GLOBAL LOGIC

What to Expect in Round 2 (Interview Pattern)

Typically, it has **2 stages**:

1. **Technical Interview** – Focus on Core Java, SQL, DSA, and project-related discussions.
2. **Managerial/HR Interview** – Communication, problem-solving mindset, project ownership, and soft skills.

Technical Interview Focus Areas

1. Core Java (Very Important)

Expect **deep conceptual + practical** questions. Common areas:

- **OOPs**: Inheritance, polymorphism, abstraction, encapsulation (real-world examples).
- **Exceptions**: Checked vs unchecked, custom exceptions, try-with-resources.
- **Collections Framework**: List, Set, Map, HashMap vs ConcurrentHashMap, fail-fast vs fail-safe.
- **Threads & Concurrency**: Thread lifecycle, synchronization, deadlock, `ExecutorService`.
- **Java 8 Features**: Streams, lambda expressions, functional interfaces, Optional.

Example Questions:

- Explain how `HashMap` works internally.
- What's the difference between `==` and `.equals()`?
- How do you handle concurrency in Java? Give examples.
- Write a Java program to detect a cycle in a linked list.

2. SQL / PostgreSQL

Since Section-1 was tricky, they'll likely probe more:

- Joins (INNER, LEFT, RIGHT, FULL).
- Subqueries & CTE (WITH clause).
- Aggregations (`GROUP BY`, `HAVING`).
- Window functions (`ROW_NUMBER`, `RANK`).

Example Questions:

- Find the **second highest salary** from an Employee table.
- Query to list departments with **more than 5 employees**.
- Difference between **`EXISTS` vs `IN`**.
- Write a query to get the **Nth highest salary** without using `TOP` or `LIMIT`.

3. Coding / DSA

They'll give **1–2 problems**, generally medium-level:

- **Arrays & Strings**: Sliding window, prefix sums, two pointers.
- **Hashing**: Frequency-based problems (anagrams, subarrays with given sum).
- **Dynamic Programming**: Classic problems (climbing stairs, coin change, subsequences).

Example Questions:

- Write a program to check if two strings are **anagrams**.
- Find the **longest substring without repeating characters**.
- Given an array, find the **maximum subarray sum** (Kadane's Algorithm).
- Implement **LRU Cache** using LinkedHashMap.

4. Full Stack / Practical Knowledge

Since it's a Full Stack track:

- Basics of **Spring Boot** (REST APIs, dependency injection, annotations).
- Frontend basics (HTML/CSS/JavaScript, React/Angular if mentioned in CV).
- REST API concepts: `GET vs POST`, status codes, request/response lifecycle.

Example Questions:

- How do you connect a Java backend to a PostgreSQL database?
- Explain `@RestController` vs `@Controller` in Spring Boot.
- What is dependency injection? Why is it useful?
- Explain REST API status codes (200, 201, 400, 404, 500).

Managerial / HR Round

This round checks **soft skills + communication + teamwork**.

- **Email Writing** type scenarios may be extended into **behavioral questions**.
- Time management, handling pressure, client interactions.

Example Questions:

- Tell me about a time you faced a **deadline issue**. How did you handle it?
- How would you explain a technical concept to a **non-technical client**?
- What would you do if you're assigned a **technology you haven't worked with** before?
- Why do you want to join **GlobalLogic**?

Preparation Strategy

1. **Revise Core Java** (Collections, OOP, Threads, Java 8).
2. **Practice SQL queries** (medium-level with joins, subqueries, aggregates).
3. **Solve 10–15 coding problems** (LeetCode Easy/Medium).
4. **Revise Full Stack basics** (Spring Boot REST APIs + database connection).
5. **Mock HR answers** (introduce yourself, strengths, weaknesses, teamwork).

In short:

- **Round 2 = Practical + Applied Knowledge.**
- Focus on **Java + SQL + DSA** for Technical.
- Be ready with **projects + communication** for HR/Managerial.

