**What is API?**

API is a way which is helping you to create seamless connectivity, you don't feel that you are connecting with so much programs. but API do.

API is interactions between applications, data and devices - they all have API’s that allow computers to operate them, and that's what ultimately creates connectivity. API’s

provide a standard way of accessing any application, data or device whether it is shopping from your phone, or accessing cloud applications at work.

**WHAT IS STATELESS**

As per the REST architecture, a RESTful Web Service

should not keep a client state on the server.

Each request

from

client to server must contain all of the information necessary to understand the request, and cannot take advantage of

any stored context on the server. Session state is therefore kept entirely on the client.

**CACHE**

When a consumer requests a resource representation, the request goes through a cache or a series of caches (local cache, proxy cache, or reverse proxy) toward the service hosting

the resource.

If any of the caches along the request path has a fresh copy of the requested representation, it uses that copy to satisfy the request. If none of the caches can satisfy the

request, the request travels all the way to the service (or origin server as it is formally known).

What is Web Service?

Web service is a standardized medium to propagate communication between the client and server applications on the World Wide Web.

The web services can be searched for over the network and can also be invoked accordingly.

When invoked the web service would be able to provide functionality to the client which invokes that web service.

HTTP

**HTTP**

- BASICALLY IS A FOUNDATION OF THE COMUNICATION OF THE INTERNET OF THE WEB

EVERY TIME CPU LOAD AN INTERNET WEB PAGE IN YOUR BROWSER IT IS MAKING HTTP REQUEST TO SERVER SOMEWHERE

IT IS POSSIBLE TO MAKE REQUEST WITH DIFFERENT PROTOCOLS BUT HTTP IS BY FAR MOST USED

The most important difference between the two protocols is the SSL certificate. In fact, HTTPS is basically an HTTP protocol with additional security. However, this additional

security can be extremely important, especially for websites that take sensitive data from its users, such as credit card information and passwords.

**How HTTPS works?**

The SSL certificate encrypts the information that users supply to the site, which basically translates the data into a code. Even if someone manages to steal the data being

communicated between the sender and the recipient, they would not be able to understand it due to this encryption.

But in addition to adding that extra layer of security, HTTPS is also secured via TLS (Transport Layer Security) protocol. TLS helps provide data integrity, which helps prevent

the transfer of data from being modified or corrupted, and authentication, which proves to your users that they are communicating with the intended website.

Users can identify whether a site uses HTTPS protocol by the web address. The very first part of the web address (before the “www”) indicates whether the site uses HTTP or HTTPS

protocols.

So, to recap, the difference between HTTP vs HTTPS is simply the presence of a SSL certificate. HTTP doesn’t have SSL and HTTPS has SSL, which encrypts your information so your

connections are secured. HTTPS also has TLS (Transport Layer Security) protocol that HTTP lacks. HTTPS is more secure than HTTP.

**How to Check HTTP Request and Response on Chrome?**

You can also open the developer console by right clicking on the page and choose “Inspect” option.

Go to “Network” tab and then reload the page. Now you will see the loading time for each single component on the page.

Click on the “Show Overview” icon to remove the timeline so that you can view other details clearly.

Click the page URL on the left bar and go to “Response” tab. (You can also vew the same details under “Preview” tab).

**MICROSERVICES**

REST is a way to implement Microservices. You can implement Microservices in multiple ways.

REST over HTTP is the most popular way to implement Microservices nowadays.

You an use REST for other purposes as well. REST is used in web apps as well as in API design. Many MVC applications also use REST to serve business data.

Microservices is an architecture style to build large scale applications that can be scaled up independently.] In Microservices we follow certain design patterns to make it

loosely coupled.

Microservices is a service-oriented architecture pattern wherein applications are built as a collection of various smallest independent service units

PROBLEMS with tradittional architucture:

-Many enterprices still sit on large monolithic apps with huge unmanagable codebases

-Monolith do not support rapid improvement of functionality

realising a small change requires whole app to be deployed

-Updaets requires growing amount of coordination

-A broken update can cause the entire app to crash new dev-rs must spend months learning codebase

-Monolithic architecture is tightly coupled. Changes in one module of code affect the other

-Change in data model affects the entire database

Microservices

Perform a single business function

focus on doing one thing well Every unit of the entire application should be the smallest, and it should be able to deliver one specific business goal.

