Ödev - 3

Q1 - SOAP vs RESTful?

A: SOAP is a web service protocol specifying data interchange processes. SOAP uses XML documents for requests and handles XML documents from responses. REST is a lightweight architectural style for APIs to adapt. REST merely loosely relies on standards and that makes it more applicable in many cases. Unlike SOAP, REST can communicate not only in XML but also in JSON, HTML, or text formats.

Q2 - Difference between acceptance test and functional test?

A: Functional test verifies that all the developed functionalities work as intended. During functional testing, qualifications of functions are monitored individually on how they perform in terms of speed, consistency, and accuracy. The acceptance test is a final product test that demonstrates real-life scenarios and verifies that the product meets business criteria. Acceptance is mostly done before releasing the product, unlike functional tests that can be done for each functionality during the development process.

Q3 - What is Mocking?

A: Mocking is a method used in unit testing. While testing behavioral structure and interaction with other complex objects, real objects are not used as real-life objects are unhandy to implement into testing. Objects that resemble real-life kinds are created to simulate more real-life-like scenarios for unit testing. Creating objects that are similar to real-life for testing purposes is called “Mocking”.

Q4 - What is a reasonable code coverage % for unit tests (and why)?

A: Despite the lack of a certain number for test coverage, a lot of programmers agree that unit tests need to cover at least 80% of your code. It is also important to note that number of the tests is not the only subject to consider while testing. Quality of tests, integration of qualities and functions, the significance of tested parts of the code has major roles in quality testing.

Q5 - HTTP/POST vs HTTP/PUT?

A: Post verb is used to create a new entry. The Put verb can be also used to create a new entry but also can perform replacement of or update on existing entries. When used for creating a new resource, Post is mostly preferred when manipulation of URI is to be done by server-side. Otherwise, Put is preferable because it provides complete control over the resource to the client. Put is also defined to be idempotent on the other Post is non-idempotent causing multiple identical requests to the server to create the same resources with different IDs.

Q6 - What are the Safe and Unsafe methods of HTTP?

A: Safe HTTP methods are read-only methods that do not change server state. “GET”, “HEAD”, “OPTIONS”, “TRACE” are safe HTTP methods, and “PUT”, “DELETE”, “POST”, “PATCH” methods are unsafe HTTP methods.

Q7 - How does HTTP Basic Authentication work?

A: HTTP has a built-in simple authentication strategy. It simply requires the request to include basic credentials in the request’s header. The username and password for the request are turned into an encoded base64 string. The process begins with the client sending a request without an authentication string. Then server denies that request with a 401 status code. Then the client responds with a request that has the necessary authentication. If the authentication is correct, the server process the request and returns an OK code.

Q8 - Define RestTemplate in Spring?

A: RestTemplate is a class in Spring that can be used to perform synchronous HTTP requests on the client-side.

Q9 – What is idempotent and which HTTP methods are idempotent?

A: Idempotency ensures that multiple numbers of identical requests will result in the same outcome. That means the server is insistent on identical requests. “GET”, “HEAD”, “OPTIONS”, “TRACE”, “PUT”, “DELETE” are idempotent HTTP methods. Even though multiple “DELETE” methods on the same resource will result in an HTTP 404 response, it still is an idempotent method as the server does not perform the method if it does not exist.

Q10 - What is DNS Spoofing? How to prevent it?

A: DNS Spoofing is a man-in-the-middle type of attack where a malicious person interrupts the DNS server to which IP address for the domain name is requested. After being directed to the malicious IP address, the client’s sensitive and critical information might be exposed to malicious people. It is prevented by using DNS Security Protocol (DNSSEC). It is commonly accepted and required by most industry standards.

Q11 - What is content negotiation?

A: Content negotiation is a system in HTTP that is used to deliver different representations of a URI end-point. It is needed because the different clients may require different types of representations. The content type is requested in the headers of the body of the HTTP request. For example, if a client is expecting URI to return in JSON format, the header can include “application/json”. On the other hand, if it is expecting a text file, the header ought to include “application/text”.

Q12 - What is statelessness in RESTful Web Services?

A: Statelessness is the limitation on which the server does not store the client’s previous states. All the necessary information to process the request is supposed to be provided by the client itself. Any other stored data is not needed to understand the request.

Q13 - What is a CSRF attack? How to prevent it?

A: CSRF is a malicious attack on a website where an HTTP request is modified to perform unintended actions for the user. Attackers introduce themselves to be a trusted end to manipulate the HTTP request. Some authentication techniques can be used to prevent CSRF attacks. Adding identifying information to request such as unique tokens can ensure safety to a certain degree. Some of the techniques are called; synchronizer token pattern, cookie-to-header token, double submit cookie.

Q14 - What are the core components of the HTTP request and HTTP response?

A: An HTTP request require the following components;

1. **Verb** — An HTTP method that indicates the desired action to be performed for a given resource. Some of these methods are; “GET”, “PUT”, “POST”, “DELETE”, “OPTIONS”.
2. **URI** — Uniform Resource Identifier identifies the resource on the server.
3. **HTTP Version** — The version of the HTTP protocol.
4. **Request Header** — Contains additional information for the servers to proceed accordingly.
5. **Request Body** — Contains message content and/or resource representation for some methods like “POST” to process.

An HTTP response message would contain the following;

1. **Response Code** — A number that declares the state of the response.
2. **HTTP Version** — Like an HTTP request, the HTTP version is included in the response.
3. **Response Header** — Like an HTTP request, the header in the response contains additional information on the response such as file type, language, etc.
4. **Response Body** — If the request is successful a response message is returned. Most of the information is given in this body. The data in the body can be an image, text, JSON, etc.