TCS interview questions:

"I am a ***Lead Application Developer*** with over ***12 years of experience*** in software development, specializing in ***Java, Spring Boot microservices, and Android (Java/Kotlin)***. I have worked across ***multiplatform applications, cloud technologies, and microservices architectures*** in industries like ***retail, banking, and construction***.

🔹 I started my career at **Syncfusion**, where I worked on **multiplatform applications (mobile, web, and Windows)**, gaining experience in full-stack development.

🔹 At **Zoho**, I contributed to the **Zoho Chat application**, focusing on **API development, UI interactions, and Play Store deployments**.

🔹 Moving to **LTI**, I took on a leadership role, managing teams, coordinating with clients, and handling **large-scale construction applications**.

🔹 Currently, at **IBM**, I collaborate with **architects and clients** to design and deliver **Spring Boot microservices-based applications**. My current focus is on **retail and banking solutions**, leveraging **cloud technologies and containerized deployments** in **Azure Kubernetes Service (AKS)**.

I am passionate about **scalable architecture, performance optimization, and cloud-native solutions**, and I’m excited to bring my expertise to this role.

Because document article image title, lot of things are there.

No. Let me tell you the document article, everything we can, whenever we are doing this click event, it is happening from search as well.

How to send mail asynchornously in old java way

1. Where and how you are deploying your services in AWS

✅ **EKS (Kubernetes-based)** – Similar to **AKS**, works well for containerized apps.  
✅ **ECS (Docker-based)** – If you don’t need full Kubernetes.

1. How to scale up in AWS

Scaling in AWS can be done in **two ways**:  
1️⃣ **Vertical Scaling (Scale Up)** – Increase **CPU, RAM, or disk space** of existing instances.  
2️⃣ **Horizontal Scaling (Scale Out)** – Increase **number of instances/pods** to handle more load.

1. If there are 10000 requests then how springboot will handle or what changes you will have to do to manage

answer. Optimize Thread Management (Increase Concurrent Handling)

1. Program to get "How are" from "Hi how are you". Filter where there is a space before and after a word. – using regex
2. How can you ignore a field from being passed from your entities to a network

Use @JsonInclude(Include.NON\_NULL) (Ignore Null Fields)

**Which One Should You Use?**

| **Method** | **Use Case** |
| --- | --- |
| @JsonIgnore | Best for hiding sensitive fields like passwords |
| @JsonIgnoreProperties | When ignoring multiple fields globally |
| @Transient | Ignore field in the database but keep it in Java |
| @JsonInclude(Include.NON\_NULL) | Ignore fields only when null |
| DTO Pattern | Best for microservices & clean API design |

1. What logging framework are you using

You can use SLF4J (LoggerFactory):

1. How can you trace error between different microservices

Use Zipkin or Jaeger (For Visualization)

1. What tracing framework are you using to trace error

| **Method** | **Purpose** |
| --- | --- |
| **Spring Cloud Sleuth** | Automatically add traceId and spanId |
| **Zipkin / Jaeger** | Distributed request tracing |
| **ELK Stack (Elasticsearch + Kibana)** | Centralized log analysis |
| **Correlation ID Logging** | Manually track requests in logs |

1. Flow of a microservice from postman
2. How can you mask a credit card number that is coming in jsonBody that could becoming from multiple request in multiple json so that it doesn't get printed in logs
3. Design patterns used in your project with detailed explanation and cross question on the pattern
4. Microservice Design Pattern
5. Write code from controller to repository
6. Exception handling in springboot

## **Using** @ExceptionHandler **(Per Controller)**

* **Handle exceptions locally** within a specific controller.

✅ **Example:**

java

CopyEdit

@RestController

@RequestMapping("/users")

public class UserController {

@GetMapping("/{id}")

public User getUser(@PathVariable Long id) {

return userService.getUserById(id)

.orElseThrow(() -> new UserNotFoundException("User not found with ID: " + id));

}

@ExceptionHandler(UserNotFoundException.class)

public ResponseEntity<String> handleUserNotFoundException(UserNotFoundException ex) {

return new ResponseEntity<>(ex.getMessage(), HttpStatus.NOT\_FOUND);

}

}

📌 **When UserNotFoundException is thrown, it returns 404 NOT FOUND instead of a generic error.**

1. Best sorting algorithm
2. How to implement Kafka
3. What is circuit breaker design pattern

**Circuit Breaker Pattern** is used in **distributed systems** to prevent cascading failures when a dependent service is slow or unavailable. Instead of repeatedly trying a failing service, it **stops requests temporarily** and **allows recovery** before retrying.

