1️⃣ Microservices – Basics & Fundamentals (10 Questions)

1. What is Microservices architecture?
2. Difference between Monolithic, SOA, and Microservices.
3. What are the core characteristics of microservices?
4. What are the advantages and disadvantages of microservices?
5. When should you NOT use microservices?
6. What is service autonomy?
7. What is bounded context?
8. Difference between synchronous and asynchronous communication.
9. What is RESTful communication in microservices?
10. How are microservices different from modular monoliths?

2️⃣ Algorithm Solving & DSA (Distributed-System Logic) (10 Questions)

These test **thinking & system logic**, not just coding.

1. Design a rate-limiting algorithm for APIs.
2. Implement retry with exponential backoff logic.
3. How do you ensure idempotency in REST APIs?
4. Design a distributed unique ID generator.
5. Implement circuit breaker logic conceptually.
6. How do you handle data consistency across services?
7. Design a service to handle high-traffic flash sales.
8. How would you shard a database?
9. Design a cache eviction strategy.
10. Explain eventual consistency with an example.

3️⃣ Microservices – Intermediate (Frequently Asked Interview Questions) (10 Questions)

1. How do services communicate with each other?
2. What is service discovery and why is it needed?
3. Difference between client-side and server-side discovery.
4. What is API Gateway?
5. How do you handle configuration management?

6. How do you manage inter-service authentication?
7. What is distributed tracing?
8. Difference between orchestration and choreography.
9. How do you handle failures in microservices?
10. What is the CAP theorem?

4️⃣ Microservices – Advanced (Architecture & Ecosystem) (10 Questions)

1. Explain Saga pattern in detail.
2. Difference between Saga and 2-Phase Commit.
3. What is Event-Driven Architecture?
4. How does Kafka fit into microservices?
5. What is CQRS and when should it be used?
6. Explain service mesh (Istio/Linkerd).
7. What is sidecar pattern?
8. How does Kubernetes support microservices?
9. What is eventual consistency vs strong consistency?
10. How do you design highly scalable microservices?

5️⃣ Microservices Communication Patterns (10 Questions)

1. REST vs gRPC – when to use which?
2. Synchronous vs asynchronous communication.
3. What is message broker?
4. How does Kafka differ from RabbitMQ?
5. What is request-reply pattern?
6. What is publish-subscribe pattern?
7. How do you ensure message ordering?
8. How do you handle duplicate messages?
9. What is dead letter queue (DLQ)?
10. What is back-pressure?

6️⃣ Microservices – Data Management (10 Questions)

1. Database per service vs shared database.
2. How do you handle joins across microservices?
3. What is polyglot persistence?
4. How do you handle schema evolution?
5. How do you handle data migration?
6. How do you ensure data consistency?
7. What is eventual consistency?
8. How do you manage read/write separation?
9. How do you implement caching?
10. How do you handle large transactions?

7️⃣ Design Patterns in Microservices (10 Questions)

1. What design patterns are used in microservices?
2. API Gateway pattern.
3. Circuit Breaker pattern.
4. Saga pattern.
5. Bulkhead pattern.
6. Strangler Fig pattern.
7. Sidecar pattern.
8. Database-per-service pattern.
9. Event Sourcing pattern.
10. CQRS pattern.

8️⃣ Microservices Security (10 Questions)

1. How do you secure microservices?
2. What is OAuth2 in microservices?
3. How does JWT work?
4. How do you secure service-to-service communication?
5. What is mTLS?
6. How do you handle secrets?
7. How do you prevent API abuse?

8. What is zero-trust architecture?
9. How do you handle authorization?
10. How do you manage compliance?

9️⃣ Microservices – Capstone Project Interview Questions

A. Architecture & Design (10 Questions)

1. Explain your microservices architecture.
2. Why did you choose microservices over monolith?
3. How did you decide service boundaries?
4. How did you handle scalability?
5. How did you design fault tolerance?
6. How did you handle data consistency?
7. How did you handle service communication?
8. How did you handle configuration management?
9. How did you handle backward compatibility?
10. What would you redesign today?

B. Implementation & Coding (10 Questions)

1. Explain a complex microservice you built.
2. How did you handle distributed transactions?
3. How did you handle retries and failures?
4. How did you implement circuit breaker?
5. How did you handle concurrency?
6. How did you ensure idempotency?
7. How did you handle API versioning?
8. How did you handle schema changes?
9. How did you implement caching?
10. Explain a production issue you resolved.

C. Production, DevOps & Observability (10 Questions)

1. How were your microservices deployed?

2. How did you use Docker?
3. How did Kubernetes help?
4. What monitoring tools did you use?
5. How did you implement logging?
6. How did you use distributed tracing?
7. How did you handle production incidents?
8. How did you scale services?
9. How did you manage CI/CD?
10. How did you ensure zero-downtime deployment?