Intro to REST APIs

Developer Student Clubs

IIEST Shibpur

HTTP/HTTPS

- **H**yper**T**ext **T**ransfer **P**rotocol / **S**ecure
- Protocol of the web
- Stateless
- Responsible for communication b/w client & web servers
- Examples
    - Fetch request, Form Submission, Visiting any website

- While in HTTPS, data sent over the protocol is secured using SSL/TLS

HTTP Verbs

GET: Retrieve resource
POST: Create resource
PUT: Update resource
DELETE: Delete resource

HTTP Status Codes

1xx: Informational
2xx: Success
3xx: Redirection Message
4xx: Client Error
5xx: Server Error

Some common HTTP Status Codes

200 -	OK
201 -	CREATED
304 -	NOT MODIFIED (can continue with the cached version)
400 -	BAD REQUEST
401 -	UNAUTHORISED
403 -	FORBIDDEN (here identity of client is known to server)
404 -	NOT FOUND
500 -	INTERNAL SERVER ERROR
503 -	SERVICE UNAVAILABLE

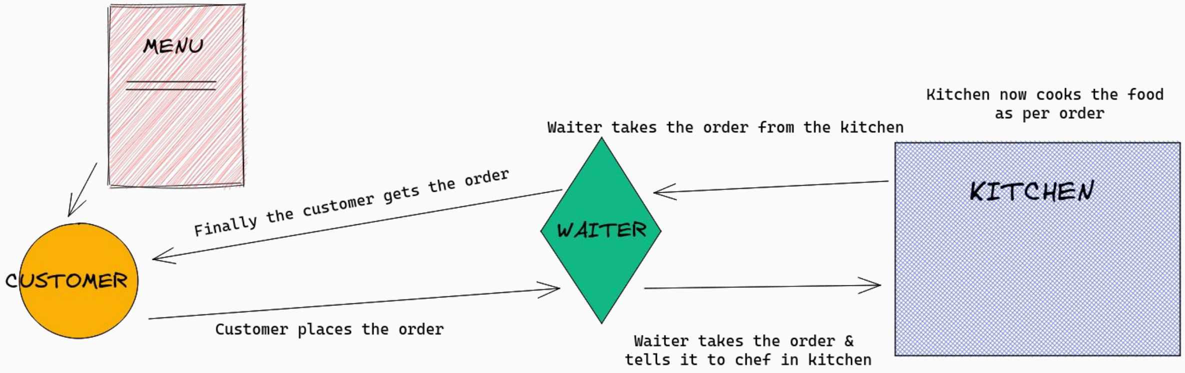
https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods
https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

API

Hi, I'm also called **Application Program Interface.**

I will be your ***messenger***.

I only take your **request**,
deliver it "***somewhere***" &
give you back the **response** ^.^



Kitchen now cooks the food as per order

Waiter takes the order from the kitchen

Finally the customer gets the order

Customer places the order

Waiter takes the order & tells it to chef in kitchen

REST API

aka **REpresentational State Transfer**

Architecture style to design network application

sometimes people get me confused with **RESTful API**, well we're pretty much the same.
Although, in terms of **RESTful Services**, it concludes to services using **REST API** as their communication method.

- All about communication, Client Server communication protocol, almost always HTTP
- Stateless
- Treats server objects as resources which can be created/retrieved/updated/deleted

Benefits

- Standardized & Simple
- High performance due to caching support
- Highly scalable as stateless
- Not bound to any specific programming language

To conclude, API is a messenger, REST lets us use the request/response to format those messages now let's move on to building the APIs