Sri Teja
teja.sript9@gmail.com | + 1 (512) 571-3855
LinkedIn | GitHub | Portfolio | Medium

Professional Summary:

- 9+ years of experience as a Full Stack Java Developer building scalable, real-time enterprise applications across banking, healthcare, and retail domains
- Extensive experience developing microservices using **Spring Boot, Spring Cloud**, and **Kafka** for high-volume, low-latency transaction systems
- Hands-on experience building responsive frontend applications using **ReactJS** and **Angular**, with emphasis on performance optimization and real-time data handling
- Proven ability to design and develop secure **RESTful APIs**, event-driven architectures, and distributed processing workflows
- Experienced in cloud-native deployments using **AWS**, **Docker**, **Kubernetes**, **Terraform**, and **CI/CD** pipelines for scalable infrastructure management
- Skilled in **Redis** caching, system monitoring, and observability using **Prometheus**, **Grafana**, and **Splunk** for production support and debugging

Technical Skills:

Java/J2EE Technologies	Java 8+, JDBC, JMS
Frameworks	Spring (Core, Boot, Security, AOP), Spring MVC, Spring Cloud, Hibernate (JPA), ReactJS, Angular, NodeJS (basic exposure)
IDE & Editors	IntelliJ IDEA, Eclipse, Visual Studio Code
Web Services	RESTful APIs, JAX-RS, JAX-WS, GraphQL
Design Methodologies	Microservices Architecture, Event-Driven Architecture, MVC Design Pattern
Open Source/Version Control	Git, Maven, JUnit, Log4j
Platforms	Windows, Linux (Ubuntu), Unix
AWS and Azure	AWS (EC2, S3, RDS, API Gateway, Lambda), Azure (Key Vault, Blob Storage, Active Directory), CloudFormation
Databases	PostgreSQL, MySQL, Oracle, MongoDB, Redis
Web Technologies	HTML5, CSS3 (Bootstrap), JavaScript (ES6+), TypeScript, JSON
Scripting Languages	Maven, Bash Scripting
Tools	JIRA, Jenkins, GitHub Actions, Terraform
Web/Application Servers	Apache Tomcat
Containers	Docker, Kubernetes
Authentication	OAuth 2.0, JWT, OIDC, MFA

Professional Experience:

PNC Bank, Farmers Branch, Texas, United States (Remote) **Feb 2023 - Present**
Role: Senior Full Stack Java Developer
Project: Real-Time Payments Orchestration & Fraud Decisioning Platform
Working on a real-time payments platform supporting instant transfer workflows such as Zelle, RTP, and internal bank transactions. The project focuses on improving transaction reliability, fraud evaluation accuracy, and overall system stability during high-volume payment activity.
Responsibilities:

- Real-time transaction lifecycle states (initiated, processing, completed, flagged) were surfaced through ReactJS and WebSocket-driven updates, allowing users to track payment progress without manual refreshes.
- Introduced a staged payment workflow with controlled validations between steps, helping reduce incomplete submissions and inconsistent transaction states.
- High-frequency interactions such as beneficiary search were optimized through debouncing and request cancellation to avoid redundant backend calls during rapid input changes.
- Selective Redux subscriptions and memoization strategies were applied to reduce unnecessary re-renders across frequently accessed account and transaction views.

- Reusable UI components were standardized around banking-specific workflows, so validation behavior, transaction states, and interaction patterns remained consistent across modules.
- Payment APIs were structured around idempotent transaction handling so retries caused by network interruptions would not result in duplicate processing.
- Kafka-based asynchronous processing was introduced for payment initiation, fraud evaluation, ledger updates, and downstream notifications, supporting the processing of millions of transaction events daily while keeping workflows decoupled under load.
- Fraud evaluation logic combined velocity checks, geo-location mismatch detection, and behavioral deviation analysis to identify suspicious transaction patterns while reducing unnecessary fraud flags.
- Short-lived transaction state and fraud evaluation context were maintained in Redis to minimize repeated database access during active workflows, helping reduce API response latency during peak transaction periods.
- Resilience4j-based circuit breaker handling was introduced around downstream integrations to isolate service instability and avoid cascading transaction failures.
- External service failures were handled through retry strategies with exponential backoff, reducing disruption caused by intermittent connectivity issues.
- Designed API-level rate limiting and throttling to handle peak loads and prevent abuse scenarios.
- Audit and reconciliation workflows were scheduled through Spring Batch to validate consistency between transactional systems and downstream financial records.
- Access to payment and fraud endpoints was protected through OAuth 2.0 and JWT-based authentication with role-scoped authorization controls.
- Deployed microservices on Kubernetes with horizontal pod autoscaling based on CPU and message queue lag, helping maintain stable performance during peak transaction windows and traffic spikes.
- Multi-stage Docker builds were used to standardize runtime environments while keeping deployment artifacts lightweight across services.
- Implemented distributed tracing using correlation IDs across services, reducing debugging and issue-resolution time for failed or delayed transactions in distributed workflows.
- Operational metrics, centralized logging, and alerting workflows were integrated through Splunk and monitoring dashboards to improve production issue visibility.
- Used Terraform for infrastructure provisioning, ensuring consistent and repeatable environment setup across development and production.
- Built and maintained CI/CD pipeline automation using Jenkins and GitHub Actions, integrating unit and integration testing using JUnit and Mockito to ensure code quality and reliable deployments.
- API response efficiency was improved through caching adjustments, payload refinement, and selective query optimization under high transaction load.

