Mansi Aher

Los Angeles, California, United States | (626) 861-9693 | mansiaher.work@gmail.com | LinkedIn | GitHub | Website

EDUCATION

California State University Los Angeles **Aug 2024 - May 2026**
Master of Science, Computer Science
• **GPA:** 3.9/4.0

Savitribai Phule Pune University **Aug 2020 - May 2024**
Bachelor of Engineering, Computer Engineering
• **GPA:** 3.8/4.0

WORK EXPERIENCE

California State University, Los Angeles **Aug 2025 - May 2026**
Research Assistant
• Built and maintained Python pipelines to collect and preprocess 27,000+ acoustic and pressure sensor observations from Raspberry Pi-based monitoring systems.
• Designed reproducible machine-learning experiments to benchmark Random Forest, SVM, XGBoost, and neural network classifiers using recall, F1-score, and confusion-matrix analysis.
• Collaborated with faculty researchers to analyze experimental results and translate findings into the core contributions of a master's thesis.

The Sparks Foundation **Feb 2022 - Mar 2022**
Web Development Intern
• Designed MySQL database schemas and configured least-privilege IAM access controls on Google Cloud Platform.

SKILLS

Programming Languages: Python, SQL, TypeScript, JavaScript, Java
Frameworks & Data: FastAPI, SQLAlchemy, Celery, Next.js, REST APIs, Pandas, NumPy, React.js, Node.js
Databases & Caching: PostgreSQL, pgvector, Redis, MySQL, SQLite, ChromaDB, Supabase
Tools, Testing & Cloud: Docker, Docker Compose, GitHub Actions, Git, pytest, Google Cloud Platform (GCP), AWS EC2 and S3, CI/CD
ML & AI: PyTorch, TensorFlow, scikit-learn, XGBoost, RAG, vector search, Hugging Face, machine learning

PROJECTS

DataGuard - Data Quality & Pipeline Monitoring GitHub **Oct 2026**
• Built a full-stack data quality platform with a Next.js and TypeScript frontend and FastAPI backend that previews sales CSVs, applies versioned validation rules, and processes records in batches using PostgreSQL, Celery, Redis, and Docker.
• Designed a quarantine and replay workflow that preserves original values and validation failures, lets users correct rejected records, and records replay outcomes in an audit trail.
• Implemented transactional checkpoints and database constraints so interrupted validation runs can resume while preventing duplicate accepted records.

DevPilot - AI-Powered Code Search GitHub
• Developed a full-stack AI-powered code search platform with a FastAPI backend and PostgreSQL with pgvector, enabling natural-language retrieval of GitHub source code with file-level and line-level citations.
• Implemented background repository indexing, automated code chunking, vector embeddings, and semantic similarity search to integrate AI-powered retrieval across indexed repositories.
• Built a browser-based interface with authentication, repository selection, and commit-specific GitHub citations to ensure traceable search results.
• Containerized the application with Docker to provide a reproducible local development and deployment environment.

RoleMatch - Resume-Based Job Discovery & Application Tracker GitHub **Oct 2026**
• Built a resume-based job discovery app using Next.js, Python, FastAPI, Celery, and Redis, with editable profiles from PDF, DOCX, or pasted text and qualifications traceable to their source passages.
• Integrated Greenhouse and Lever job boards and combined PostgreSQL full-text search with pgvector similarity search to rank openings by documented qualifications.
• Built an application tracker with stage history, submitted resume versions, deadlines, and calendar export.

AI Knowledge Copilot: RAG Document Q&A Platform GitHub
• Engineered a full-stack Retrieval-Augmented Generation (RAG) platform using Python, FastAPI, SQLAlchemy, and Streamlit to ingest PDFs, generate vector embeddings, and deliver document-grounded answers with page-level citations.
• Designed a modular backend with 6 REST API endpoints, separating API routes, business logic, and database repositories to support configurable SQLite and PostgreSQL persistence.
• Implemented an abstract LLM provider interface supporting interchangeable local Hugging Face models and hosted OpenAI or Anthropic APIs without modifying retrieval or prompting logic.
• Containerized backend and frontend services using Docker Compose and automated continuous integration with GitHub Actions, executing 11 pytest tests and optimizing test startup through lazy model loading.