HW#6

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

Manuel Test	Automation Test
Using manual testing, it can be difficult to test the application on different operating systems.	With the help of automation testing, we can easily test the application on different Operating systems.
Test cases run manually.	Test scenarios are executed with the help of tools.
It less costly.	It is more expensive.
It wastes time for some test cases.	It takes less time to run cases as it is a machine.
Humans can make mistakes therefore less accuracy.	The machine makes almost no mistakes .
It is useful to check the ease of access to the application, as it involves human intervention.	Contains tools that cannot control availability or accessibility.
It can be difficult to manual test in different browsers.	Automation provides the advantage of testing software in different browsers.
In manuel test, you need to sit in front of your system and run test cases as it involves human intervention.	Just run the automation scripts and you can have them running overnight.
In manuel test, you need write reports on your own.	The tool will generate the test case execution reports . TestNG is the framework that will generate reports for you.

2 – What does Assert class ?

Assert class provides a set of assertion methods useful for writing tests. Only failed assertions are recorded. These methods can be used directly: *Assert.assertEquals(...),* however, they read better if they are referenced through static import:

```
import static org.junit.Assert.*;
  ...
  assertEquals(...);
```

Aleyna ZENGİN

3 - How can be tested 'private' methods ?

1. Just call it

The simplest and most straightforward way to call a private method is, you know, call it.
To “just call it” in Java we need to change visibility of the method.
The first way is to make method *package private* (no access modifier) and put tests into the same package. This is a fairly common practice.
The second way is to make method public. To let people know you still don’t want to call this method you can use *@VisibleForTesting* annotation

2. Nested class

You can also put a test class inside a tested one but that doesn’t look great. You have to use the same file for both classes, and your production binaries will actually contain test code.

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. The components in this architecture are designed as interdependent rather than loosely coupled.”

Advantages

- Manageability and monitoring are easy.
- It is easy to develop and maintain for small scale projects. Application can be developed quickly.
- The interoperability of the functionalities is consistent.
- Transaction management is easy.

Disadvantages

- As the application grows, it becomes more difficult to develop new features and maintain existing code.
- With the increase in the number of teams and employees working on the project, development and maintenance becomes more difficult.
- Because of their interdependencies, a change in one functionality can affect other places.
- Version management becomes difficult.
- The same programming language and the same frameworks must be used in the application.

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

1. Unit tests should be maintainable and readable
2. Name your tests well.
3. Write reliable and trustworthy unit tests.
4. Make automated unit testing a rule.

5. Automate tests using CI/CD tools.
6. Focus on single use-case at a time.
7. Aim for 100% code coverage.
8. Avoid test interdependencies.
9. Write tests during development, not after it.
10. Update the tests periodically.

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

JUnit is usually designed in a way that it deals with smaller tests and is capable of running each assessment with a boundary of separate analysis. Due to this, it reports only the first failure on each test case attempt**.**

7 - What are the benefits and drawbacks of Microservices ?

Benefits

1. Whether the application is very large or very small, adding new features and maintaining existing code is easy. Because it is sufficient to make changes only within the relevant service.
2. Since each service is independent of each other and has only its own business logic, the code base of the service will be quite simple.
3. Teams can work more efficiently and quickly. Friends who are new to the project can easily adapt without getting lost in the code base.
4. Services can be scaled independently of each other.
5. Versioning is easy.
6. A change to a service does not require other services to be deployed. It is sufficient to deploy only the relevant service.
7. Services can be written in different languages and with different frameworks. Each service can have its own different database.

Drawbacks

1. Transaction management will be difficult because there are multiple services and multiple databases.
2. The manageability and monitoring of these services will become more difficult.
3. If services become more independent than necessary, it will be difficult to manage.

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.

Typing parameters such as *actuator/beans, actuator/health* at the end of our Spring Boot application link, we can reach the endpoints and get information.
For example, we can check whether our application is standing by typing /*actuator/health*.
While our application is standing, a json data is returned:

```
{
    "status" : "UP"
}
```

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

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.

Designing a distributed architecture
It can be difficult to break out of the usual software mold. An architecture with a monolitic structure must be distributed and each domain must be determined very accurately. Each service should be designed for what it will be customized for, and its interactions with other services should be considered.