Environment:
Java, JavaScript (ES6+), ReactJS, Redux, WebSockets, Spring Boot, Spring Security, Resilience4j, Spring Batch, Apache Kafka, OAuth 2.0, JWT, MFA, PostgreSQL, MongoDB, Redis, AWS (EKS, API Gateway, S3, RDS), Azure (AKS), Docker, Kubernetes, Jenkins, GitHub Actions, Terraform, Splunk, Prometheus, Grafana, Idempotency, Event-Driven Architecture, Circuit Breaker, Rate Limiting, Distributed Tracing

Prime Healthcare, North Las Vegas, Nevada, United States **April 2021 - Jan 2023**
Role: Full Stack Java Developer
Project: Cardiac Monitoring & Predictive Risk Analysis Platform
Developed a cardiac monitoring platform that processes continuous patient telemetry and vitals data to help identify potential cardiovascular risks. The system focused on analyzing changes in patient readings over time to improve alert accuracy and support clinical monitoring workflows.
Responsibilities:

- Clinician dashboards were structured around continuous timelines so ECG activity and heart-rate fluctuations could be interpreted over extended monitoring windows instead of isolated snapshots.
- Patient vitals were streamed into the monitoring interface in near real time through event-driven updates rather than relying on periodic polling cycles.
- Filtering capabilities were added for time ranges, anomaly categories, and severity levels, helping clinicians focus on clinically relevant segments without scanning full datasets manually.
- Telemetry ingestion services were optimized to continuously process high-frequency wearable device data while maintaining stable throughput during spikes in incoming signals from multiple concurrent monitoring devices.
- Applied sliding window evaluation to analyze heart rate variability across moving intervals instead of relying on single-point checks. This made it possible to capture irregular rhythm patterns that only appear over a sequence of readings.

- Anomaly evaluation logic considered both sudden spikes and sustained deviation from baseline readings to better distinguish natural fluctuation from clinically relevant abnormalities.
- Represented incoming vitals as time-series data, allowing efficient retrieval of both recent activity and historical trends when needed for comparison.
- Structured data ingestion and analysis as separate asynchronous flows so that incoming signals are not delayed by processing logic. This kept the system responsive even when data volume increased, reducing processing latency by ~30%.
- Added preprocessing steps to standardize incoming device data, handling differences in sampling rates and noise levels before analysis.
- Frequently accessed vitals and intermediate processing results were maintained in Redis to reduce repeated access to long-term storage during active monitoring sessions.
- Designed ingestion logic to handle intermittent device disconnections gracefully. Data received after reconnection is processed in sequence, maintaining continuity in patient records.
- Enforced role-based access to ensure only authorized personnel could view or act on sensitive patient information.
- Periodic aggregation workflows were scheduled to generate longer-duration patient summaries for trend analysis across monitoring windows.
- Deployed services on Kubernetes with scaling behavior tied to incoming data volume, allowing the system to handle fluctuations in device activity without manual intervention.
- Docker-based runtime environments were standardized across services to maintain consistency between development, testing, and deployment stages.
- Introduced structured logging and metrics to track ingestion flow, processing delays, and anomaly detection frequency. This made it easier to trace issues when system behavior deviated.
- Inter-service communication channels were protected through encrypted transport and strict API-level access enforcement for sensitive patient data.
- Took ownership of the anomaly detection flow end-to-end, from defining evaluation rules to integrating it with real-time ingestion and alert generation. Collaborated with stakeholders to refine how normal versus abnormal patterns are interpreted for different patient profiles.
- Proposed shifting from single-reading evaluation to time-window-based analysis after observing that isolated thresholds were producing noisy alerts, reducing false alert noise by ~20%.

