166. Why is testing microservices more challenging than monoliths?

Due to distribution, network dependencies, multiple data sources, and complex inter-service communication.

167. What types of testing are needed in microservices?

- Unit Testing
- Integration Testing
- Contract Testing
- End-to-End Testing
- Performance Testing

168. What is Contract Testing?

It verifies that consumer and provider services agree on the interface/contract without needing both to run at the same time.

169. Tools for Contract Testing?

- Pact
- Spring Cloud Contract
- Postman Mock Servers

170. What is the benefit of contract testing?

Ensures interface compatibility, catches issues early, and supports independent deployment of services.

171. What is the difference between unit and integration tests?

Unit tests focus on a single class/method with mocks. Integration tests validate multiple components with real dependencies.

172. What are test doubles?

- Mock: Controlled behavior
- Stub: Predefined response
- Fake: In-memory logic
- Spy: Tracks method calls

173. What is Testcontainers?

A Java library to launch real Docker containers during integration tests. Useful for databases, Kafka, etc.

174. Why use Testcontainers in microservices?

Provides a real test environment with production-like services. Enhances reliability of integration tests.

175. Best practices for testing microservices?

- Use all test layers
- Automate in CI/CD
- Isolate failures with mocks/stubs
- Use contract testing
- Prefer Testcontainers or in-memory DBs