Understanding of API

API stands for Application Programming Interface. It acts like a middleman that allows two different software systems to talk to each other.

API is a bridge that lets one app use features of another app without building them from scratch.

Example: Uber and Google Maps

- Uber is one application (project).
- Google Maps is another separate application.

When you book a ride in Uber, it shows you the map, your route, and nearby drivers but Uber didn't create the map itself.

Instead, Uber uses Google Maps API to access map features like location, directions, traffic, etc.

How it works:

- Uber sends a request to Google Maps API like: "Show me the location of the user and the route to the destination."
- Google Maps API responds with the map data.
- Uber shows this data to the user.

Example: Restaurant and API

The Setup:

- Customer = Your app (like Uber, Facebook, etc.)
- Waiter = API (middleman)
- Kitchen = The server or another system (like Google Maps, database, etc.)

You (the customer) sit at a table and look at the menu. You tell the waiter (API) what you want to eat. The waiter (API) goes to the kitchen (server) and tells the chef your order. The kitchen prepares the food. The waiter (API) brings the food back to your table.

In software terms:

- You make a request using the API.
- The API sends that request to the server.
- The server processes it and sends back the response.
- The API delivers that response to your app.

Types of APIs

1. Public API (Open API)

Anyone can access it. Available to the public, often with API v keys. Used to encourage developers to build on top of a service. Usually well-documented and easy to integrate.

Example:

- Google Maps API – Any app can use it to show maps or directions.
- OpenWeather API – Anyone can use it to get weather data.

2. Partner API

Shared only with selected partners. Not open to the public Requires business agreements or approval. Offers more control over usage and data sharing.

Example:

- Uber & PayPal Integration – Uber uses PayPal's Partner API to offer in-app payments.
- A flight aggregator using APIs from specific airlines to fetch prices and seat availability.

3. Private API (Internal API)

Used only inside a company. Hidden from external developers. Connects internal systems like web frontend and backend. Improves development speed and modularity within an organization.

Example:

- A bank's internal API used by its own mobile app to fetch account balances.
- Amazon using internal APIs to connect their product, payment, and shipping systems.

Type	Access Level	Who Uses It	Example
Public	Open to everyone	Any developer	Google Maps API
Partner	Restricted to partners	Approved third parties	Uber using PayPal API
Private	Internal only	Inside the organization	Internal API for order tracking

API Provider vs API Consumer

1. API Provider Creates and exposes the API. Handles the logic, data, and functionality behind the scenes. Maintains documentation, authentication, rate limits, etc.

2. API Consumer Uses the API to access data or features. Sends requests and receives responses from the provider.

What is a URL

URL stands for Uniform Resource Locator. It's the web address you enter in a browser to visit a website or access a resource (like an API endpoint, image, or document).

```
https://www.example.com:8080/products/search?q=phone#reviews
```

Part	Name	Explanation
`https`	Protocol (Scheme)	Tells the browser how to communicate (e.g., HTTP, HTTPS, FTP)

Part	Name	Explanation
`www.example.com`	Domain (Host)	The web server's address (domain name or IP address)
`:8080`	Port (optional)	Port number the server is listening on (default is 80 for HTTP, 443 for HTTPS)
`/products/search`	Path	Specific resource or location on the server
`?q=phone`	Query String	Data passed to the server (used for search, filters, etc.)
`#reviews`	Fragment (Anchor)	Points to a section within the page (not sent to the server)

Understanding of REST API

REST stands for Representational State Transfer.

It is an **architecture style for designing web APIs** like rules for how software should talk to each other over the internet.

- REST APIs use URLs to access data.
- REST APIs use HTTP methods (GET, POST, etc.)
- REST APIs use JSON format to send/receive data.

1. Endpoint

An endpoint is a specific URL used to access something. Like a door to a resource.

Example Endpoints:

```
/movies/ → All movies
/movies/123/ → Movie with ID 123
```

2. HTTP Methods (CRUD)

Method	Action	Meaning	Example
GET	Read	Get data	`/movies/` → get all movies
POST	Create	Send new data	`/movies/` + new movie JSON
PUT	Update	Replace existing data	`/movies/123/` + updated movie data
DELETE	Delete	Remove data	`/movies/123/` → delete movie

3. Headers

Headers are like ID cards sent with a request. They tell the server extra information.

```
Content-Type: application/json
Authorization: Bearer your-token
```

4. Data / Body

When you send data (POST or PUT), it goes in the body. Format is usually JSON.

```
{
  "title": "Inception",
  "year": 2010,
  "rating": 8.8
}
```

Understand url below

```
https://www.api.movielist.com/movies/
```

return complete list and we can perform GET and POST

```
https://www.api.movielist.com/movies/123/
```

return or connect to individual item and perform GET, PUT, DELETE