Environment:
Java, JavaScript, ReactJS, D3.js, Chart.js, Spring Boot, Spring Security, Time-Series Processing, Sliding Window Analysis, MongoDB, PostgreSQL, Redis, AWS, Azure, Kubernetes, Storage Services, Docker, Jenkins, Prometheus, Grafana, Time-Series Modeling, Anomaly Detection, Data Normalization, Asynchronous Processing, Fault Tolerance

Lowe's Companies Inc, Charlotte, North Carolina, United States **Feb 2019 - March 2021**
Role: Java Developer
Project: Store-Based Fulfillment & Pickup (BOPIS) Platform
Contributed to Lowe's e-commerce fulfillment platform with a focus on Buy Online Pick Up In Store (BOPIS) workflows. The work involved improving store-specific inventory handling, checkout flows, and order processing to provide a smoother pickup experience for customers.
Responsibilities:

- Product availability was aligned to store-specific inventory so listing pages, cart flows, and checkout behavior consistently reflected the selected pickup location.
- Store selection state was maintained through Angular services to avoid inconsistencies when users navigated across fulfillment and checkout flows.
- RxJS stream composition was used to synchronize store selection, inventory updates, and product response handling across dynamic UI states.
- Enforced cart validation when users changed pickup locations, preventing stale or invalid items from moving forward.
- Delivery and pickup workflows were structured to allow users to switch fulfillment modes without disrupting cart progress or selected items.
- Integrated Google Maps APIs to support proximity-based store selection and reduce manual input errors.
- Improved responsiveness of large product lists by optimizing Angular change detection and trackBy usage, reducing unnecessary UI re-rendering during filtering and store switching.
- Segmented store-related workflows into lazy-loaded modules, reducing impact on initial application load.
- Extended backend services to return store-scoped inventory data, replacing global availability assumptions.
- Inventory reservation handling was introduced during order confirmation to reduce conflicts between online purchases and in-store availability.

- Handled payment retries using idempotent transaction identifiers, ensuring duplicate charges are avoided.
- Non-blocking operations such as notifications and downstream updates were routed through RabbitMQ to keep checkout workflows responsive.
- Frequently requested store-product combinations were cached to reduce repeated lookup overhead during high-traffic shopping periods.
- Optimized store-level queries through indexing and query tuning, addressing performance overhead from location-based filtering.
- Maintained transactional consistency across order placement, ensuring payment and inventory updates remain synchronized.
- Ownership: Led the shift to a store-scoped availability model, aligning frontend behavior and backend contracts to eliminate inconsistencies in pickup workflows.

Environment:
Java, Angular, TypeScript, RxJS, HTML5, CSS3, Spring Boot, Spring MVC, RabbitMQ, Oracle DB, PostgreSQL, AWS (EC2, S3, RDS), Docker, Jenkins, Log4J, Splunk

Greaves Technologies Limited, Aurangabad, India **July 2016 - Jan 2019**
Role: Software Developer
Project: Employee Management & Attendance Tracking System
Worked on an internal employee management application used for maintaining employee records, attendance tracking, and basic reporting. The project focused on simplifying administrative tasks and improving the consistency of employee data management.
Responsibilities:

- Assisted in developing basic UI screens for employee registration and profile management using HTML, CSS, and JavaScript.
- Created form handling logic to capture user inputs and perform client-side validation before submitting data to the backend.
- Contributed to backend modules using Java and Spring MVC to support CRUD operations for employee and attendance data.
- Wrote SQL queries to insert, update, and retrieve records from relational databases, ensuring data consistency across modules.
- Implemented simple server-side validations to prevent incorrect or incomplete data from being stored.
- Attendance tracking screens were enhanced to allow daily records to be captured and viewed through structured tabular layouts.
- Basic reporting functionality was added to generate employee and attendance summaries based on selected date ranges and filters.
- UI-related issues involving layout alignment and form behavior were resolved to improve consistency across browsers.
- Assisted in integrating frontend forms with backend endpoints, ensuring smooth data flow between layers.
- Debugging efforts included resolving data display inconsistencies and form submission issues across employee management workflows.
- Made minor enhancements to existing modules based on feedback from internal users.
- Collaborated with team members to understand application workflows and contribute to feature-level enhancements.
- Gained exposure to version control using Git for managing code changes and collaborating within the team.

Environment:
Java, Spring MVC, HTML, CSS, JavaScript, JDBC, MySQL, Git

Education: Master of Science, Texas A&M University

Certifications: Amazon Web Services Certified Solutions Architect – Associate (SAA-C03)