TOPICS:

·         Your experience with any Cloud, Deployment of applications on Cloud etc.

·         DevOps – Docker or Kubernetes, Jenkins

·         Should be able to talk about the current project effortlessly (Logical view, Sequence Diagrams, Application Layering, End to end View)

·         Involvement in Non Functional Requirements like application scalability, Caching, Security etc.

Technology Selection/Comparision

**Microservices:**

MS patterns: Orchestration, Choreography, Entity Modeling, Transactions (please research on these)

Aware of creating Rest endpoints using Springboot. Knows about Swagger. Aware of versioning concept. Able to explain the REST best practices. Able to explain how OAuth works. Has knowledge of JWT and was able to explain the JWT structure. Has used AWS in his projects for deploying Microservices.  Was able to explain the benefits of Microservices Architecture vs Monolith. Able to explain the different communication methods used in Microservices architecture. Is aware of the patterns **like CQRS, SAGA, API composer**, Service Discovery and API gateway Has knowledge of the circuit breaker pattern and was able to explain with example. Caching (in MS or Application level).

Exposure on Hystrix, Eureka – Netflix Suite – (preferred)

Swagger for API Documentation

How to roll back the Transactions in case of exceptions, How to cancel?

CQRS is API Strategy and DB Strategy

API Security – Auth 2.0, JWT etc.

API gateways: Any 3rd party API gateways, Policies, roles and responsibilities, filters etc.

**Core Java:**

Jackson Library, Json to map etc.

1.       OOPS & Design Principles:

·         Person should be able to talk about each of these with examples. COMPOSITION, AGGREGATION, ASSOCIATION, ENCAPSULATION, ABSTRACTION, POLYMORPHISM, INHERITANCE.

·         Deep understanding of SOLID design principles – Must have

2.       Design patterns:

·         Understands the need of Design Patterns.

·         Singleton, Factory, Builder, Template Stategy, Adaptor, Decorator –  Should have good understanding of atleast 2 DPs from each category: Structural, Behavioral, and Creational. And should be able to explain with examples and usage in previous project.

3.       Algorithms:

·         Can provide their (Sorting, Searching, Hashing, etc.) implementation in java. Understands O(N) analysis; provides a reasonably optimal solution to a simple problem - generally involves using map/list DS;  (array duplicity problem/ palindrome/ middle value of list, find element from list etc.)

5.       Core Java Fundamentals:

Should know purpose of Keywords, static binding , dynamic binding , overloading rule , overriding rule in terms of access modifier, exception handling, impact of dynamic linking on performance , how to improve performance by using final keyword, what’s default implementation of hash code and equal, cloning, immutability, advantage of immutability , importance of final in security , Exception handling rules

6.       Generics:

Upper and Lower Bounds, Wild Card, Type Eraser

7.       Collections:

Good Understanding of:

·         Collections like: LinkedHashSet, TreeSet, , LinkedHashMap, TreeMap, ArrayList, Vector, LinkedList, Hashmap, HashTable,  HashSet.etc. and is able to explain their usage through examples like when to use what.

·         HashMap concepts like Hashing, Collision, Rehashing, LoadFatcor, etc

·         Is able to write code to sort objects using Comparable and Comparator.

·         Equals and hashcode contract and its implications.

·          Failsafe and failfast iterators and their impact when used on various collections

**8.       Concurrent API**

·         How concurrent hash map internally managing a lock how segmentation works

·         How many thread can work on concurrent hash map

·         Benefit of using concurrent hash map over hash table and synchronize map

·         What kind of business use cases can be implemented using  concurrent hash map

·         How blocking queue works? What kind of problems can be solved by using blocking queue?

·         When should we use linked blocking queue and when array blocking queue

·         Implementation of blocking queue

·         How to use blocking queue in inter thread communication, what’s fail safe iterator

·         How to implement thread pool , what’s advantage of thread pool , how many type of thread pool do we have ,

·         How we can use executor service , how to use executor service to implement parallel /pipe line processing.

·         What kind of business problem can be solved by cyclic barrier and count down latch and how it works?

