# What is an API?

An application programming interface is a set of guidelines/rules on how applications interact. An API allows one app to retrieve info from another app. Ex. One application is the customer, and another is the cook, where as the API is the server that tells requests and gives responses to those requests.

Three main components:

* Request (make request to server)
* Response (receive response from server)
* Resource (response contains data regarding this)

Anatomy:

* Endpoint (The url being requested for)
* Method (The type of the request)
* Headers (Additional info for the client or server)
* Body (Information sent to the server)

# What is HTTP?

Hypertext Transfer Protocol is a protocol generally used by web services for serving HTML documents. A client requests to a server for a resource, and the server sends a response.

HTTP methods:

* GET (used to request data from a server)
* POST (used to send data to the server)
* PATCH (partially modify resource on server)
* PUT (updates the entire resource on the server)
* DELETE (used to delete a specified resource)
* HEAD (used to get data about the resource’s header without a body. Similar to get but without the Body)
* CONNECT (used to open a tunnel that can be used to access websites using SSL)
* OPTIONS (used to obtain information about the communication options that are available between the requested URL and the server)

Status codes of a response:

* Information is 100s
* Success is 200s
* Redirection is 300s
* Client error is 400s
* Server error is 500s

Safe Method: does not alter the server’s state apart from logging

Idempotent Method: multiple calls yield the same result and leave the server in the same state.

# What is a REST API?

Follow the design principles of REST architecture. REST API’s communicate via HTTP requests to perform CRUD (create, read, update, delete) operations Ex. Send a POST request to CREATE a new user.

Principles:

* Client-Server Separation: Application which is requesting the resource (client) and application which has the resource (server) are kept independent from each other.
* Stateless: Servers are not allowed to store any data related to the client. No session or authentication state allowed
* Cacheable: resources should be able to cache themselves are either on the client or server side. Response from the server will contain information about whether the resource can be cached or not and for how long. This will improve the performance of the client by making it easier to load the resource.
* Layered System: there can be multiple intermediaries between the client and the server such as security, traffic, redirection handlers that the client and server are not aware of.
* Uniform Interface: All requests and responses in a REST API should follow a common protocol. This way the client and server can interact with each other independent of the architecture they are based on.

# HTTP headers

When client requests to server, they can pass additional information as a part of the request via HTTP headers. Headers can contain information about the type of data that the client is sending to the server.

* Request headers: pass info about the request including type of request, URL to endpoint, authentication details, cache policy as well as the user agent
  + Accept: informs the server about the type of data that the client can understand
  + Accept-encoding: informs the server about the type of encoding the client is able to understand
  + Authorization: used to pass credentials which let the server authenticate a client and provide access to protected resources
* Accept-Language: lets server know which local is preferred and understood by the client
  + Cache-control: holds caching instructions for the client and server
* Response Headers: contain information about the resource body, which is sent in a response
  + Content-type: indicates the media type of the resource before any form of content-encoding
  + Content-encoding: used to decode the message payload to obtain the original format of the payload. It is used primarily to do lossless compression of the payload
  + Content-language: indicates the language intended for a particular type of audience.
  + Content-location: indicates an alternative location for the response. Based on the accept request header, the server can respond with different response headers
* Payload headers: contain information about the payload like the length of the message, range of the resource carried in the payload, encoding etc. These headers can be present in both the request and response messages.
  + Content-Length: size of the message body in bytes
  + Content-range: position of a partial message in a full body message