Microservices Theory Assignment-1

1) What is microservices?

Microservice is a distinctive method of developing software systems that tries to focus on building single-function modules with well-defined interfaces and operations. The trend has grown popular in recent years as Enterprises look to become more **Agile** and move towards a DevOps and continuous testing.

Microservices are a way of breaking large software projects into loosely coupled modules, which communicate with each other through simple Application Programming Interfaces (**APIs**).

Following are some rules that we need to keep in mind while developing a Microservice-oriented application.

- **Independent** Each microservice should be independently deployable.
- **Coupling**: All microservices should be loosely coupled with one another such that changes in one will not affect the other.
- **Business Goal**: Each service unit of the entire application should be the smallest and capable of delivering one specific business goal.

2) Challenges with monolithic oriented architecture

- Application is too large and complex to fully understand and made changes fast and correctly.
- The size of the application can slow down the start-up time.
- You must redeploy the entire application on each update.
- Impact of a change is usually not very well understood which leads to do extensive manual testing.
- Continuous deployment is difficult.
- Monolithic applications can also be difficult to scale when different modules have conflicting resource requirements.
- Another problem with monolithic applications is reliability. Bug in any module (e.g. memory leak) can potentially bring down the entire process. Moreover, since all instances of the application are identical, that bug will impact the availability of the entire application.

3) Advantages and disadvantages of microservices

Advantages of microservices

- **Improved fault isolation**: Larger applications can remain mostly unaffected by the failure of a single module.
- **Ease of understanding:** With added simplicity, developers can better understand the functionality of a service.
- **Smaller and faster deployments**: Smaller codebases and scope = quicker deployments, which also allow you to start to explore the benefits of Continuous Deployment.
- **Scalability**: Since your services are separate, you can more easily scale the most needed ones at the appropriate times, as opposed to the whole application. When done correctly, this can impact cost savings.
- **Eliminate vendor or technology lock-in**: Microservices provide the flexibility to try out a new technology stack on an individual service as needed. There won't be as many dependency concerns and rolling back changes becomes much easier. With less code in play, there is more flexibility.

Disadvantages of microservices:

- **Communication between services is complex**: Since everything is now an independent service, you have to carefully handle requests traveling between your modules. In one such scenario, developers may be forced to write extra code to avoid disruption. Over time, complications will arise when remote calls experience latency.
- **Global testing is difficult**: Testing a microservices-based application can be cumbersome. In a monolithic approach, we would just need to launch our WAR on an application server and ensure its connectivity with the underlying database. With microservices, each dependent service needs to be confirmed before testing can occur.
- **Debugging problems can be harder**: Each service has its own set of logs to go through. Log, logs, and more logs.
- **Deployment challengers**: The product may need coordination among multiple services, which may not be as straightforward as deploying a WAR in a container.
- **Large vs small product companies**: Microservices are great for large companies, but can be slower to implement and too complicated for small companies who need to create and iterate quickly, and don't want to get bogged down in complex orchestration.