Karan Sreedhar

Chicago, IL | karansreedhar15@gmail.com | karansreedhar.me
linkedin.com/in/karansreedhar1505 | github.com/karan1505

MS Computer Science graduate from UIC (May 2026) with experience in backend engineering, ML systems, and cloud-native infrastructure. Built and deployed production systems spanning distributed storage, RAG pipelines on AWS, and full-stack web applications. Proficient in Python, Scala, and TypeScript, with practical experience in Spark, Flink, Docker, and Kubernetes. Currently pursuing CKAD. Seeking backend, ML engineering, or DevOps roles.

Education

University of Illinois at Chicago — Chicago, IL
Master of Science in Computer Science — *Aug 2024 – May 2026*

- **Relevant Coursework:** Advanced Machine Learning, Cloud Computing, Secure Web Application Development, Responsible AI Engineering, Security and Privacy in Networked and Distributed Systems, Natural Language Processing

SRM Institute of Science and Technology — Chennai, India
Bachelor of Technology in Computer Science Engineering — *Sep 2020 – May 2024*

- **Relevant Coursework:** Artificial Intelligence, Database Management Systems, Cloud Native Architecture

Experience

Canvendor, Software Apprenticeship (Part-time) — Chicago, IL (Remote) — Jun 2025 – Aug 2025
Evaluated DevSecOps and observability tooling including SonarQube, Grafana, and OWASP ZAP, across existing codebase, producing documentation and implementation solutions for integrating security scanning and monitoring into CI/CD workflows.

Lumiq (Crisp Analytics), Summer Intern — Chennai, India — Jun 2023 – Sep 2023
Built PySpark and Pandas data pipelines and SQL queries to process health insurance analytics data for fintech clients, automating recurring data processing tasks to reduce manual effort across the team. Used the GreatExpectations library to run data quality checks across pipelines, ensuring clean and validated data at each stage of processing.

Projects

Adversarial NLP Stress-Testing for Fact Verification — *Jan 2026 – May 2026*
karansreedhar.me/project/1

- A research study to stress-test fact-checking AI models by designing misleading statements created to appear plausible, to study and analyze what makes these models fail. Worked in a team of 4.
- Extracted 500 candidate claims from ˜80,000 FEVER entries using a custom complexity scoring system to filter for the hardest, most ambiguous claims suited for counterfactual creation.
- Developed a Python GTK4 annotation tool with live local LLM inference to verify model failure in real time, ensuring each example actually fooled an LLM before being included in the final dataset.
- Created a final dataset of 2,000 claims with 552 hand-crafted counterfactuals and evaluated 7 fact-checking models on AWS EC2 (NVIDIA L40S), uncovering 64–96% adversarial failure rates and universal failure on 34.4% of attacks across all models, even when provided the correct supporting evidence.

Distributed Object Store with Erasure Coding and Integrity Verification — *Jan 2026 – May 2026*
karansreedhar.me/project/2

- A distributed storage system where files are stored in fragments across multiple nodes, with fault tolerance and fingerprinting that ensure file retrieval is guaranteed even with node failures or data corruption. Developed solo.
- Deployed the system using 6 Docker containers and a FastAPI coordinator, with a React frontend and a CLI tool for testing and evaluation.
- Implemented Reed-Solomon erasure coding in Python to achieve fault tolerance, so even in the case of 40% of nodes failing, file retrieval remains possible, reducing storage overhead by 44% over naive replication-based fault tolerance methods.
- Implemented homomorphic fingerprinting over GF(256) so each storage node could detect its own silent data corruption; if the fingerprint does not match, the coordinator skips the corrupt node and reconstructs from the remaining valid fragments.
- Verified end-to-end: after corrupting one fragment by flipping a bit and killing two fragments by terminating the nodes, the system reconstructed the original file from 3 remaining fragments, confirmed by SHA-256 match.

Medical Chatbot Strategy Comparison System — *Jan 2026 – May 2026*
karansreedhar.me/project/3

- A comparison platform evaluating 4 different LLM strategies for medical Q&A to understand which approaches perform best and under what conditions. Collaborated in a team of 5.
- The four strategies compared were: simple LLM, multi-turn LLM, simple LLM with RAG, and multi-turn RAG, with 28,000+ indexed abstracts from the PubMed dataset, using the Llama 3.3 70B API hosted by Groq, with a ChromaDB vector store for data retrieval.

- Implemented the baseline simple LLM strategy using FastAPI and LangChain, which all other strategies were benchmarked against.
- Built an automated evaluation script using Python that ran 600 benchmark sessions across all 4 strategies, measuring hallucination rate, semantic similarity, and context recall.
- The evaluation also included a hedge pre-filter that improved misclassification rate, by not factoring in cases where the model accurately reports that it does not have relevant information.
- Concluded that adding RAG and memory eliminated hallucinations entirely, but the simplest strategy delivered 3.5× better cost-performance at nearly identical answer quality, revealing that added complexity may not always pay off.

