**APPLICATION PROGRAMMING INTERFACE (API)**

Application Programming Interface(API), is a set of rules, protocols, and tools that allows different software applications to communicate with each other. APIs define how different components of software should interact, making it easier for developers to integrate various functionalities into their applications without needing to understand the underlying implementation details.

1. In simpler terms, an API acts as an intermediary that allows different software systems to talk to each other and exchange data or perform actions.
2. It specifies the methods and data formats that applications can use to request and exchange information.
3. APIs can be used for various purposes, such as accessing web services, retrieving data from databases, interacting with hardware devices, integrating third-party services, and more.
4. APIs play a crucial role in modern software development by enabling interoperability between different systems and facilitating the creation of complex, interconnected applications.

**TYPES OF APIS**based on their functionality APIs are classified as fallows  
 1) WEB APIs

2)Library APIs

3) Database APIs

4)Cloud APIs

**Web APIs (HTTP-based APIs):**

These are APIs that use HTTP protocols for communication over the web.

Web APIs are widely used for building client-server applications and enabling interaction between different software systems.

Examples include RESTful APIs, SOAP (Simple Object Access Protocol) APIs, and GraphQL APIs.

**Library APIs:**

Library APIs provide a set of functions or classes that developers can use to interact with a specific programming language or framework.

These APIs are often bundled with software libraries or development kits and help developers perform common tasks efficiently.

Examples include Java API, .NET Framework API, and Python Standard Library.

**Database APIs:**

Database APIs enable applications to interact with databases for storing, retrieving, and manipulating data.

These APIs provide methods for executing database queries, transactions, and managing database connections.

Examples include JDBC (Java Database Connectivity) for Java applications, ADO.NET for .NET applications, and SQLAlchemy for Python applications.

**Cloud APIs:**

Cloud APIs provide access to cloud-based services and resources such as storage, computing power, networking, and machine learning algorithms.

These APIs allow developers to integrate cloud services into their applications and leverage cloud infrastructure for scalability and flexibility.

Examples include AWS (Amazon Web Services) API, Google Cloud API, and Microsoft Azure API.

**REST API**

**\*\*** RESTful APIs are based on the REST architectural style, which emphasizes a stateless client-server communication model and resource-based interactions.

**Statelessness:** One of the key principles of REST is statelessness. This means that each request from a client to the server must contain all the information necessary to understand and process the request. The server should not store any client context between requests. This simplifies the server architecture and improves scalability.

**Client-Server Architecture:** REST APIs follow a client-server architecture, where the client and server are independent components that communicate over the network. The client sends requests to the server, and the server processes those requests and returns responses.

**Uniform Interface:** REST APIs have a uniform interface, which means that they use standard HTTP methods (GET, POST, PUT, DELETE) to perform operations on resources. Each resource is identified by a unique URI (Uniform Resource Identifier), and the client interacts with these resources through HTTP methods.

**Resource-Based:** In REST, data is represented as resources, which can be any object or concept that can be uniquely identified. Resources are typically represented using standard formats such as JSON (JavaScript Object Notation) or XML (eXtensible Markup Language).

**Representation:** Clients interact with resources by exchanging representations of the resource state. These representations can be in various formats, such as JSON, XML, HTML, or others. Clients can request different representations of the same resource using content negotiation.

In REST API operations are performed by using the URL to reach out particular resource.

**URL (uniform resource identifier).**  
the address to locate the resource in on the server.  
It consists of several components, including the protocol (e.g., HTTP, HTTPS), domain name or IP address, port number (optional), path, and query parameters.   
  
Example: **https://www.example.com/search?q=query**

Parts:

**Protocol**(Scheme): https or http

**Host:** [www.example.com](http://www.example.com) 🡪 host where the resources are stored( domain name or ip address) domain name and ip address represent the same (example.com or 127.0.12)

**Path**: /search 🡪 / represents path parameter (to locate the resource in server)

**Query:** q=query 🡪 ? represents query parameter

**HTTP METHODS**To perform the CRUD operation on the resources through the REST api we use HTTP methods.  
  
they are.  
 1) GET: this method is used to retrieve data from the server.

2) POST: To create a data on the server.   
 3)PUT: this method is used to update the existing data.  
 4)DELETE: this methos is used to delete data from the server.   
  
 TESING REST APIs   
for testing the REST apis or any other APIs commonly used tools are

1) postman.

2)insomnia.

3)Swagger.

**TESTING APIs using POSTMAN**

**Postman:** Postman is a widely-used API testing tool that allows users to test, and debug APIs quickly and easily. It provides a user-friendly interface for sending HTTP requests, inspecting responses, and automating tests.

**collections:** a collection in postman is a collection of similar type of Http requests.  
 \*\* in post mam we can create collections like a file hierarchy.  
 \*\* we can share the collection to the ream by importing it.   
**environments:** we can create different environments for the API testing like   
 1) local environments  
 2) dev environments  
 3) QA environments   
 4)production environments

**variables:** there are two types of variables available in postman.  
 1) global variable : this values can be used in any environment.   
 2) environment variables: this can be accessed on respective environments only.   
  
 BASIC TABS on POSTMAN   
 1) params : by using this tab we pass the path parameters and query parametsrs.  
2) Authorization : ApIs are configured with the security mechanisms called Authorization.  
 to send a request to the server we need to get the authorization.   
 for this we pass the required authorization details here. Like access TOKEN.  
3) BODY: here we pass the request body , the data to be created.(used in POST,PUT ).  
4)Pre-requisite Script: here we write the code in JAVA SCRIPT to test and validate the API responses.

**STATUS codes**

STATUS codes: along with the response we get the Status codes. To better understand and validate the response and to understand the errors and their cause.

**\*200 OK:** The request has succeeded. The response typically contains the requested resource.

**\*201 Created:** The request has been fulfilled, and a new resource has been created as a result.

**204 No Content:** The server successfully processed the request but does not need to return any content.

**400 Bad Request:** The server cannot process the request due to a client error, such as invalid input parameters.

**\*\*401 Unauthorized:** The request requires authentication, and the client failed to provide valid credentials.

**403 Forbidden:** The server understood the request but refuses to authorize it.

**404 Not Found:** The requested resource could not be found on the server.

**405 Method Not Allowed:** The HTTP method used in the request is not supported for the requested resource.

**422 Unprocessable Entity:** The server understands the content type of the request entity but was unable to process the contained instructions.

**500 Internal Server Error:** The server encountered an unexpected condition that prevented it from fulfilling the request.

**502 Bad Gateway:** The server, while acting as a gateway or proxy, received an invalid response from the upstream server.

**503 Service Unavailable:** The server is currently unable to handle the request due to temporary overloading or maintenance of the server.