## Richa A

Email: 0fj-f2f-pwf@mail.dice.com

Phone: 5592063320

### Summary

|  |  |
| --- | --- |
| Previous |  |
| Preferred | Java Full Stack Developer |
| Location | Boston, MA, US |
| Desired Work Settings | No Preference |
| Willing to Relocate | No |
| Work Authorization(s): | Authorized to work in the United States on a full-time basis. Now or in the future will require visa sponsorship for employment in the United States. |
| Employment Type | Contract - Corp-to-Corp Contract to Hire - Corp-to-Corp |
| Total Experience | Unspecified |
| Education | Unspecified |
| Profile Source | Dice |
| Profile Downloaded | Monday, December 9, 2024 |

******Richa**

Full Stack Java Developer

Professional Summary

Skilled **Full Stack Java Developer** with 6 years of extensive experience in designing, developing, and deploying scalable web applications and microservices, proficient in both frontend and backend technologies, with a deep understanding of cloud platforms, DevOps practices, and API development.

* Proficient in **Java, JavaScript and TypeScript for** developing scalable backend systems with Java and Python, while creating interactive frontends using JavaScript and TypeScript for seamless cross-platform applications.
* Experienced with **Spring Boot, Spring MVC, Angular, and React**, enabling rapid development of microservices-based backends and responsive web applications, enhancing user engagement and overall performance.
* Skilled in **MySQL, PostgreSQL, MongoDB, and Redis**, designing optimized database schemas and implementing caching mechanisms to improve data access speed and reduce application latency.
* Expertise in **RESTful APIs, OAuth 2.0, SOAP, and JWT**, building secure and scalable APIs for smooth integration between services, handling authentication, and ensuring safe data exchange.
* Extensive experience with **AWS and GCP,** deploying cloud-based applications with services like **Lambda, EC2, ECS, S3, and RDS,** ensuring fault tolerance, scalability, and efficient resource management.
* Proficient in **Infrastructure as Code (IaC)** with Terraform and AWS CloudFormation, automating infrastructure provisioning and ensuring consistency across environments through reusable, modular code.
* Skilled in **Docker, Kubernetes, and Amazon ECS**, orchestrating containerized applications to improve scalability, automate deployments, and manage complex microservices architectures efficiently.
* Hands-on experience with **DevOps tools** like Jenkins, Git, Maven, and AWS CodePipeline, automating CI/CD processes to streamline code integration, testing, and deployment in fast-paced environments.
* Proficient in **monitoring tools** such as Prometheus, Grafana, AWS CloudWatch, and X-Ray, implementing real-time monitoring to ensure application reliability and proactive issue resolution.
* Familiar with **distributed tracing tools** like Jaeger and Zipkin, identifying and resolving performance bottlenecks in microservices environments to improve system response times.
* Proficient in **testing frameworks** including JUnit, Mockito, JMeter, and Gatling, ensuring high code quality through unit, integration, and load testing to maintain application stability under stress.
* Experienced with **Apache Kafka**, designing event-driven architectures for real-time data processing, enabling efficient asynchronous communication and handling high-throughput data streams.

Skills

**Programming Languages**: Java, JavaScript, TypeScript,

**Frameworks**: Spring Boot, Spring MVC, Angular, React

**Databases**: MySQL, PostgreSQL, MongoDB, Redis

**APIs**: RESTful APIs, SOAP, OAuth 2.0, JWT

**Cloud Platforms**: AWS (Lambda, EC2, ECS, S3, RDS, CloudFront), Google Cloud Platform (GCP)

**Infrastructure as Code (IaC)**: Terraform, AWS CloudFormation

**Containerization**: Docker, Kubernetes, Amazon ECS, Fargate

**DevOps Tools**: Jenkins, Git, Maven, AWS CodePipeline

**Monitoring & Logging**: Prometheus, Grafana, AWS CloudWatch, AWS X-Ray

**Distributed Tracing**: Jaeger, Zipkin

**Testing Frameworks**: JUnit, Mockito, JMeter, Gatling

**Messaging**: Apache Kafka

Work Experience

**Full Stack Java Developer** | **Mckesson *Nov 2021-Present***

**Project: *Prescription Management and Delivery System***

**Summary:**McKesson, as one of the largest healthcare services companies, needed a reliable and scalable platform for managing prescriptions and ensuring timely delivery of medications to pharmacies and patients. I was a part of the team that worked on developing and maintaining a Prescription Management and Delivery System, designed to streamline the end-to-end process of handling prescription orders, tracking inventory, and managing deliveries efficiently. This project aimed to integrate McKesson’s large network of healthcare providers and pharmacies with a single digital solution, ensuring real-time tracking and secure handling of sensitive patient data.

