Introduction:

Hello, I'm **Kishan Kumar**, a software developer with **3.7 + years of experience** specializing in **Java, Spring Boot, Hibernate, React.js, Kafka, Redis, and microservices architecture**. I have worked extensively on **backend development, system optimization, cloud integrations, and performance tuning**. Currently, I am working at **Mphasis**, where I contribute to modernizing and optimizing **enterprise applications**.

I have worked on two major projects:

1. **A high-performance banking application**, where I focused on **security, scalability, and efficient database operations**.
2. **A global logistics tracking system**, where I was responsible for **migrating legacy code, enabling real-time tracking, and optimizing system performance using event-driven architecture**.

Throughout these projects, I have tackled challenges like **thread management, database query optimization, real-time data processing, security enhancements, and cloud storage** to improve system efficiency and scalability.

Project 1: Banking Application (March 2022 – October 2022)

Problems Faced & Solutions Implemented

❌ **Problem: Thread Pool Exhaustion in Embedded Tomcat Server**

- The server struggled to handle multiple concurrent requests due to **limited thread pool size**, causing slow response times and occasional failures.

✅ **Solution:**

- Increased the **Tomcat thread pool size** and tuned **connection timeouts** to handle a higher number of concurrent requests.
- Introduced **asynchronous processing using @Async and WebFlux** to prevent thread blocking, improving request handling efficiency and system responsiveness.

❌ **Problem: Slow Database Queries Causing Performance Bottlenecks**

- Certain queries were taking **too long to execute** due to **N+1 query issues** in Hibernate and missing indexes.

✅ **Solution:**

- Used **Hibernate's JOIN FETCH** to eliminate N+1 issues.
- Created **indexes** on frequently queried columns to speed up database lookups.
- Implemented **HikariCP connection pooling** to optimize database connections and minimize latency.

❌ **Problem: No Structured Way for Customers to Report Issues**

- Customers had no proper system to **report banking issues** or upload supporting documents.

✅ **Solution:**

- Designed a **microservices-based issue reporting system** with REST APIs, allowing users to log issues and upload images securely.
- Used **AWS S3** for scalable and secure image storage.

❌ **Problem: Large File Uploads Slowing Down System Performance**

- Uploading large documents directly to the server was **slowing down the application** and consuming excessive storage.

✅ **Solution:**

- Integrated **Amazon S3** for file storage, ensuring efficient large file handling.
- Used **pre-signed URLs** to allow secure, direct file uploads without overloading the backend.

❌ **Problem: Monolithic Architecture Making Deployment & Scaling Difficult**

- The application was built as a **monolithic system**, leading to **difficult deployments and scalability issues**.

✅ **Solution:**

- Implemented a **microservices architecture**, decoupling services like **issue logging, reporting, and authentication**.
- This improved **scalability, fault tolerance, and independent service deployment**.

❌ **Problem: Security Vulnerabilities in User Authentication**

- The authentication system was vulnerable to **security risks like unauthorized access and session hijacking**.

✅ **Solution:**

- Implemented **Spring Security with OAuth2** to ensure **secure authentication and role-based access control (RBAC)**.
- Used **JWT tokens** for secure, stateless authentication.

❌ **Problem: Unoptimized Project Dependencies Causing Build Delays**

- The project had **excessive, unnecessary dependencies**, increasing **build times** and causing **compatibility issues**.

✅ **Solution:**

- Used **Apache Maven** for dependency management and build automation, ensuring efficient project workflows.

Project 2: Logistics Tracking System (November 2022 – Present)

Problems Faced & Solutions Implemented

❌ **Problem: Legacy System Using Java 1.4 with Synchronized Locks, Causing Deadlocks**

- The system relied on **synchronized locks**, leading to **frequent deadlocks and slow performance** in multi-threaded environments.

✅ **Solution:**

- Migrated the system from **Java 1.4 to Java 1.8** and replaced **synchronized locks** with **ExecutorService and Java Concurrency API**.
- This improved **thread management**, **reduced deadlocks**, and **enhanced system performance**.