Autonomous

can be changed and deployed independently

-Fault isolation is easy. Even if one service goes down, other can continue to function.

-Small Focused Teams. Parallel and faster development

-Interacts with other microservices by using well-defined interfaces

-No cross-dependencies between code bases. You can use different technologies for different Microservices.

**API interview questions**

1. What is an API?  
   2) What is the difference between client, API and Database?  
   3) What is Postman?  
   4) What type of APIs do you know?  
   5) Explain what is Rest API  
   6) What are the key differences between REST API and SOAP?  
   7) List key features of Rest API  
   8) What is ifference between HTTP and HTTPS?  
   9) Briefly explain what is URL?  
   10) What is Microservice Architecture? Why do we use it?  
   11) What is differnce between Monolithic and Microservice Architecture  
   12) What is Rest API endpoint and what are the most common HTTP methods?

1) What is JSON? When do we use it?  
2) What can be stored in JSON?  
3) What are status codes?  
4) What types of status codes do you know?  
5) What HTTP Methods do you know? Describe them  
6) What HTTP Request consists of?  
7) What is path parameters and resource?  
8) What are Headers?  
9) What is the difference between Authorization and Authenticaton?  
10) What type of Authorization/Authenticaton do you know?  
11) What is OAuth?  
12) What is the difference between API Keys, Bearer Tokens and OAuth?

1) What are Query Parameters?  
2) What is the difference between Query, Path and Body Parameters?  
3) What is Content-Type header?  
4) What is Accept header?  
5) What is the difference between Content-Type and Accept  
6) What is a root-endpoint?  
7) What is Api?  
8) what is JSON?  
9) What is Authorization header?  
10) List status codes you are familiar with

**AUTHORIZATION**

Basic Auth: Username and Password

Bearer: Server generates Bearer token when user sing ins and sends it to us in a response. And we use it to make HTTP

Requests to private REST API Endpoints.

Syntax - "Bearer" + " " + token

API Key: Server generate API Key and then provides to user to access private REST API Endpoints.

Bearer vs API Key:

1) Bearer token gets generated when user sign ins. API Key does not neet user to sign in to generate itself.

2) Bearer token can be used only in Headers parameters. API Key can be used in Headers parameters,

in query parameters and inside body parameters

OAuth 1.0

OAuth 2.0 - The service that is used to authenticate users to different websites using their Google,

Facebook, Github, etc accounts. By authenticating using OAuth 2.0 you are providing some information

about yourself to the websites you are trying to authenticate.

Parts of Request:

HTTP Method - GET, POST, DELETE, PUT, PATCH ,

URL (Rest API endpoint) - root-endpoint, resource, path parameters, query parameters

Headers - Authorization header, ...., etc

**Headers**

Content-Type - Specify the data format of the request body

- application/json

- text/plain

- application/xml

Accept - Specify the data format of the response that we want to recieve from the API

- application/json

- text/plain

- application/xml

Query Parameters:

Query Parameters are part of the URL that are used at the end of the URL and they start with

a "?" mark. To filter out data or add additional parameters to the Request that is

not related to the resource.

Ex:

https://gorest.co.in/public-api/users?page=5

API Testing:

API specification review:

GET https://gorest.co.in/public-api/users

Headers

- Authorization - Bearer token

- Accept - JSON.

Path Parameters

- none

Body

- none

POST https://gorest.co.in/public-api/users

Headers

-Authorization - Bearer token

-Content-Type - JSON

-Accept - JSON

Path Parameters

-none

Query Parameters

-none

Body parameters

{

"name": "string", \*Required

"email": "string", \*Required \*Unique

"gender": "string" \*Required

"status": "string" \*Required

}

Positive Test Case:

-Create user

Negative Test Case:

-Create user without Authorization

-Create user without name

-Create user without email

-Create user without gender

-Create user without status

-Create user using the same email

HOMEWORK:

Write documentation for the remaining Rest APIs in gorest and develop test cases

for each of them.