🔹 **Why use it?**  
✅ Prevents system overload  
✅ Improves fault tolerance  
✅ Ensures graceful degradation

**Mansur Allam**

1. what is the spring bot and if the request is more than 20000 then how to handle it

Key Techniques to Handle 20,000+ Requests

**1.Scaling Approaches**

### ****2. Load Balancing (Distribute Requests Across Multiple Instances)****

Use **Nginx, AWS ALB, Azure Load Balancer, or Kubernetes Ingress** to distribute traffic.

### ****2.Caching (Reduce Database Load)****

Use **Redis, Memcached, or Hazelcast** to cache frequently accessed data.

### ****3. Rate Limiting (Prevent Overload & DDOS Attacks)****

Use **Bucket4j or Redis Rate Limiting** to limit excessive API calls.

### ****4. Asynchronous Processing (Handle Requests in Background)****

For tasks that don't need immediate results, **use Kafka, RabbitMQ, or CompletableFuture**.

### ****6. Circuit Breaker (Prevent System Failures)****

If external services (DB, API, Auth) fail, use **Resilience4j Circuit Breaker**.

### ****7. Kubernetes Auto-Scaling (Dynamically Adjust Resources)****

If deployed on **Kubernetes**, use **Horizontal Pod Autoscaler (HPA)**.

2.how to configure code commit, code build and code deploy in AWS.

3. if the producer produces a message to topic and the consumer reads it successfully but when it is

Find missing number from Numbers will be from in array 0-9 (3, 7,9,5,6,8,1,2,0)

Disadvantage of lambda

**Vijay Mani (Java Micros)**

Design patterns, Communicate microservices

How Streams works @transaction annotations

## **Enable Distributed Transactions using** @Transactional

Spring Boot does **not support transactions across multiple databases out of the box**, but we can achieve this using **Chained Transactions**.

## **2️⃣ Modify the** DatabaseService **to Handle Insert & Update Operations**

java

CopyEdit

How jpa handles 10000 req What is actuator,

Components in springboot

Kafka working

Vinoth VS

1. Finding 5th largest salary of employee table

2. Doing the same in java streams

3. Fibonacci series

4. Implement multithreading

5. Ordering the threads

6. What happens when an error occurs when consuming data in kafka

7. Convert list of employee class to map using java streams

8. Day to day routine work

9. About current Project

Sorting array.

Spring web flux,

Reactive programming.

Write restcontroller to create a customer,

Implement Functional interface,

Creating thread.

has context menu.

**Vijay Mani (Java Microservices)**

Design patterns, Communication between microservices

code

How Streams works @transaction annotation

How jpa handles 10000 requests

What is actuator, What is the use of webflux

Components in springboot

Kafka working

And how kafka send messages synchronously

What is the use of bootstrap.yml

How to store jsonObject in entity

How to filter requests

Callable vs Runnable

What is Future

X

**Sumit Kumar (Java Microservices)**

2. Microservice, Multi layer application architecture.

3. It should have a input filed with label on its top.

Q2. Problem statement:- Create a component with following requirements:-

When clicked again, the input field should be enabled again and also the state should be updated simultaneously.

Nandita

Swap two numbers without using temp variable

Find missing number from list

Numbers will be from 0-9 and identify time complexity

(3, 7,9,5,6,8,1,2,0)

Disadvantage of lambda expression.

What tech and cloud are you using to deploy microsevices

Sofing security and how it's implemented in your project

Vijay Mani (Java Microservices)

more

Design patterns, Communication between microservices How Streams works @transient annotatin

1. If there are 10000 requests then how springboot will handle or what changes you will have to do to manage

answer. Optimize Thread Management (Increase Concurrent Handling)

When your Spring Boot application faces **high traffic (10,000+ requests)**, you must ensure **scalability, performance, and reliability**. Here’s how you can handle this efficiently:

Spring Boot uses **Tomcat** by default, which has limited worker threads. Increasing the thread pool improves handling concurrent requests.

**✅ Increase Tomcat Thread Pool (Embedded Server)**

Modify application.properties:

properties

server.tomcat.threads.max=200

server.tomcat.threads.min-spare=50

server.connection-timeout=10s

💡 **Why?** More threads = More concurrent requests handled.

For **Jetty** or **Undertow**, configure their respective properties.

