Aditya Dhage

aditya.s.dhage@gmail.com | Los Angeles, California - 90018 | LinkedIn | GitHub | Portfolio

Technical Skills

Programming	-	Java, JavaScript, TypeScript, Python, SQL, NoSQL, Shell Scripts
Frontend	-	React.js, Next.js, HTML, CSS, TailwindCSS, UI/UX Design
Backend	-	Spring Boot, Node.js, Express.js, JWT, OAuth, REST APIs, GraphQL APIs
Databases	-	MySQL, MongoDB, PostgreSQL, Oracle DB, Prisma ORM
Cloud & DevOps	-	AWS, GCP, Webpack, Docker, Kubernetes, Microservices, CI/CD
Tools & Misc.	-	Git, Jenkins, Figma, Maven, NPM, Generative AI, JIRA, Agile/SCRUM

Education

Master of Science in Computer Science — *Jan 2023 - Dec 2024*

University of Southern California (USC), Los Angeles, CA

Experience

Software Engineer (Full-Stack) — *Feb 2025 - Present*

One Pacific Hub, Long Beach, California

- Built a full-stack e-commerce platform from scratch using Next.js, Node.js, and PostgreSQL, tailored for the unique logistics of an import-export business with more than 50 real customers
- Implemented core shopping features, including product catalog, dynamic cart, user management, billing, order management, and payment integrations with secure, role-based user access

Software Developer (Full-Stack) — *Aug 2018 - Nov 2022*

Hansen Technologies, Pune, MH, India

- Designed, implemented, and integrated B2B telecommunications software solutions using Java, Node, and React while diagnosing and resolving critical production issues to ensure 24/7 service availability
- Overhauled a key workflow UI microservice with React, significantly improving user experience and reducing load time to 3.5 seconds
- Implemented advanced error handling and auto-correction workflows that improved overall system reliability by 5%, reducing downtime and customer support incidents
- Led a two-person team to build prototypes and proof-of-concept features, directly helping to secure a 1-year product and services contract with a major client
- Developed a microservice to automate QA testing, cutting manual effort by 20%, shortening the delivery cycle to 4 days
- Engineered a well-documented library for OSS request translation incorporating low-level design patterns to streamline future system upgrades and reduce maintenance overhead
- Built and maintained CI/CD pipelines for Java and Node.js projects, streamlining multi-platform deployments on AWS EKS and improving deployment speed and consistency by 85%

Projects

Song-Pool: A collaborative music curator — *Mar 2025 - Jun 2025*

React.js, Node.js, Express, Socket.io, Spotify Web API, Next Auth, PostgreSQL, TailwindCSS (GitHub)

- Designed and implemented a collaborative music curation platform for ambient environments (cafes, co-working spaces), allowing patrons to vote on what music plays in real-time
- Integrated Spotify Web API for central music control, managed real-time user sessions, created real-time voting (web-sockets) and music recommendation system for ambient music curation according to the majority of patrons' taste

One-Clip: A zero-friction personal clipboard — *Feb 2025 - Apr 2025*

Next.js, React.js, MongoDB Atlas, Next Auth, TailwindCSS, Prisma (GitHub)

- Built a browser-based clipboard app with instant copy-paste functionality, rich-text support, cloud sync, integration with Google OAuth through NextAuth for secure login, protected user-specific data, and clip storage on MongoDB Atlas and Prisma ORM

Crypto Board: A real-time crypto sentiment dashboard (Academic Project) — *Nov 2024 - Dec 2024*

React, Django, MongoDB, TextBlob, Docker, Pymongo, Material UI, GCP (GitHub)

- Developed a full-stack cryptocurrency dashboard with real-time price tracking, volume metrics, and social sentiment analysis
- Scraped and parsed crypto-related text data, using TextBlob for NLP-based sentiment scoring, Automated Docker and Google Cloud deployment for frontend and backend services

Video Library Search with Video Clip Query (Academic Project) — *Dec 2023*

Python, Numpy, CV2, PyQt (GitHub)

- Developed a video library pre-processing algorithm (shot boundary detection, frame histogram calculation and hashing) in Python to streamline video data indexing and retrieval with an interactive desktop video player interface (PyQt5)
- Enabled querying the library using a short video clip as input, achieving precise frame matches with an average lookup time of 200–300 ms for a 100+ video database