**A Brief Overview of Web Api :**

A **Web API** (Web Application Programming Interface) is a set of rules and protocols that allows different software applications to communicate and exchange data over the internet. It1 essentially defines how a client (e.g., a web browser, a mobile app, or another server) can request and receive information or functionality from a server.

Here's a brief overview of key aspects:

* **Communication over HTTP:** Web APIs primarily use the Hypertext Transfer Protocol (HTTP), the same protocol that powers the web. This means they leverage standard HTTP methods like GET (to retrieve data), POST (to send data), PUT (to update data), and DELETE (to remove data).
* **Data Formats:** Data exchanged between clients and servers via Web APIs is typically formatted in **JSON** (JavaScript Object Notation) or **XML** (Extensible Markup Language). JSON is currently more common due to its lightweight nature and ease of parsing in web applications.
* **Endpoints:** A Web API exposes one or more "endpoints," which are specific URLs that represent different resources or functionalities on the server. For example, https://api.example.com/users might be an endpoint to access user data.
* **Request-Response Model:** The interaction with a Web API follows a request-response model:
  + A client sends a **request** to a specific API endpoint (e.g., an HTTP GET request to retrieve user data).
  + The server processes the request and sends back a **response**, which usually contains the requested data (in JSON or XML) and a status code indicating the success or failure of the operation.
* **Client-Side vs. Server-Side:**
  + **Server-side Web APIs** are built by developers to expose their application's data or functionality to external clients. This is what most people refer to when they talk about "Web APIs." Examples include APIs for social media platforms, weather services, or payment gateways.
  + **Client-side Web APIs** are built into web browsers and allow JavaScript to interact with the browser's functionality or the surrounding environment. Examples include the DOM (Document Object Model) API for manipulating HTML/CSS, the Geolocation API for accessing a user's location, or the Fetch API for making network requests.
* **Statelessness:** Many Web APIs, particularly those following the **RESTful** architectural style, are stateless. This means that each request from a client to a server contains all the information needed to understand the request, and the server doesn't store any client-specific context between requests.
* **Authentication and Authorization:** To secure access to data and functionality, Web APIs often implement authentication (verifying the identity of the client) and authorization (determining what the authenticated client is allowed to do). Common methods include API keys, OAuth, or JWT (JSON Web Tokens).
* **Ubiquity and Impact:** Web APIs are fundamental to modern web development. They enable:
  + **Microservices architectures:** Breaking down large applications into smaller, independent services that communicate via APIs.
  + **Integration:** Allowing different applications to seamlessly share data and functionality (e.g., embedding a YouTube video or a Google Map on a website).
  + **Mobile apps:** Providing the backend data and logic for mobile applications.
  + **Data access:** Governments and businesses often provide public Web APIs to access large datasets.

In essence, Web APIs act as a universal language that allows disparate software systems to communicate and collaborate, forming the backbone of the interconnected digital world.

| **Verb** | **Action** | **Description** |
| --- | --- | --- |
| **GET** | Read | Retrieve data from the server (e.g., a list of users or a single product). |
| **POST** | Create | Submit data to the server to create a new resource. |
| **PUT** | Update/Replace | Update an existing resource (typically replaces it completely). |
| **DELETE** | Delete | Remove a resource from the server. |