**Key Responsibilities:**

* Developed backend microservices in Java (Spring Boot) to handle various functionalities of the prescription lifecycle, including order creation, validation, inventory checks, and delivery scheduling. Each service was designed with a clear separation of concerns, allowing for better maintainability and scaling as the system grew.
* Collaborated with front-end teams to integrate Java-based REST APIs with React, ensuring seamless data flow and synchronization between the server-side logic and client-side interfaces. This enabled efficient data rendering and improved the overall performance of McKesson's web applications, providing a responsive user experience.
* Designed and developed a secure API layer using RESTful services to expose core business logic to external systems and integrate with partner pharmacies. The APIs enabled pharmacies to place prescription orders and check order statuses in real-time. This API layer adhered to strict healthcare data security standards (HIPAA) and incorporated OAuth 2.0 for secure authentication.
* Created dynamic, responsive UIs using React for pharmacy staff and patients to easily manage orders, track delivery status, and view prescription history. This involved building reusable React components, using Redux for state management, and ensuring optimal performance through lazy loading and code splitting.
* Leveraged GCP’s Cloud SQL and Cloud Functions to handle prescription data processing. Cloud SQL was used for managing structured patient and inventory data, while Cloud Functions automated tasks like sending notifications to pharmacies when prescription orders were ready or generating delivery reports.
* Implemented real-time order tracking using WebSockets, which allowed both pharmacies and patients to monitor the status of their prescription deliveries. This feature provided up-to-the-minute information, improving transparency and customer satisfaction while reducing manual order updates.
* Integrated the platform with McKesson’s existing supply chain system to automate inventory checks and order fulfillment. Used Spring Batch to implement scheduled jobs that synced prescription orders with the inventory, ensuring orders were fulfilled without delays.
* Implemented caching strategies using Redis and Memcached to enhance application performance by reducing database load and speeding up response times for frequently accessed data. This was particularly useful for optimizing API responses and ensuring a smoother user experience for McKesson's supply chain management system.
* Applied containerization techniques using Docker for consistent development, testing, and production environments. Docker containers encapsulated microservices and their dependencies, ensuring smooth deployments across cloud infrastructure and minimal downtime during updates.
* Implemented CI/CD pipelines with Jenkins to automate testing and deployment processes, reducing the time from code commit to production release. Every code change was automatically built, tested, and deployed, ensuring rapid delivery cycles without sacrificing quality.
* Optimized database queries using Hibernate and JPA, focusing on reducing latency in fetching prescription and inventory data. Optimizations included query caching, indexing frequently accessed tables, and using pagination for large data sets to improve response times.
* Collaborated with cross-functional teams, including UI/UX designers and healthcare compliance experts, to ensure the system was both user-friendly and compliant with healthcare regulations. This included adding accessibility features and securing sensitive patient data at all stages of the prescription lifecycle.

Cloud Engineer| **Netflix *Apr 2019-Nov 2021***

**Project: *High-Availability Video Streaming Infrastructure with Java Integration***

**Summary:**As part of the cloud engineering team for Netflix, I contributed to developing and maintaining a highly scalable, resilient cloud infrastructure on AWS to support Netflix’s global video streaming services. My role involved working closely with development teams using Java to optimize microservices, ensuring efficient cloud resource management and seamless content delivery. The project integrated AWS cloud capabilities, DevOps automation, and microservices architecture to ensure Netflix could handle traffic spikes and maintain service reliability.

