API

(Application Programming Interfaces)

1. **What is an API?**  
   An API, or Application Programming Interface, is a set of rules and protocols that allows different software applications to communicate with each other. It defines the methods and data formats that applications can use to request and exchange information.
2. **Types of APIs**:
   * **Web APIs**: These are APIs that are accessed over the web using HTTP requests. They are commonly used for building web services, microservices, and integrating with third-party services.
   * **Library APIs**: These are APIs provided by programming libraries or frameworks that allow developers to interact with the functionality provided by the library.
   * **Operating System APIs**: These are APIs provided by operating systems that allow applications to interact with system resources such as files, processes, and hardware devices.
3. **RESTful APIs vs. SOAP APIs**:
   * **RESTful APIs**: Representational State Transfer (REST) APIs use standard HTTP methods (GET, POST, PUT, DELETE) to perform CRUD (Create, Read, Update, Delete) operations on resources. They typically use JSON or XML as data formats and are lightweight and scalable.
   * **SOAP APIs**: Simple Object Access Protocol (SOAP) APIs use XML-based messaging protocol for communication. They are more rigid and have strict standards for message format and protocol.
4. **API Documentation**:
   * Good API documentation is crucial for developers to understand how to use an API effectively. It should include:
     + Endpoint URLs and methods.
     + Request and response formats.
     + Authentication requirements.
     + Error handling.
     + Usage examples and code snippets.
   * Tools like Swagger, OpenAPI, and Postman are often used to create and document APIs.
5. **API Security**:
   * APIs need to be secured to prevent unauthorized access and protect sensitive data. Common security measures include:
     + Authentication (e.g., API keys, OAuth tokens).
     + Authorization (e.g., role-based access control).
     + Encryption (e.g., HTTPS).
     + Rate limiting and throttling.
6. **API Versioning**:
   * As APIs evolve over time, it's essential to have a strategy for versioning to maintain backward compatibility and ensure that existing clients don't break when changes are made. Versioning can be done using URL paths, request headers, or query parameters.
7. **API Best Practices**:
   * Follow RESTful principles for designing APIs, such as using nouns for resources and HTTP methods for actions.
   * Keep APIs simple, intuitive, and consistent.
   * Provide clear and concise documentation.
   * Version APIs to manage changes and updates.
   * Monitor API usage and performance.
8. **API Testing**:
   * API testing involves validating the functionality, reliability, performance, and security of APIs.
   * Techniques include functional testing (testing individual API endpoints), integration testing (testing interactions between APIs), load testing, and security testing.
   * Tools like Postman, SoapUI, and JMeter are commonly used for API testing.

Understanding APIs and how to design, use, and manage them effectively is essential for building modern software applications that leverage the capabilities of various services and systems.

**How do APIs Work?**

The working of an API can be clearly explained with a few simple steps. Think of a client-server architecture where the client sends the request via a medium to the server and receives the response through the same medium. An API acts as a communication medium between two programs or systems for functioning. The client is the user/customer (who sends the request), the medium is the **application interface programming**, and the server is the backend (where the request is accepted and a response is provided). Steps followed in the working of APIs –

* The client initiates the requests via the APIs URI (Uniform Resource Identifier)
* The API makes a call to the server after receiving the request
* Then the server sends the response back to the API with the information
* Finally, the API transfers the data to the client

APIs are considered safe in terms of attacks as it includes authorization credentials and an API gateway to limit access so as to minimize security threats. To provide additional security layers to the data, HTTP headers, query string parameters, or cookies are used.

## Types of APIs

There are three basic forms of API –

### 1. WEB APIs

A **Web API** also called Web Services is an extensively used API over the web and can be easily accessed using the HTTP protocols. A Web **application programming interface is** an open-source interface and can be used by a large number of clients through their phones, tablets, or PCs.

### **2. LOCAL APIs**

In this type of API, the programmers get the local middleware services. TAPI (Telephony Application Programming Interface), and .NET are common examples of Local APIs.

### **3. PROGRAM APIs**

It makes a remote program appear to be local by making use of RPCs (Remote Procedural Calls). SOAP is a well-known example of this type of API.

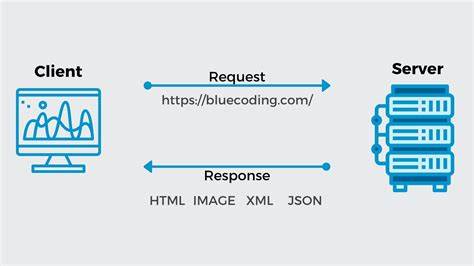
**Few other types of APIs:**

* **SOAP (SIMPLE OBJECT ACCESS PROTOCOL):**It defines messages in XML format used by web applications to communicate with each other.
* **REST (Representational State Transfer):**It makes use of HTTP to GET, POST, PUT, or DELETE data. It is basically used to take advantage of the existing data.
* **JSON-RPC:**It uses JSON for data transfer and is a lightweight remote procedural call defining a few data structure types.
* **XML-RPC:**It is based on XML and uses HTTP for data transfer. This API is widely used to exchange information between two or more networks.

**What are REST APIs?**

REST stands for Representational State Transfer, and follows the constraints of REST architecture allowing interaction with RESTful web services. It defines a set of functions (GET, PUT, POST, DELETE) that clients use to access server data. The functions used are:

* GET (retrieve a record)
* PUT (update a record)
* POST (create a record)
* DELETE (delete the record)



THANK YOU