HOMEWORK-6 gulnisaslan@gmail.com

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

Manual testing, test case manually (by human or other methods) without supported from tools or scripts. Automated Testing manual testing is the opposite.

Manual testing and Automated testing two vast area: There are black box testing, white box testing, integration testing , system, performance and load testing. Some of that methods suitable better manual testing, In some best performed though automation.

MANUAL TESTING:

-In manuel testing is not true all time human errors

-It is practical

-It takes time and needs human resource.

-Manual testing does in human obversation

-Manuel testing is suitable User-Friendliness and customer experience

AUTOMATED TESTING:

-Automated testing is most trusted but It performed tools and/or scripts.

-Automated testing is than most quickly manual testing.

-Automated testing really is practical when the test cases repeatedly over a long time.

-Automated testing is not obversation human and can not guarantee customer experience and user friendliness.

2 – What does Assert class ?

Assert Class is in Junit library a class. We write useful test cases with Assert Class.

Only failed test cases is recored.

3 - How can be tested 'private' methods ?

For Private method testing, we use import java.lang.reflect.method library. There is powermock library. Private method tests most quickly with whiteBox.Invoke method in powermock library.

4 – What is Monolithic Architecture ?

Monolithic applications are designed to handle multiple related tasks. They're typically complex applications that encompass several tightly coupled functions.

For example, consider a monolithic ecommerce SaaS application. It might contain a web server, a load balancer, a catalog service that services up product images, an ordering system, a payment function, and a shipping component.

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

We've covered a lot of ground by talking about the fundamentals of unit testing. After learning the basics of unit testing, you're now ready for the main part of the post, in which we'll walk you through nine best practices you can use to get the most out of your unit testing.

1-Tests Should Be Fast

2. Tests Should Be Simple

3. Test Shouldn't Duplicate Implementation Logic

4. Tests Should Be Readable

5. Tests Should Be Deterministic

6. Make Sure They're Part of the Build Process

7. Distinguish Between The Many Types of Test Doubles and Use Them Appropriately

8. Adopt a Sound Naming Convention for Your Tests

9. Don't Couple Your Tests With Implementation Details

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

Reporting multiple failures in a single test is generally a sign that the test does too much and it is too big a unit test. JUnit is designed to work best with a number of 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 ?

Microservices Benefits:

1- Multilingual architecture
2- Easy scaling
3- Better team management
4- It's easier to innovate.
5- Microservices have their own database
6- more focused
7- They do not affect other services while being implemented.

Microservice drawbacks:

1-It is not easy implementation.

2-It is not easy debugging.

3-It is not easy fault debugging

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

Actuator is mainly used to expose operational information about the running **application** — health, metrics, info, dump, env, etc. It uses HTTP endpoints or JMX beans to enable us to interact with it.

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

A microservices approach means designing and developing an application as a group of loosely coupled services that communicate among themselves to achieve a stated business objective.

A rising pattern of wider adoption rate confirms that there are several use cases in which businesses benefit from using microservices to develop and deploy applications. After all, microservices are often considered the first step to embracing a DevOps culture, which enables:

- Automation
- Improved scalability
- Manageability
- Agility
- Faster delivery & deployment

For the same reason, a microservice architecture is usually the first choice for businesses looking to rewrite legacy applications to use modern programming languages and technology stack

10 - How independent microservices communicate with each other?

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 ?

The word Domain used in context of software development refers to *business*. In the process of application development, term domain logic or business logic is commonly used. Basically, business logic is area of knowledge around which application logic revolves. The business logic of an application is a set of rules and guidelines that explain how business object should interact with each other to process modeled data.

12 – What is container in Microservices ?

Microservice business logic is stored in a Container. (If you want this Container, you can think of it as Docker) It gives outward Services and uses some services. It contains Config, Policy(Rules), Security(Security) and Data.

13 - What are the main components of Microservices architecture ?

1. Clients
2. Identity Providers
3. API Gateway
4. Messaging Formats
5. Databases
6. Static Content
7. Management
8. Service Discovery

14 - How does a Microservice architecture work?

Microservices architecture focuses on classifying the otherwise large, bulky applications. Each microservice is designed to address an application's particular aspect and function, such as logging, data search, and more. Multiple such microservices come together to form one efficient application.

This intuitive, functional division of an application offers several benefits. The client can use the user interface to generate requests. At the same time, one or more microservices are commissioned through the API gateway to perform the requested task. As a result, even larger complex problems that require a combination of microservices can be solved relatively easily.