Api meaning & Types

What is an API?

An application programming interface is a software interface that helps in connecting between the computer or between computer programs. It is an interface that provides the accessibility of information such as weather forecasting. In simple words, you can say it is a software interface that offers service to other pieces of software.

Api Types

1. **Public APIs**.
2. **Private APIs**.
3. **Composite APIs**.
4. **Partner APIs**.

1. Public APIs

- **Definition**:
  - Public APIs, also known as **Open APIs**, are APIs that are accessible to external developers or the general public. These APIs are designed to be used by anyone, including developers from outside the organization that created the API.
- **Key Characteristics**:
  - **Open Access**: Typically, these APIs are available over the internet and can be accessed by anyone who has the proper credentials, like an API key.
  - **Documentation**: Public APIs are usually well-documented to facilitate easy integration by third-party developers.
  - **Scalability**: Designed to handle large volumes of requests from potentially millions of users.
- **Examples**:
  - Social media APIs like the Twitter API, which allows developers to interact with Twitter's data.
  - Payment processing APIs like Stripe, which allow developers to integrate payment services into their applications.

2. Private APIs

- **Definition**:
  - Private APIs are restricted APIs intended for use within an organization or by a specific group of developers, often for internal development purposes. These APIs

are not exposed to the public and are used to enhance internal systems, tools, and processes.

- **Key Characteristics**:
  - **Restricted Access**: Access to these APIs is limited to internal developers or specific partners. They are not accessible to the general public.
  - **Security**: Since they are used internally, private APIs often have stricter security measures to protect sensitive data.
  - **Customization**: Private APIs are typically tailored to the specific needs of the organization, often to streamline internal processes or integrate with other internal systems.
- **Examples**:
  - APIs used within a company to connect internal applications like an HR management system, a payroll system, or a CRM (Customer Relationship Management) system.
  - An API used by a company's mobile app to connect to its backend services.

3. Composite APIs

- **Definition**:
  - Composite APIs are designed to allow clients to make a single API call to access multiple endpoints or services. These APIs aggregate data from multiple sources and return a single response, making them highly efficient for complex operations.
- **Key Characteristics**:
  - **Efficiency**: By bundling multiple API requests into one, Composite APIs reduce the overhead of making multiple network requests. This is particularly useful when interacting with microservices, where a single operation may involve multiple services.
  - **Complexity**: Composite APIs can handle complex business logic, integrating data from various sources to present a unified result.
  - **Use Cases**: Commonly used in scenarios where a single operation needs data from multiple services or databases.
- **Examples**:
  - A dashboard API that retrieves data from various services like user profiles, order history, and payment details in a single API call.
  - An e-commerce application that uses a Composite API to get product details, reviews, and inventory status in one request.

4. Partner APIs

- **Definition**:
  - Partner APIs are APIs that are shared with specific business partners and are not open to the general public. These APIs facilitate collaboration between two or more businesses by allowing them to integrate their systems and share data or services.
- **Key Characteristics**:

    - **Controlled Access**: Partner APIs are not public; they are accessible only to specific partners who have agreed to certain terms of use.
    - **Collaboration**: These APIs enable businesses to collaborate on projects, share data, or provide services to one another.
    - **Security and SLAs**: Often, these APIs are governed by Service Level Agreements (SLAs) that define the terms of use, availability, and performance metrics.
- **Examples**:
    - An airline company providing a Partner API to travel agencies to access flight availability and pricing.
    - A payment processor providing an API to an e-commerce platform to facilitate payment processing services.

Summary of the Explanation:

- **Public APIs** are open for anyone to use and integrate into their applications, often with easy access and thorough documentation.
- **Private APIs** are restricted to internal or specific usage within an organization, offering higher security and customization.
- **Composite APIs** allow multiple services or endpoints to be accessed with a single API call, optimizing efficiency and handling complex operations.
- **Partner APIs** are shared with specific partners for collaborative purposes, with controlled access and often under formal agreements like SLAs.

CORS Policy (Cross-Origin Resource Sharing) - Summary

CORS (Cross-Origin Resource Sharing) is a security feature implemented by web browsers to control how web pages can request resources from a different domain (origin) than the one from which the page was served.

Key Points:

- **Same-Origin Policy**: By default, browsers follow the same-origin policy, which restricts web pages from making requests to a different domain than the one that served the web page. This is a security measure to prevent malicious websites from accessing sensitive data on another domain.
- **CORS**: CORS is a mechanism that allows servers to specify who can access resources on their domain by including specific HTTP headers in the response. It is a way to relax the same-origin policy.
- **Headers Involved**:
    - `Access-Control-Allow-Origin`: Specifies which origin(s) are allowed to access the resource. It can be a specific domain or `*` to allow all domains.
    - `Access-Control-Allow-Methods`: Specifies the HTTP methods (e.g., GET, POST, DELETE) allowed when accessing the resource.
    - `Access-Control-Allow-Headers`: Specifies which headers can be used in the actual request.

    - `Access-Control-Allow-Credentials`: Indicates whether credentials (like cookies) can be included in the requests.
- **Preflight Requests**: For certain requests, especially those that modify data (e.g., POST, DELETE), the browser first sends an HTTP OPTIONS request (preflight) to check if the actual request is safe to send.

Why CORS is Important:

- **Security**: CORS helps protect users from malicious cross-origin attacks like Cross-Site Request Forgery (CSRF).
- **Flexibility**: It allows servers to control access to their resources while enabling legitimate cross-origin requests, such as from APIs used in web applications.