Sarath – technical question, tmrw

1. What spring module you are using?

## **Core Spring Boot Modules You Likely Use:**

1. **Spring Boot Starter Web** (spring-boot-starter-web)
   * For building RESTful APIs and web applications.
   * Uses **Spring MVC** and **Tomcat (embedded server)**.
2. **Spring Boot Starter Data JPA** (spring-boot-starter-data-jpa)
   * For database access using **Hibernate** and **JPA**.
   * Works with **EntityManager**, **Repositories**, etc.
3. **Spring Boot Starter Security** (spring-boot-starter-security)
   * For authentication & authorization (e.g., OAuth2, JWT, LDAP).
   * You might also integrate **Spring Security OAuth2** for OpenID Connect (net.openid:appauth).
4. **Spring Boot Starter Cache** (spring-boot-starter-cache)
   * Works with **Redis**, **Hazelcast**, and **Ehcache** to improve performance.
   * Useful for caching microservices data.
5. **Spring Boot Starter Kafka** (spring-kafka)
   * For event-driven messaging using **Apache Kafka**.
   * Handles producer-consumer patterns efficiently.
6. **Spring Boot Starter Actuator** (spring-boot-starter-actuator)
   * For monitoring and managing production applications.
   * Exposes health check endpoints (/actuator/health).
7. **Spring Boot Starter Cloud** (spring-cloud-starter)
   * Helps in **microservices development**.
   * Includes **Spring Cloud Config, Eureka, Feign, and Circuit Breaker (Resilience4j/Hystrix)**.
8. **Spring Boot Starter AOP** (spring-boot-starter-aop)
   * For **Aspect-Oriented Programming (AOP)**.
   * Useful for logging, security, and transaction management.
9. **Spring Boot Starter Validation** (spring-boot-starter-validation)
   * Uses **Jakarta Bean Validation (JSR-380)**.
   * For input validation in REST APIs (@Valid, @NotNull).
10. **Spring Boot Starter Batch** (spring-boot-starter-batch)

* For processing large amounts of data in chunks.
* Useful in data migration and ETL pipelines.

## **For Distributed & Scalable Applications:**

1. **Spring Boot Starter Redis** (spring-boot-starter-data-redis)
   * To use **Redis caching** and **Pub/Sub messaging**.
2. **Spring Boot Starter Elasticsearch** (spring-boot-starter-data-elasticsearch)
   * For full-text search capabilities and log analysis.
3. **Spring Boot Starter RabbitMQ** (spring-boot-starter-amqp)
   * If using **RabbitMQ for messaging** instead of Kafka.
4. **Spring Boot Starter GraphQL** (spring-boot-starter-graphql)
   * If building APIs using **GraphQL** instead of REST.

## **For Deployment & Scaling in AWS/Azure:**

1. **Spring Boot Starter Kubernetes** (spring-cloud-starter-kubernetes)
   * If deploying microservices in **Azure Kubernetes Service (AKS)** or AWS **EKS**.
2. **Spring Boot Starter AWS** (spring-cloud-starter-aws)
   * If using **S3, DynamoDB, SQS, or Lambda**.

What version of Java and What version of Spring you are using ?

(need to check with sugu)

Do you know lamda expression?

MathOperation add = (a, b) -> { a + b };

System.out.println(add.operate(5, 3)); // Output: 8

Do you know how to write custom in lamda function?

what about spring weblex?

 **Spring WebFlux** is best for **highly concurrent** applications.

 Use Mono<T> for **one result**, Flux<T> for **multiple results**.

 Works great with **MongoDB, R2DBC (PostgreSQL, MySQL, etc.)**.

 Use WebFlux for **WebSockets, Streaming APIs, and High Traffic** scenarios.

Have you done multi threading ?

Yes! I have extensive knowledge of **multithreading** in Java, including concepts like **Thread, Runnable, ExecutorService, Fork/Join, CompletableFuture, Synchronization, Locks, Atomic Variables, and Parallel Streams**.

 @Async runs the method **in a separate thread**.

 Requires **@EnableAsync** in your Spring Boot main class.

14) the life cycles of a thread?

## **📌 Summary Table**

| **State** | **Description** |
| --- | --- |
| **New** | Thread is created but not started (new Thread()). |
| **Runnable** | Thread is ready but waiting for CPU (start()). |
| **Running** | Thread is executing (run()). |
| **Blocked** | Thread is waiting for a resource (lock). |
| **Waiting** | Thread is waiting indefinitely (wait()). |
| **Timed Waiting** | Thread is waiting for a fixed time (sleep(), join(time)). |
| **Terminated** | Thread has completed execution (run() finished). |

15) Have you created / used spring kafka producers or spring kafka consumers?

What is the difficulties faced in your project and how to tackle that