API Design – Assessment

* What are the best practices to design Rest API with example

1. Use nouns for resource identification.

Eg : For Employee Management System, Use

Employee

Department

Address

Faculty

1. Use plural nouns to name a resource - Don't mix up singular and plural, use plural nouns to name a resource.

Eg: Use /employees instead of ./employees

1. Use PUT, POST, and DELETE methods instead of the GET method to alter the state. Do not use GET for state changes
2. In REST, the relationships are often modeled by a sub-resource. Use the following pattern for sub-resources.

Eg: GET  /{employeee}/{employee-id}/{address}

1. Use proper HTTP methods

GET – Fetch a resourece

PUT – updates existing resource

POST – creates a new resource

DELETE – Deletes a resource

1. When the client raises a request to the server through an API, the client should know the feedback, whether it failed, passed or the request was wrong

200 -OK

201 – Created

400- Bad Request

401 – Unauthorised

403 – Forbidden

404 – Not Found

500- Internal Server Error

* What are some architectural styles for creating an API

1. **R**epresentational **S**tate **T**ransfer (REST) is an architectural style that defines a set of constraints to be used for creating web services. **REST API** is a way of accessing web services in a simple and flexible way without having any processing. A request is sent from client to server in the form of a web URL as HTTP GET or POST or PUT or DELETE request. After that, a response comes back from the server in the form of a resource which can be anything like HTML, XML, Image, or JSON. But now JSON is the most popular format being used in Web Services.
2. Simple Object Access Protocol (SOAP) is a message specification for exchanging information between systems and applications. When it comes to application programming interfaces, a SOAP API is developed in a more structured and formalized way. Think of SOAP as being like the national postal service: It provides a reliable and trusted way to send and receive messages between systems (and within enterprise applications). It is older, established, and dependable—but it can be slower than competing architectural styles like REST.
3. GraphQL is an [open-source](https://en.wikipedia.org/wiki/Open-source_software) data [query](https://en.wikipedia.org/wiki/Query_language) and [manipulation](https://en.wikipedia.org/wiki/Data_manipulation_language) language for [APIs](https://en.wikipedia.org/wiki/API) and a query [runtime engine](https://en.wikipedia.org/wiki/Runtime_system). GraphQL enables [declarative](https://en.wikipedia.org/wiki/Declarative_programming) data fetching where a [client](https://en.wikipedia.org/wiki/Client%E2%80%93server_model) can specify exactly what data it needs from an API. Instead of multiple [endpoints](https://en.wikipedia.org/wiki/Service-oriented_architecture) that return separate data, a GraphQL server exposes a single endpoint and responds with precisely the data a client asked for.