**1** – SOAP vs Restful ?

SOAP, developed by Microsoft, was meant to take the place of binary technologies in the past that were not able to work with the internet the way we know it today. It is an XML-based protocol that allows multiple computers to communicate with each other. It works in many [programming languages](https://www.fiverr.com/categories/programming-tech/online-coding-lessons) with varying degrees of difficulty. It also comes with error reporting that can be automated, which is important to your security, especially when using another company’s web service.

REST is another communication protocol like SOAP, but you don’t have to build the XML structure that’s required every time you perform a task. This makes REST a more streamlined version of SOAP. The most common web services output data in CSV (Command Separated Value), JSON (JavaScript Object Notation), and RSS (Really Simple Syndication). Also, [75% of developers](https://martech.zone/htaccess-wordpress-file/) use REST.

**2** - Difference between acceptance test and functional test ?

Functional tests are tests testing functions of the software, *what it does*. For example, if every menu item in the browser you have open now does intended job.

Acceptance tests are tests against some *criteria of acceptance*. Like some team tests the software on behalf of who ordered it to see if it fulfills contract requirements or some team within company shipping the software tests if it matches the readiness for shipping acceptance criteria.

**3** - What is Mocking ?

Mocking is primarily used in unit testing. An object under test may have dependencies on other (complex) objects. To isolate the behavior of the object you want to replace the other objects by mocks that simulate the behavior of the real objects. This is useful if the real objects are impractical to incorporate into the unit test.

In short, mocking is creating objects that simulate the behavior of real objects.

**4** - What is a reasonable code coverage % for unit tests (and why) ?

it is generally accepted that **80% coverage** is a good goal to aim for. Trying to reach a higher coverage might turn out to be costly, while not necessary producing enough benefit. The first time you run your coverage tool you might find that you have a fairly low percentage of coverage.

Why?

Code coverage basically **show you how much of your code is actually being used by your unit tests**. Running a code coverage report helps show what code is not being used to help you write more unit tests. Code coverage can also show which branches in conditional logic are not being covered

**5** – HTTP/POST vs HTTP/PUT ?

HTTP POST: This header is usually sent, when new data is arrived at server or Api. It's like an insert statement to Api.

HTTP PUT: Consider this as a UPDATE statement to update existing data on server.

**6** - What are the Safe and Unsafe methods of HTTP ?

HTTP methods are safe: **GET , HEAD , or OPTIONS** . All safe methods are also idempotent, but not all idempotent methods are safe. For example, PUT and DELETE are both idempotent but unsafe.

**7** - How does HTTP Basic Authentication work ?

HTTP basic authentication is a simple response mechanism with which a server can request authentication information (a user ID and password) from a client. The client passes the authentication information to the server in an Authorization header. The authentication information is in base-64 encoding.

**8** - Define RestTemplate in Spring ?

RestTemplate. is the central class within the Spring framework for executing synchronous HTTP requests on the client side

**9** – What is idempotant and which HTTP methods are idempotant ?

**A request method is considered "idempotent" if the intended effect on the server of multiple identical requests with that method is the same as the effect for a single such request**. Of the request methods defined by this specification, PUT, DELETE, and safe request methods are idempotent.

Like the definition of safe, the idempotent property only applies to what has been requested by the user; a server is free to log each request separately, retain a revision control history, or implement other non-idempotent side effects for each idempotent request.

**10** – What is DNS Spoofing ? How to prevent ?

Domain Name System (DNS) Spoofing is also known as DNS cache poisoning. **DNS Spoofing** is an attack in which DNS records are altered to redirect users to a fraudulent website that may resemble the user’s intended destination.

DNS spoofing is an attack that tricks your computer redirecting it to a dangerous domain address. One of the most popular tactics is DNS cache poisoning. It happens when a hacker gains control over a site’s DNS server and changes information on it. It is mostly done by altering the DNS records and redirecting traffic to a malicious server which belongs to an attacker. For instance, if someone changed the entry for google.com, any visitor would be redirected to a wrong IP address or, more specifically, to a fake website.

**11** – What is content negotiation ?

Content negotiation is **the process of selecting one of multiple possible representations to return to a client**, based on client or server preferences.

**12** – What is statelessness in RESTful Web Services ?

Statelessness means that every HTTP request happens in complete isolation. When the client makes an HTTP request, it includes all information necessary for the server to fulfill that request. The server never relies on information from previous requests. If that information was important, the client would have sent it again in this request.

**13** - What is CSRF attack? How to prevent ?

A key design principle that protects you from CSRF attacks is **using GET requests for only view or read-only actions**. These types of requests should not transform data and must only display recorded data. This limits the number of requests that are vulnerable to CSRF attacks

**14** - What are the core components of the HTTP request and HTTP response ?

HTTP requests are:

* HTTP Version – Indicates version
* Request Body – Represents message content
* Request Header – Contains metadata, such as cache settings and client type, for the HTTP request message
* URI – Identifies the resource on the server
* Verb – Indicates HTTP methods such as GET, POST, and PUT

HTTP response :

* HTTP Version – Indicates the present version of HTTP
* Response Body – Represents the response message content
* Response Header – Consists of metadata, like content length and server length, for the HTTP response message
* Status/Response Code – Indicates the server status for the requested resource