❌ **Problem: No Real-Time Tracking Available for Couriers**

- Users had no way to **track shipments in real-time**, leading to **customer dissatisfaction**.

✅ **Solution:**

- Developed a **React-based UI integrated with Spring Boot**, allowing users to **view estimated delivery times in real-time**.

❌ **Problem: High API Response Times Due to Excessive Database Queries**

- The system made **repetitive database queries**, increasing **response times** and **server load**.

✅ **Solution:**

- Implemented **Redis caching** to store **frequently accessed data**, reducing **redundant API calls** and improving performance.

❌ **Problem: Delayed Updates in Shipment Tracking**

- The system **processed updates in batches**, causing **delays in tracking status changes**.

✅ **Solution:**

- Integrated **Kafka** for **real-time messaging**, ensuring instant **shipment status updates**.

❌ **Problem: Manual Proof-of-Delivery Storage Leading to Inefficiencies**

- Delivery proof images were **stored manually**, making retrieval and management inefficient.

✅ **Solution:**

- Utilized **AWS S3** for **secure, scalable proof-of-delivery storage** with easy retrieval.

❌ **Problem: Inefficient Query Execution Causing Performance Lags**

- Complex queries and **lack of caching** were **slowing down system performance**.

✅ **Solution:**

- Implemented **Hibernate second-level caching** and used **Hibernate JPA templates** for optimized database interactions.

❌ **Problem: Security Risks in the Courier Tracking System**

- The application was vulnerable to **unauthorized access** and potential **data breaches**.

✅ **Solution:**

- Secured the system with **Spring Security (OAuth2)**, restricting unauthorized access with **role-based authentication**.

❌ **Problem: Difficulty in Scaling Due to Monolithic Architecture**

- Scaling the system required **deploying the entire application**, increasing **downtime and complexity**.

✅ **Solution:**

- Designed **microservices for shipment tracking, notifications, and proof-of-delivery**, allowing independent **scalability and fault tolerance**.

❌ **Problem: High System Latency Due to Inefficient Thread Execution**

- Poor multi-threading design led to **slow system performance** and **resource underutilization**.

✅ **Solution:**

- Optimized **multi-threading using Java Concurrency utilities**, improving **task execution and resource utilization**.

Conclusion

Through these projects, I have gained expertise in **building scalable, high-performance applications** with modern **Java and Spring Boot frameworks**. My experience spans across **microservices architecture, database optimizations, caching mechanisms, multi-threading, real-time processing, and cloud integrations**. I am always eager to tackle challenging problems and optimize system efficiency.

Achievements in This Project:

- Successfully **optimized system performance**, reducing **response times by 40%**.
- Conducted **code reviews**, improving code quality and maintainability.
- Received **high client ratings for problem-solving and efficiency improvements**.

HR Questions

Q. Why do you want to switch organisation?

Ans. I have been with my current organization for the past 5 years. Initially, I started with frontend development, but gradually I was also given backend tasks. As someone who is always eager to learn and grow, I saw this as a great opportunity and took the initiative to upskill myself. Over the years, I've built strong expertise across both frontend and backend technologies.
Now, I'm looking for a new environment where I can continue to challenge myself, expand my skill set, and contribute to larger, more complex projects. I believe that switching organizations at this stage will expose me to new architectures, workflows,and team dynamics — all of which are important for my professional growth and learning.

Q. What do you know about our company?

Ans. I know your company is doing great work in [domain], and I've heard a lot of positive things about your products and engineering team. I really like that you use modern technologies and give developers a chance to learn and grow. That kind of environment is exactly what I'm looking for in my next role.

Q. Can you describe a challenge you faced and how you handled it?

Q. What are your salary expectations?

Ans. Yes, I'd love to know more about the tech stack your team is using and how you approach learning and development internally.

Links

1. Javascript : namaste javascript
2. DSA : striver sde sheet
3. SQL Database:SQL DATABASE
4. Mongodb : MongoDb