1️⃣ Java REST API – Basics & Fundamentals (10 Questions)

1. What is a REST API?
2. What are REST principles and constraints?
3. Difference between REST and SOAP.
4. What are HTTP methods and their use cases?
5. What is a resource in REST?
6. What are HTTP status codes? Explain common ones.
7. What is statelessness in REST?
8. Difference between PUT and PATCH.
9. What is idempotency?
10. What is content negotiation?

2️⃣ Java REST API – Algorithm Solving & API Logic (10 Questions)

These are **logic-based backend coding questions** commonly asked.

1. Design an API for pagination.
2. Design an API for sorting and filtering.
3. Implement rate limiting for an API.
4. Handle duplicate request submission (idempotent API).
5. Design an API for file upload/download.
6. Implement search API with multiple parameters.
7. Design API versioning strategy.
8. Handle partial updates using PATCH.
9. Design API response for large datasets.
10. Handle concurrency in REST APIs.

3️⃣ Java REST API – Intermediate (Frequently Asked Interview Questions) (10 Questions)

1. How does Spring Boot simplify REST API development?
2. Difference between @Controller and @RestController.
3. What is @RequestBody vs @RequestParam vs @PathVariable?
4. How does Spring handle JSON serialization/deserialization?
5. What is ResponseEntity and why is it used?

6. How do you handle exceptions globally?
7. What is validation and how is it implemented?
8. What is HATEOAS?
9. How do you handle CORS in REST APIs?
10. How do you handle API documentation?

---

4 Java REST API – Advanced (Architecture & Ecosystem) (10 Questions)

1. How do you design scalable REST APIs?
2. How do you handle backward compatibility?
3. What is API Gateway and why is it needed?
4. How do you secure REST APIs?
5. How does OAuth2 work in REST services?
6. What is JWT and how is it used?
7. How do you implement caching in REST APIs?
8. How do you handle distributed transactions?
9. REST vs gRPC – when to choose what?
10. How do you design APIs for microservices?

---

5 Java REST API – Performance & Optimization (10 Questions)

1. How do you improve REST API performance?
2. How does HTTP caching work?
3. What is ETag and how is it used?
4. How do you handle large payloads?
5. How do you implement async REST APIs?
6. How do you handle connection pooling?
7. How do you optimize database access?
8. How do you handle N+1 query problem?
9. How do you handle API timeouts?
10. How do you handle bulk operations?

---

6 Java REST API – Security (10 Questions)

1. What are common REST API security threats?
2. How do you implement authentication?
3. Difference between authentication and authorization.
4. How do you implement role-based access?
5. How do you secure sensitive data?
6. How do you protect APIs from brute-force attacks?
7. What is CSRF and how do you handle it?
8. How do you secure inter-service communication?
9. How do you manage secrets?
10. What is zero-trust security?

7️⃣ Design Patterns in Java REST APIs (10 Questions)

1. Which design patterns are commonly used in REST APIs?
2. DTO pattern and why it's needed.
3. Facade pattern in REST services.
4. Builder pattern for response objects.
5. Singleton pattern in Spring beans.
6. Strategy pattern for validation or processing.
7. Circuit Breaker pattern.
8. API Gateway pattern.
9. Saga pattern.
10. CQRS pattern.

8️⃣ Java REST API – Capstone Project Interview Questions

A. Architecture & Design (10 Questions)

1. Explain your REST API architecture.
2. Why did you choose REST over other approaches?
3. How did you design resource URLs?
4. How did you handle API versioning?
5. How did you handle scalability?
6. How did you ensure loose coupling?

7. How did you handle backward compatibility?
8. How did you handle API documentation?
9. How did you handle monitoring?
10. What would you redesign today?

B. Implementation & Coding (10 Questions)

1. Explain a complex API you implemented.
2. How did you handle validations?
3. How did you handle exceptions?
4. How did you implement security?
5. How did you handle pagination and filtering?
6. How did you handle file uploads?
7. How did you implement caching?
8. How did you handle concurrency?
9. How did you test REST APIs?
10. Explain a production issue you fixed.

C. Production, DevOps & Observability (10 Questions)

1. How were your REST APIs deployed?
2. How did you configure environments?
3. How did you monitor APIs?
4. What logging strategy did you use?
5. How did you handle production incidents?
6. How did you ensure high availability?
7. How did you manage CI/CD?
8. How did you handle zero-downtime deployment?
9. How did you handle rollback?
10. How did you scale APIs under load?