A company built a monolithic application. Every time they change a small UI feature, they must redeploy the entire app. What is the main problem here?

A. Deployment complexity  
B. Memory leak issue  
C. Database failure  
D. Lack of frontend developers

**Answer: A**  
**Explanation:** In monolithic architecture, all modules are tightly coupled, so even small changes require full redeployment.

**✅ 2.**

Your team wants to scale only the “Search” feature because of high traffic, not the entire application. Which architecture supports this easily?

A. Monolithic  
B. Microservices  
C. Desktop application  
D. Monolith with multiple threads

**Answer: B**  
**Explanation:** Microservices allow independent scaling of specific services.

**✅ 3.**

In a monolithic system, a bug in the “Payment” module crashes the entire application. Why?

A. Database corruption  
B. Shared memory usage  
C. Tightly coupled modules (single codebase)  
D. Wrong server configuration

**Answer: C**  
**Explanation:** In monoliths, all modules are in the same process, so one failure can crash everything.

**✅ 4.**

A company using microservices notices high network latency. What is the likely cause?

A. All modules are in the same process  
B. Inter-service communication over network  
C. Database query is slow  
D. Memory issue in one module

**Answer: B**  
**Explanation:** Microservices rely on API calls over the network, which adds latency.

**✅ 5.**

Different teams want to build different modules using Java, Node.js, and Python. Which architecture supports this flexibility?

A. Monolithic  
B. Microservices  
C. Client-Server  
D. Layered architecture only

**Answer: B**  
**Explanation:** Microservices enable polyglot architecture (different languages per service).

**✅ 6.**

Your monolithic application is huge (500k+ lines of code) and new developers take weeks to understand it. This is mainly due to:

A. Lack of comments  
B. No CI/CD pipeline  
C. Low modularity / high complexity  
D. Too many tests

**Answer: C**  
**Explanation:** Large monoliths become hard to maintain, understand, and onboard.

**✅ 7.**

In microservices, each service has its own database. What advantage does this give?

A. Less memory usage  
B. Better performance always  
C. Loose coupling and independent data models  
D. Easy joins between services

**Answer: C**  
**Explanation:** Independent databases prevent tight coupling, allow domain-specific schemas, and autonomous services.

**✅ 8.**

Your microservices system faces an issue: when one service fails, the others also crash. What design pattern is missing?

A. Singleton  
B. Observer  
C. Circuit Breaker  
D. Factory

**Answer: C**  
**Explanation:** Circuit Breaker prevents cascading failures across services in microservice architecture.

**✅ 9.**

A startup with a small team wants to launch fast, with minimal infrastructure and simple deployment. Which architecture is better initially?

A. Microservices  
B. Monolithic  
C. Serverless  
D. Peer-to-Peer

**Answer: B**  
**Explanation:** Monolith is faster to build, deploy, and manage for small teams and early-stage products.

**✅ 10.**

During testing, the microservices application becomes complex because multiple services need to run and coordinate. Why is testing harder?

A. No unit tests  
B. Distributed services + network dependencies  
C. Not enough developers  
D. Poor documentation

**Answer: B**  
**Explanation:** Microservices introduce distributed complexity, making integration testing and environment setup harder.