Round 2 for GlobalLogic Java Full Stack:

- **Section A: Core Java Must-Practice Questions**
- **Section B: SQL/PostgreSQL Questions**
- **Section C: Coding/DSA Problems**
- **Section D: Spring Boot & Full Stack Questions**
- **Section E: HR/Managerial Questions**

Section A: Core Java Questions

OOPs & Basics

1. Explain **polymorphism** with examples (compile-time vs runtime).
2. Difference between **abstract class** and **interface** in Java.
3. How does Java achieve **platform independence**?
4. Difference between `==` and `.equals()` in Java.
5. What is a **marker interface**? Example?

Collections & Data Structures

6. Explain internal working of **HashMap** (buckets, load factor, hashing).
7. Difference between **HashMap** and **ConcurrentHashMap**.
8. What is the difference between **ArrayList** and **LinkedList**?
9. Explain **fail-fast** vs **fail-safe** iterators.
10. How does `TreeMap` maintain order?

Exceptions & Threads

11. Difference between **checked** and **unchecked** exceptions.
12. Write code to create a **custom exception**.
13. How do you handle **deadlock** in multithreading?
14. Difference between `Runnable` and `Callable` interfaces.
15. What is the difference between **synchronized block** and **synchronized method**?

Java 8 Features

16. What is a **lambda expression**? Example?
17. Explain **Streams API** with an example (filter, map, collect).
18. What is an **Optional** in Java 8? Why do we use it?
19. Difference between `map()` and `flatMap()`.
20. What are **functional interfaces**? Give examples.

Section B: SQL/PostgreSQL Questions

21. Write a query to get the **second highest salary** from Employee table.
22. Difference between **INNER JOIN** and **LEFT JOIN** with examples.
23. How does **EXISTS** differ from **IN** in SQL?
24. Query to find employees with **duplicate salaries**.
25. What is a **window function** in PostgreSQL? Example with `ROW_NUMBER()`.
26. Write a query to find departments with **more than 5 employees**.
27. Explain difference between **HAVING** and **WHERE**.
28. Write a query to display the **Nth highest salary**.
29. What is the difference between **UNION** and **UNION ALL**?
30. How do indexes improve query performance in PostgreSQL?

Section C: Coding / DSA Problems

31. Implement a function to check if a string is a **palindrome**.
32. Write a program to find the **maximum subarray sum** (Kadane's algorithm).
33. Find the **longest substring without repeating characters**.
34. Implement **binary search** in Java.
35. Given two sorted arrays, find the **median** of combined array.
36. Write a program to check if two strings are **anagrams**.
37. Implement a **LRU Cache** in Java.
38. Reverse a linked list (iterative & recursive).
39. Write a program to find the **first non-repeating character** in a string.
40. Implement a program to find the **majority element** in an array.

Section D: Spring Boot & Full Stack Questions

41. Explain the difference between `@Controller`, `@RestController`, and `@Service`.
42. What is **dependency injection** in Spring Boot? Why is it useful?
43. Difference between `@Autowired` and constructor injection.
44. How do you connect a Spring Boot app with PostgreSQL? (properties config).
45. Explain Spring Boot **REST API** request lifecycle.
46. What are **Spring Boot starters**?
47. Difference between `GET`, `POST`, `PUT`, and `DELETE` methods in REST.
48. What is **CORS** and how do you enable it in Spring Boot?
49. Explain the difference between **Monolithic** and **Microservices** architecture.

50. How do you handle **exception handling** in Spring Boot REST API?

Section E: HR / Managerial Questions

51. Tell me about yourself (prepare a structured 2-min intro).
52. Why do you want to join **GlobalLogic**?
53. Describe a time you solved a **technical challenge** under pressure.
54. If your teammate is not completing work on time, how will you handle it?
55. Explain a project where you applied **Java + SQL + Full Stack** skills.
56. How do you keep yourself updated with new technologies?
57. What are your **strengths and weaknesses**?
58. If assigned a **new technology**, how will you learn and adapt?
59. How do you handle a situation when your **client disagrees** with your approach?
60. Where do you see yourself in the next 3 years?