PRINCE VERMA

+91-7096277820 | princevermasrcc@gmail.com | in princevermaedignite
2500+ DSA problems solved | Leetcode | Portfolio | NPM | Book Reads

SUMMARY

Software Engineer II with 4 years of experience at OneAssist, specializing in architecting scalable Full Stack solutions. Expertise spans Java Spring Boot microservices, React.js (Server-Driven UI), and System Security, with a proven track record of migrating legacy monoliths to distributed architectures. Adept at solving complex engineering challenges—from PII Tokenization vaults to resilient media ingestion systems—while driving engineering excellence through mentorship, tooling optimization, and a relentless focus on business impact.

SKILL SET

Languages: Java 11/17, TypeScript, JavaScript (ES6+), SQL.

Frontend: React.js (Hooks, Redux/Context API), Next.js, React Native, Angular, HTML5, CSS3 (SASS/Tailwind).

Backend: Spring Boot (Microservices, Spring Security, JPA), Node.js, Express.js.

Data & Messaging: PostgreSQL (or MySQL), MongoDB, Redis, Apache Kafka.

Testing: Jest, React Testing Library, JUnit, Mockito.

Infrastructure & Tools: Docker, AWS (EC2/S3), Jenkins (CI/CD), Git, Webpack, Storybook, Postman, Kibana.

WORK EXPERIENCE

OneAssist Consumer Solutions - Software Engineer II **Dec 2021 - Present**

Key achievements and Projects

- **Server-Driven UI (SDUI) Dashboard Engine** | React.js, JSON Schema, Optimization Patterns
  Problem: The partner platform faced scalability bottlenecks due to hardcoded, partner-specific dashboards. This resulted in high code redundancy and required a full deployment cycle for minor layout changes, increasing maintenance overhead.
  Solution:
  - **Architected a Schema-Driven Rendering Engine:** Designed a recursive component mapper that ingests JSON configurations to dynamically generate UI layouts. This decoupled the frontend release cycle from backend logic.
  - **Resilient Network Layer:** Engineered a fault-tolerant data layer using Axios interceptors with exponential backoff strategies. Integrated Skeleton loading states to optimize Cumulative Layout Shift (CLS) during data fetching.
  - **Optimized State Management:** Replaced prop-drilling with a centralized reducer pattern to manage complex, nested configuration updates, ensuring 60fps rendering performance during layout shifts.

  Impact:
  - **Reduced code duplication by 40%** by unifying diverse partner requirements into a single configurable codebase.
  - **Accelerated feature deployment by 25%** and enabled hot-swapping of UI layouts without requiring engineering downtime.

- **Service Center Media Ingestion Module (Large File Architecture)** | React.js, MediaDevices API, Blob API
  Problem: Service center operations faced high liability costs due to lack of visual proof for device conditions before/after repair. The system required a reliable way to ingest high-resolution video evidence (100MB+) from diverse devices (mobile/desktop) over unstable field networks without crashing the browser.
  Solution:
  - **Resilient Chunked Upload Architecture**: Engineered a file slicing mechanism using the Blob API to break large videos into manageable chunks, bypassing server-side payload limits and preventing browser memory leaks.
  - **Fault-Tolerant Queue System**: Developed a custom upload manager that maintains a "last-successful-byte" pointer. Implemented offset-based recovery, allowing interrupted uploads to resume automatically from the last successful chunk rather than restarting, solving for network flakiness.
  - **Hardware-Agnostic Capture Interface**: Integrated the MediaDevices/WebRTC API with complex logic to detect, enumerate, and hot-switch between available video inputs (Front/Back cameras vs. External Webcams) while managing browser permission lifecycles.

Impact:
- Reduced upload failure rates by 90% on low-bandwidth networks via the resume-capability, ensuring operational continuity.
- **Eliminated false damage claims** by enabling 100% reliable capture of pre-repair device conditions, directly impacting operational cost savings.

- **Centralized PII Vault & Tokenization Service** | Java, Spring Boot, Spring Batch, AES-256

  Problem: With tightening data privacy regulations and strict security audit requirements, storing sensitive customer PII in plain text across microservices posed a significant security risk and compliance liability. The system needed a way to decouple business logic from sensitive data handling without impacting API latency.

  Solution:
  - **Architected a Tokenization Service**: Built a standalone Spring Boot security microservice that intercepts PII, encrypts it using AES-256/GCM standards, and returns opaque UUID reference tokens. This ensured that downstream services only processed tokens, keeping raw data isolated.
  - **High-Volume Data Migration**: Engineered a fault-tolerant Spring Batch pipeline to migrate legacy data. Implemented partitioning and multi-threading to encrypt millions of existing records with zero data loss, utilizing a chunk-based processing model to manage memory efficiency.
  - **Secure Key Management**: Integrated with a Key Management Service (KMS) for rotating encryption keys, ensuring that cryptographic material was never hardcoded or exposed in the codebase.

  Impact:
  - **Achieved 100% Regulatory Compliance** by pseudonymizing sensitive data, effectively removing PII from application logs and databases.
  - **Secured 100K+ customer records** via the batch migration strategy with zero downtime.
  - **Maintained <10ms latency overhead for encryption/decryption calls**, ensuring no degradation in the core checkout and claim processing flows.

Additional achievements and projects
- **Monolith-to-Microservices Migration**: Decoupled the Authentication module from a legacy monolith into a standalone Spring Security microservice. Implemented centralized OAuth/JWT authorization, enabling scalable access control across the distributed ecosystem.
- **Developer Experience (DX) & Tooling**: Spearheaded the development of "ComponentBook" (an internal npm library), standardizing React UI components and utility APIs. This reduced frontend boilerplate code by 30% and ensured design consistency across products.
- **Payment Gateway Integration**: Engineered the S2S (Server-to-Server) payment capture module, integrating with multiple gateways. Designed the system to handle transaction state consistency, directly impacting revenue collection.
- **Engineering Mentorship**: Conducted team-wide tech talks on advanced JS/Java patterns (Async/Await, Event Loop, Memory Management), improving code review quality and reducing technical debt.
- **Web Performance & SEO**: Boosted organic search traffic by 25% by transforming the application into a Progressive Web App (PWA) and optimizing Core Web Vitals (reduced LCP by 30%, CLS by 50%) via Service Workers, lazy loading, and WCAG 2.1 accessibility compliance.
- Utilized modern developer tools such as Microsoft Clarity, Cursor, Tabnine, and GitHub Copilot to improve code accuracy, productivity, and debugging efficiency.
- Recognition:
  - **Best Performer Award (2022)**: Recognized for critical contributions to the React to Angular migration project release and zero-defect delivery.
  - **Ideathon Finalist**: Ranked Top 4 (among 50+ teams) for prototyping the "Config Driven UI" engine, which was later adopted into the product roadmap.

EDUCATION

Bachelor of Technology (B.Tech.)

Sardar Vallabhbhai National Institute of Technology (NIT), Surat (2016 - 2020)

COMMUNITY CONTRIBUTIONS & KNOWLEDGE SHARING

- Sharing regular technical insights, best practices, and analysis of industry trends with 12K+ LinkedIn followers.
- Topics include frontend performance, JavaScript deep dives, accessibility practices, and architectural patterns.