**9.       Multithreading**

Executor Framework, Deadlock, Fork & Join, Synchronization, Latch and barrier, Reentrantlock and Concept of Threadlocal, should know how wait , notify , sleep , join, how locking works , what’s class level lock , what’s object lock, how to implement inter thread communication by using wait and notify , how volatile works , volatile keyword , how to implement thread pool in java 4 , how important is immutability in multithreading , what code can create deadlock , what code can create starvation

Few Real time Questions:

How do you take Heap dump in Java?

What’s the difference etweek heap and Thread Dump?

Blocking Queue functionality (Array and Linked)

Concurrent Hashmap internal working, Difference between Concurrent Hashmap and hashmap

Concept of Threadlocal, Completable Future

Threadpool and its implementation. How to write a custom threadpool? Different types of Threadpool – Fixed, Cached, Single, Scheduler, Work Stealing

Object level vs class level locking

Wait, Notify and Notifyall – Use cases

Difference between Callable and Runnable Interfaces

Difference between Yield and Join

Semaphores concept

10.   Java memory Management (JMM)

Profiler, Finalize method, JVM Configuration, GC Algos (Mark & Sweep, Series & parallel GC, Full and partial GC), Should know java memory model , should know heap , how garbage collection works , how to optimize memory , should know the reason of Perm gen Exception , reason of Out of memory exception , should be aware how to do memory profiling , how to identify which code consuming

11.   Java 8 Features:

·         Is able to explain in some detail about some of the features in Java 8, such as Lambda Expressions, Streams (Parallel, Serial, lazy), Streams, Predicate, Function Interface and its various flavours, CompletableFuture, Optional in Java 8

12.   RDBMS:

·         Able to write queries using different types of aggregate functions, Group By, Having Clause. – This is something you can most definitely expect.

·         Able to write queries using joins (Outer, inner, left, right)

·         Good understanding of indexes – Clustered and Non Clustered, Concept of sharding in databses

13.   JMS (ActiveMQ, RabbitMQ or KAFKA, AWS SQS) -  any one is fine

Pub-Sub, Queue and Topic (Difference), Event based programming, Distributed Tracing, Scaling

14.   Spring REST:

Key attributes of the HTTP protocol – get, Put, Post and delete and their differences

Versioning, Pagiing, Pagination, Mocking Concepts

Error response Codes

Undesrtands how @EnableWebMvc annotation helps to do some useful things in Spring REST, along with @Controller  annotation which acts as a central artifact in the entire Web Tier of the RESTful API.

REST principles (Uniform interface,  Stateless interactions ,Cacheable, Client-Server, Layered System, Code on Demand ).

16.   Spring Boot:

Should be able to demonstrate knowledge of multiple approaches to create Spring Boot project like using Spring Maven Project, Spring Starter Project Wizard,Spring Initializr,Spring Boot CLI etc.

\* Is able to create a simple Spring Boot Application.

\* Is able to demonstrate knowledge on Spring Boot Annotations.

\* Is able to demonstrate knowledge about Spring Boot Starters

18.   Production Support/Production Deployment & Bug Fixing (Blue Green deployment)

 How have you been involved? Production deployment checklist that was followed and operational steps taken?

20.   JUNIT/TDD & Mockito:

Is able to demonstrate some working knowledge of writing unit tests.

\* Is able to talk about critical classes of Unit testing framework, including key Annotations

   and TestSuites and how to use them.

\* Is able to explain the concept of Mocking frameworks.

\* Is able to  talk about some features of Unit tests and Mocking frameworks.

\* Is able to understand the concept of TDD's.

Log Monitoring Tools:

ELK Stack, Greylog, Splunk, appdynamic, Grafana, Sumologic, etc… - any one is fine

|  |
| --- |
|  |
| Shubhada Late Associate Talent Acquisition |

------------------------------------------------------------------------  
Disclaimer The information in this email and any attachments may contain proprietary and confidential information that is intended for the addressee(s) only. If you are not the intended recipient, you are hereby notified that any disclosure, copying, distribution, retention or use of the contents of this information is prohibited. When addressed to our clients or vendors, any information contained in this e-mail or any attachments is subject to the terms and conditions in any governing contract. If you have received this e-mail in error, please immediately contact the sender and delete the e-mail.