Umut Yıldız

1)

- Rest supports different data formats but Soap supports only XML.
- Rest is faster than SOAP generally and it uses less bandwidth.
- Rest is more flexible.
- SOAP has strict rules.
- SOAP is language, platform and transport independent.

2) **Functional Testing**: This is a test to see if we built a properly functioning product. Is the software up to date and meets the needs of the company? We have test cases for this sort of testing that cover all of the potential scenarios we can figure of, even if they are unlikely to occur "in the real world." We strive for 100% code coverage while performing this sort of testing. We utilize any test environment we can get our hands on at the moment; it doesn't have to be "production" quality, just workable.

Acceptance Testing: This is a test to see whether we built the correct item. Is this actually what the client wants? This is frequently done in collaboration with the client, or by a customer proxy from within the company. We utilize test cases for this form of testing to cover the common circumstances in which we expect the product to be used. This test must be performed in a "production-like" environment, using hardware that is similar to, but not identical to, that which a customer would use. It also related with some non-functional requirement types as reliability, scalability, usability, security, performance, maintability etc.

3) Unit testing is where mocking plays a great role. Other (complicated) objects may be dependent on the object under test. To isolate an object's activity, mocks that replicate the behavior of real objects should be used to substitute the other objects. This is useful if including real items into the unit test is problematic. In a summary, mocking is the process of building things that copy the behaviour of real-world objects.

4) For code coverage, 70-80% is acceptable but it can be less than this value according to project domain. For example, we want to reach 100% but it can be problematic because we can have complex things in our project and to achieve this value, it can be expensive.

5)POST is used for creation generally on the other hand, PUT is used for updating. PUT is idempotent method but POST is not. (Idempotent: if we request several times at same time, method should not have any side effects.)

6) Safe methods: GET, HEAD, OPTIONS (All safe methods are idempotent)

Unsafe methods: Rest of HTTP methods are unsafe as POST, PUT etc.

7) Basic authentication is described in RFC 7617 and it transmits credentials as user id and user password using BASE64. It is not secure because base64 encoding system can be decodable.

Client →(Sends a request with credentials using Base64) → Server

Client ← (200 OK) or (401 Unauthorized)← Server

8) The Rest Template is used to construct RESTful Web Service-consuming apps. For any HTTP methods, we may use the exchange() function to consume web services.

9) If an identical request may be issued once or multiple times in a succession with the same impact and the server remains in the same state, an HTTP method is idempotent. To put it another way, an idempotent technique should have no side effects. GET, OPTIONS, TRACE, PUT, DELETE, HEAD methods are idempotent.

10) DNS spoofing is a type of attack in which tampered DNS records are used to redirect online traffic to a fake website that looks like the real one. This involves implementing DNSSEC or another encryption technology, such as DNS over HTTPS or DNS over TLS, on their DNS servers. Spoofing can also be avoided by using complete end-to-end encryption, such as HTTPS, wherever available. This is where Cloud Access Security Brokers (CASB) come in handy.

11) In HTTP, content negotiation is a mechanism for delivering multiple representations of a URI end-point. It is required because different sorts of representations may be required by different clients. The content type is specified in the headers of the HTTP request body. If a client expects a URI to return in JSON format, for example, the header might add "application/json." If it's expecting a text file, though, the header should include "application/text."

12) A client state should not be kept on the server by a RESTful Web Service. Statelessness is the term for this constraint. The client is responsible for passing its context to the server, which the server can then keep in order to perform the client's subsequent requests.

13) Cross-site request forgery (CSRF or XSRF) attacks are used to deliver malicious requests to a web application from an authorized user. Because the attacker is unable to view the replies to the falsified requests, CSRF attacks concentrate on state changes rather than data theft. CSRF attacks that are successful can have serious implications. Prevention approaches include the synchronizer token pattern, cookie-to-header token, Double Submit Cookie, SameSite cookie attribute, and client-side protections.

14)

Request:

Verb	URI	HTTP Version
Header		
Body		

Response:

Code	HTTP Version
Header	
Body	