Deployment
If you intend to pass the Microservice architecture, manual deployment will be very difficult for you. Since your application, which was once a single application, has been divided into many different services and their number has increased, you should create a CI / CD area suitable for you and make automatic deployment.

Monitoring and Logging
Monitoring infrastructure should be established to monitor the status of each service. Logging is one of the most critical points for software developers.

Integration Testing
Similar to debugging, the testing phase can be more difficult than it used to be, as many services need each other for testing.

Configuration Management
There are many different services and if we consider that these services are dev, test and prod environments, it will be necessary to manage service configurations centrally.

Communication
It is necessary for the services to communicate with each other, with the API Gateway, or with the mobile or web applications to determine and manage the communication methods correctly.

10 - How independent microservices communicate with each other?

There are two basic methods, synchronous and asynchronous communication, which are accepted and widely used for communication from microservice to microservice. All communication methods are built on these two structures. These two methods are often used together in Microservice architectures. This is called *hybrid architecture*.

Synchronous Communication
When a microservice is performing its operations in synchronous communication,it needs information from a different service, it sends a request to that service and waits for a response. As a result of the response, it continues its operations.

HTTP is the most suitable protocol to implement synchronous communication between microservices. It can be implemented with HTTP, REST or SOAP. Communication with Restful over HTTP protocol is the most widely used for synchronous transactions.

Asynchronous Communication
In asynchronous transactions, the requesting service does not expect a response from the service it receives. Thus, the requesting service does not become dependent on the service it receives, and this is exactly what we want in the microservice architecture (Loosely coupled services). The service continues to run smoothly even when the asynchronous request fails.

Asynchronous communication also allows one-to-many communication, when a message is sent, it can be provided to reach more than one service. In synchronous communication, the client has to make a request to each different service one by one.

11 - What do you mean by Domain driven design ?

“Domain Driven Design (DDD) is a software development approach that deeply connects the ever-changing fundamental business rules in a world of complex requirements. “

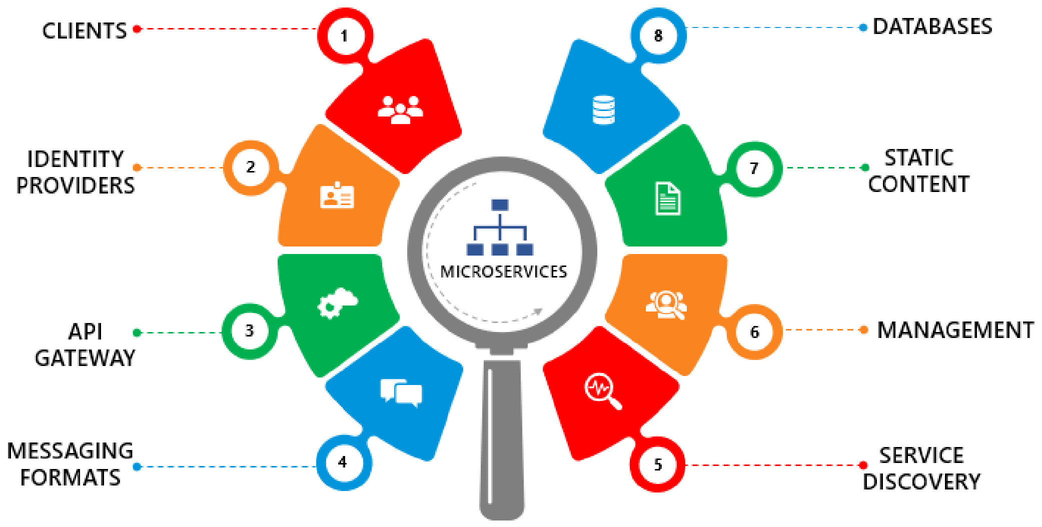
The main purpose of DDD is; is to produce easily scalable, flexible and highly available software that enables rapid adaptation to rapidly changing requirements. In DDD, the business rules included in the application are logically distributed to the domains. Responsibly, the most related units are kept on the same domain, while other units are moved to different domains.

12 – What is container in Microservices ?

Containers are a lightweight alternative to VMs for providing isolated operating environments for your workloads.
Microservices are better suited to containers, which offer fine-grained control over resource allocation and more efficient use of your computing capacity.

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 ?



14 - How does a Microservice architecture work?

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.