HM#6 **Doğukan Bayrak**

1 – What is the difference between manual testing and automated testing?

Manual Testing is done manually by the QA analyst (Human), while Automation Testing is done by a tester using script, code and automation tools (computer). The Manual Testing process is not accurate due to human errors whereas the Automation process is reliable as it is code and script based. Manual Testing is a time-consuming process while Automation Testing is very fast. Manual Testing is possible without programming knowledge whereas Automation Testing is not possible without programming knowledge. Manual Testing allows random Testing while Automation Testing does not allow random Testing.

2 – What does Assert class ?

The assert keyword has been a feature of the Java programming language since Java 1.4. Assertion allows developers to test the assumptions in their programs to troubleshoot and fix bugs. The assert statement is the best way to catch the test phase normal state instead of throwing the rare errors that will not be used very often.

3 - How can be tested 'private' methods ?

First, a public method inside the same class is created. Then that public method contains the private method. Since they are in the same class, the public method can access the private method. In this way, we put the public method to the test. We have indirectly tested the private method.

4 – What is Monolithic Architecture?

Monolithic architecture means that the software is designed as self-contained. We can also say that it is formed as a “single piece” in line with a standard. Components in this architecture are designed as interdependent rather than loosely coupled. Manageability and monitoring are easy. It is easy to develop and maintain for small scale projects. Quick application can be enhanced. The interoperability of the functionalities is consistent. Transaction management is easy.

5 - What are the best practices to write a Unit Test Case ?

We should aim for %100 code coverage. Our unit tests should be maintainable and readable. If we change the project , we should change the unit tests too. For the good code readability, we should well naming the convenions. Alson unit test need to be trustworthy. Also our test case should focus single case a time. Our unit tests need to isolated from other units and dependencies. Lastly, units tests should be automated.

6 - Why does JUnit only report the first failure in a single test ?

Reporting multiple failures on a single test is often a sign that the test is doing too much compared to what a unit test should do and it is too big a unit test.While it's okay to report multiple bugs for functional,acceptance tests, if it is a unit test, then it is too big a unit test. JUnit is designed to work best with several small tests. It executes each test within a separate instance of the test class. It reports failure on each test.

7 - What are the benefits and drawbacks of Microservices ?

Adding new features is easy regardless of the size of the app. Usually, it is sufficient to make changes within the relevant service. Each service is independent of each other. Therefore, the code base is simple. Newcomers to the team can adapt practically without getting lost in the code base. Versioning is easy. Different frameworks can be written for services. Each service can have its own database.

Because each service is independent of each other, it contains more than one service and more than one database. This makes the transaction difficult. Having many services makes it difficult to manage and monitor services.

8 - **What is the role of actuator in spring boot ?**

By adding the spring boot actuator as a dependency to our project, we can easily access information such as the working status of our project (standing / not standing), traffic status, last hundred http requests, the status of our database, etc., through endpoints.

9 - What are the challenges that one has to face while using Microservices ?

Design: Compared to monolithic apps, organizations face increased complexity when designing microservices.

- Each microservice's size
- Optimal boundaries and connection points between each microservice
- The framework to integrate services

Security: Microservices are often deployed across multi-cloud environments, resulting in increased risk and loss of control and visibility of application components—resulting in additional vulnerable points. Compounding the challenge, each microservice communicates with others via various infrastructure layers, making it even harder to test for these vulnerabilities.

Testing: The testing phase of any software development lifecycle (SDLC) is increasingly complex for microservices-based applications. Given the standalone nature of each microservice, we must test individual services independently. Exacerbating this complexity, development teams also must factor in integrating services and their interdependencies in test plans.

Communication: Independently deployed microservices act as miniature standalone applications that communicate with each other. To achieve this, you have to configure infrastructure layers that enable resource sharing across services. A poor configuration may lead to:

- Increased latency
- Reduced speed of calls across different services

In this situation, we've got a non-optimized application with a slow response time.

10 - How independent microservices communicate with each other?

One of the biggest challenge when moving to microservices-based application is changing the communication mechanism. Because microservices are distributed and microservices communicate with each other by inter-service communication on network level. Each microservice has its own instance and process. Therefore, services must interact using an inter-service communication protocols like HTTP, gRPC or message brokers AMQP protocol.

11 - What do you mean by Domain driven design ?

Domain Driven Design is the creation of real-world business models with a common language (Ubiquitous Language) that everyone can understand. And it specifies how software should be modeled in order to adapt it to the digital world.

12 – What is container in Microservices ?

Containers are a form of operating system virtualization. A single container might be used to run anything from a small microservice or software process to a larger application. Inside a container are all the necessary executables, binary code, libraries, and configuration files.

13 - What are the main components of Microservices architecture ?

Clients, API Gateway, Communication between services (Rest, Message Bus), Data Storage and Database Structure, Service Management, Service Discovery, Logging and Monitoring, Devops.

14 - How does a Microservice architecture work?

The microservice architecture contains components depending on the business requirements.

- API Gateway- Clients need API Gateway as it is an entry point, which forwards the call to the specific services on the back end. Here API gateway helps in collecting the responses from different services and returns the response to the client.
- Microservices- As the name itself suggests that microservices are the services that help in dividing the service into small services that perform a certain business capability like user registration, current orders, or wish list.
- Database- Microservices can either share the same database or an independent database.
- Inter-microservices communication- REST or Messaging are the protocol to interact with each other.