**Key Responsibilities:**

* Developed cloud-native Java microservices in AWS Lambda and ECS (Elastic Container Service) for Netflix’s backend systems, handling real-time requests for content delivery and subscription management. Leveraged Spring Boot and Spring Cloud for building lightweight, scalable services that integrated seamlessly with AWS cloud infrastructure.
* Automated infrastructure provisioning using Terraform and AWS CloudFormation, building Infrastructure-as-Code (IaC) templates for deploying multi-region, highly available applications. This provided consistency and agility in managing environments and allowed rapid deployments of Java-based services across different AWS regions.
* Optimized application performance using Amazon EC2 Auto Scaling and Elastic Load Balancing to handle large spikes in traffic during new show releases. Configured auto-scaling policies and leveraged EC2 instances tailored for Java applications, ensuring the platform could dynamically adjust capacity based on demand while minimizing costs.
* Integrated Java microservices with AWS S3 and RDS for secure storage and efficient data management, ensuring that content metadata, user preferences, and payment records were stored and retrieved securely. Used Hibernate with Java for ORM and optimized queries for fast data access in Netflix’s high-traffic environment.
* Implemented secure, role-based access control (RBAC) using AWS IAM for Java applications. Integrated OAuth 2.0 and JWT authentication mechanisms to protect sensitive user data and ensure only authorized systems could access critical microservices and APIs.
* Enhanced service reliability and performance by implementing distributed tracing using tools like **Jaeger** and **Zipkin** to monitor microservices communication across Netflix's infrastructure. This helped identify performance bottlenecks, trace failed requests, and improve response times, enabling faster debugging and optimization of Java-based applications.
* Automated integration testing for Java microservices using **JUnit** and **Mockito**, ensuring that each component functioned as expected in the cloud environment. Established test-driven development (TDD) practices with the team, improving the quality of new features and preventing regressions during deployments, leading to more stable releases.
* Streamlined CI/CD pipelines with Jenkins and AWS CodePipeline for deploying Java-based microservices. Set up automated build and test processes using Jenkins and Maven for continuous integration, ensuring new features and updates could be deployed to production with minimal downtime and risk.
* Optimized video streaming performance using AWS CloudFront and Global Accelerator to ensure low-latency content delivery across Netflix’s global audience. Integrated Java APIs with AWS CloudFront’s edge locations, providing users with a faster and more reliable viewing experience by reducing the round-trip time to data centers.
* Deployed containerized Java applications using Docker and Amazon ECS, enabling consistent deployments across development, staging, and production environments. Managed serverless containers with AWS Fargate, reducing the operational overhead of managing infrastructure and allowing developers to focus on writing Java code.
* Improved cloud security with AWS KMS and Secrets Manager, managing encryption keys for Java applications and ensuring secure handling of sensitive information like API keys, user credentials, and streaming data. This enhanced overall data security and compliance with regulations like GDPR.
* Collaborated with cross-functional teams to migrate legacy Java applications to cloud-native architectures. Worked with development and DevOps teams to refactor monolithic Java applications into microservices, deployed using AWS ECS and Lambda, which enhanced scalability and simplified the deployment process.

Software Developer| **AirBnB *Feb 2018-Apr 2019***

**Project**: ***Booking and Reservation Platform Enhancement***

**Summary**:  
As part of the core engineering team at Airbnb, I was involved in the enhancement and optimization of Airbnb’s booking and reservation platform. This project aimed to scale the system to handle increasing user traffic, ensure low latency, and introduce new features to improve the user experience. The work involved backend development using Java, integrating third-party APIs, database optimization, and ensuring seamless deployment of new features.

**Key Responsibilities:**

* Developed and optimized Java-based microservices to manage Airbnb's core booking processes, ensuring efficient handling of millions of reservations daily. Utilized Spring Boot for developing RESTful APIs and ensured these services were lightweight and scalable for a global user base.
* Integrated third-party payment gateways (e.g., Stripe) to handle secure transactions within Airbnb’s platform. Leveraged OAuth 2.0 for secure authentication and JWT (JSON Web Tokens) to ensure secure, stateless communication between services, improving payment process efficiency.
* Collaborated with front-end teams to design and implement APIs that seamlessly integrated with mobile and web applications. Worked closely with UI/UX designers to deliver responsive and interactive booking experiences, ensuring consistency across platforms.
* Optimized SQL and NoSQL databases (MySQL and MongoDB) to improve performance of data retrieval and storage, specifically for user bookings, property listings, and search functionalities. Implemented query optimization, indexing strategies, and database partitioning to handle high transaction volumes and maintain quick response times.
* Implemented asynchronous processing with Apache Kafka to handle event-driven architecture for tasks like sending booking confirmations, notifications, and managing real-time availability. This allowed the system to process large volumes of messages efficiently, without blocking the main application thread.
* Ensured code quality and performance by implementing unit and integration tests using JUnit and Mockito. Set up continuous integration (CI) pipelines with Jenkins, automating testing and deployment processes to ensure smooth releases of new features.
* Enhanced system reliability by implementing a retry mechanism and circuit breaker pattern using Resilience4j, preventing cascading failures and improving fault tolerance in Airbnb’s booking platform. This ensured high availability even during service interruptions or external API failures.
* Implemented Redis for caching frequently accessed data, such as property listings and user sessions, significantly improving system response times. This caching strategy helped reduce database load and ensured a faster experience for users searching for properties.
* Collaborated with DevOps teams to implement containerization using Docker and Kubernetes, ensuring consistency in deployments across development, staging, and production environments. This allowed for easier scaling and reduced the complexity of managing Java microservices in a distributed environment.
* Improved the platform’s scalability and performance by load testing using tools like Gatling and JMeter, identifying bottlenecks in the system and implementing fixes to handle peak loads effectively, such as during promotions or high-traffic seasons.