WHAT IS REST

- REST stands for REpresentational State Transfer. REST is web standards-based architecture and uses HTTP Protocol.

- The World Wide Web is the best example of a RESTful system.

REST ARCHITECTURE PRINCIPLES

An Architecture is called RESTful if it conforms with the following 6 architectural principles:

- Uniform interface
- Client-server
- Stateless
- Cacheable
- Layered system
- Code on demand(optional)

REST ARCHITECTURE PRINCIPLES

- **Uniform interface**

The developer of the interface should decide the interface for resources inside the system which are exposed to API consumers and follow religiously. A resource in the system should have only one logical URI and that should provide a way to fetch related or additional data.

- **Client–server**

This essentially means that client application and server application MUST be able to evolve separately without any dependency on each other. A client should know only resource URIs and that's all.

REST ARCHITECTURE PRINCIPLES

- **Stateless**

The Server does not store anything about the latest HTTP request from the client. It treats each request as new.

- **Cacheable**

Caching of data and responses is of utmost importance wherever they are applicable/possible. Caching brings performance improvement for client side, and better scope for scalability for a server because the load reduces.

In REST, caching shall be applied to resources when applicable and then these resources MUST declare themselves cacheable. Caching can be implemented on the server or client side.

REST ARCHITECTURE PRINCIPLES

- **Layered system**

REST allows you to use a layered system architecture where you deploy the APIs on server A, and store data on server B and authenticate requests in Server C, for example. A client cannot ordinarily tell whether it is connected directly to the end server, or to an intermediary along the way.

- **Code on demand (optional)**

Most of the time you will be sending the static representations of resources in form of XML or JSON. But when you need to, you are free to return executable code to support a part of your application e.g. clients may call your API to get a UI widget rendering code.

Each REST request is of one of the following types:

- **GET** – Provides a read only access to a resource. This is used to retrieve data from the backend system.
- **POST** – Used to create a new resource. The request payload contains the data to be created. The response also returns the created data. The response would also include any data generated/calculated by the system.
- **DELETE** – Used to remove a resource. The response in this request does not include any data.
- **PUT** – Used to update an existing resource. The response in an update request does not return any data, as the data is already known to the client
- **PATCH**: Update single properties of an existing resource.