) Client can send HTTP req to this endpt. to talk to ~~this~~ our service

~~We can~~

(3) /customers : This part of address is referred to as RESOURCE

We can expose our resources such as customers movies, rentals and various endpts. It is our endpt. to work with the customers.
All the ops around customers such as creating a customer / updating a customer etc. would be done by sendin an HTTP req. to this endpt.

(4) The type of HTTP req. determines the kind of op. Every HTTP req has a verb / method that determines its type or inten^n

HTTP meths
i) GET $\xrightarrow{\text{for}}$ gettin data. eg:- reading ~~to~~ Fetch API.
ii) PUT $\xrightarrow{\text{for}}$ updatin data. eg:- web forms submission (using GET is unsafe)
iii) POST $\xrightarrow{\text{for}}$ creatin data. eg:-
iv) DELETE $\xrightarrow{\text{for}}$ deletin data.
v) HEAD → same as GET but no body returned
vi) OPTIONS → returns supported HTTP meths
vii) PATCH → update specific resources

cust = customer

Endpt. :- URI/URL where ~~API~~ HTTP req are sent to

To get list of all customers, we should send an HTTP get req to this address.

Method	Inten^n	Req. (sent by us to address endpt.)	Resp. (sent by server)
GET	Get list of all customers	GET /api/customers plural, so list of customers	arr. of customer obj. [{id: 1, name: ' '}, {id: 2, name: ' '},] ....
	To get single customer	GET /api/customer/1 customer referenced thru id	cust. obj. {id: 1, name: ' '}
PUT	To update a cust.	PUT /api/cust/1 specify id of cust. to be updated {name: ' '} cust. obj. included in body of req with updated properties in cust. obj	Updates cust. with given id according to values given {id: 1, name: ' '}
DELETE	To delete a cust.	DELETE /api/custs/1 no need to include cust. obj. in req. body cuz all we need to delete cust is an id	
POST	To create a cust.	POST /api/custs plural cuz accessing whole cust. list to add new cust to it {name: ' '} → new cust obj.	Gets new cust. obj and creates the cust. for us {id: 1, name: ' '}

The REST ful conven" in short

GET	/api/customers
GET	/api/customers/1
PUT	/api/customers/1
DELETE	/api/customers/1
POS	/api/customers

RESOURCE

We expose our <u>custs.</u> using a simple, meaningful address and suppor various ops. around em. such as creatin or updatin em. using std. HTTP methods.

<u>What is an API?</u>

1) API → Applica^n Prog. Interface.
2) APIs are everywhere → web apis, ~~a~~ api s in computer, OS, smartphone, some refrigerators etc.
3) It is a contract provided by one piece of software to another piece of software.
4) Consist of structured req. and resp.
~~5~~ One piece of software says gimme this info formatted in this way and I'll give you back this f^n or data or whater that resp. may be
5)

Orderer	Waiter	Kitchen
Client	API	Server

Page.