Part 4

A problem with services is that a service can be down. If the StockService is down, the ProductService cannot work properly. Describe how you would solve this problem?

Circuit Breaker Pattern

If there are failures in your microservices ecosystem, then you need to fail fast by opening the circuit. This ensures that no additional calls are made to the failing service, once the circuit breaker is open. So, we return an exception immediately. This pattern also monitors the system for failures and once things are back to normal, the circuit is closed to allow normal functionality.

This is a very common pattern to avoid cascading failure in microservice ecosystem.

It is possible use some popular third-party libraries to implement circuit breaking in an application, such as Polly and Hystrix.

Retry Design Pattern

This pattern states that you can retry a connection automatically which has failed earlier due to an exception. This is very handy in case of temporary issues with one of your services. A lot of times a simple retry might fix the issue. The load balancer might point you to a different healthy server on the retry, and your call might be a success.

Timeout Design Pattern

This pattern states that you should not wait for a service response for an indefinite amount of time throw an exception instead of waiting too long. This will ensure that you are not stuck in a state of limbo, continuing to consume application resources. Once the timeout period is met, the thread is freed up.