Academic Research Paper Search Tool with RAG on AWS *Sep 2025 – Dec 2025*
karansreedhar.me/project/4

- Research document search tool powered by LLMs, that automatically re-indexes when documents change, to look up information across a corpus of technical literature. Working solo.
- Built in Scala on AWS using a three-stage cloud data pipeline. The system ingests 600+ academic software engineering papers, supports natural language search across the full corpus, automatically re-indexes only changed documents on each run, and maps concept relationships across all papers.
- Designed the search engine using a Lucene HNSW vector index and Ollama embeddings, achieving ˜200ms end-to-end query latency via a REST API endpoint, replacing the need for paid external vector database services like Pinecone or Weaviate.
- Implemented a Spark-based delta indexer on AWS EMR using SHA-256 content hashing so only new or modified documents are re-embedded on each run, reducing re-indexing compute by ˜90% compared to full corpus reprocessing.
- Deployed an Apache Flink streaming pipeline on AWS EKS that processes the corpus as a stream to extract 5,000+ concept nodes and their co-occurrence relationships into Neo4j, with an LLM classifying the relationship type between each concept pair.
- Exposed the knowledge graph via a REST endpoint returning concept nodes and weighted co-occurrence edges, allowing users to query which concepts appear together and how frequently across the corpus.
- Built the entire pipeline from scratch with no external RAG frameworks, without the likes of LangChain, LlamaIndex, or a vector database SDK, every component implemented in Scala.
- Unified batch ingestion (MapReduce), incremental indexing (Spark), and stream processing (Flink) into one Scala codebase deployable locally or on AWS with a single build flag, containerized via Docker and deployed on Kubernetes.

Selective Language Modeling for Efficient LLM Training *Sep 2025 – Dec 2025*
karansreedhar.me/project/5

- Replicated key findings from a NeurIPS 2024 paper by fine-tuning TinyLlama-1.1B with LoRA on a single GPU, achieving baseline perplexity using only 50% of training tokens, demonstrating that selective training can halve compute without quality loss.
- Implemented and compared three token selection strategies (random, stochastic top-k, and deterministic top-k), finding that deterministic selection achieved faster convergence but at the cost of training stability compared to stochastic approaches.
- Tracked gradient similarity and token difficulty patterns across training runs to understand which tokens contributed most to learning and which were redundant across all three strategies.

Quizzify: Full-Stack Music Quiz Platform *Aug 2024 – Dec 2024*
karansreedhar.me/project/6

- A full-stack music quiz platform where users log in with Spotify, listen to song previews, and identify tracks from album art, worked in a team of two.
- Created 3 difficulty levels, timed rounds, a live scoreboard, and 9 curated quiz modes with a custom mode for user-generated quizzes. Deployed at quizzify.space.
- Designed the frontend using ReactJS with live audio playback, with FastAPI handling the backend and 12+ REST endpoints covering Spotify OAuth, playlist fetching, quiz generation, answer validation, and score persistence.
- Built a Playwright-based web scraper to source 30-second song preview URLs from Apple Music, necessary because Spotify's API removed audio previews during project development.
- Deployed as 3 independent cloud services: React frontend and FastAPI backend on Render, and the Apple Music scraper on Google Cloud Run, with inter-service communication via REST and MongoDB Atlas for data persistence.
- Shipped a production website with persistent scoreboards and a user-generated quiz pipeline where any Spotify playlist is automatically enriched with preview URLs on demand.

Interactive Handwritten Character Recognition System *Jan 2025 – Apr 2025*
karansreedhar.me/project/7

- Trained and compared Logistic Regression, RNN, and CNN architectures on the EMNIST dataset for handwritten character recognition, measuring accuracy and complexity tradeoffs across all three.
- Built a Python inference backend with a consistent preprocessing pipeline and a React drawing interface where users draw characters and see live model predictions.
- Deployed the full pipeline end-to-end, from model training through a working interactive demo.

Dataset Shift Analysis & Implementation *Jan 2025 – Apr 2025*
karansreedhar.me/project/8

- Implemented three types of dataset shift: covariate, prior probability, and concept shift using scikit-learn on real-world datasets to study how models fail when training and test distributions diverge.
- Simulated covariate shift by training on sunny weather conditions and testing on rainy data, showing how feature distribution changes degrade Random Forest performance in practice.
- Measured prior probability shift by varying class imbalance ratios from 50% to 90% majority class, and visualized concept shift breakdown using confusion matrices and ROC curves.

Technical Skills

Languages: Python, JavaScript, Scala, TypeScript, SQL, Bash

Web & APIs: FastAPI, ReactJS, Node.js, REST APIs, OAuth 2.0

Databases: MongoDB, PostgreSQL, MySQL, ChromaDB, Neo4j

Distributed Systems: Apache Spark, Apache Flink, Hadoop MapReduce, Parquet, HDFS

Machine Learning: PyTorch, TensorFlow, scikit-learn, Pandas, NumPy, RAG, LangChain, Ollama, LoRA Fine-tuning, LLM Training

Cloud: AWS (S3, EMR, EC2, IAM, VPC, EKS), Google Cloud (Cloud Run)

DevOps & Tools: Git, Docker, Kubernetes, CI/CD, Playwright, GTK4, OWASP ZAP, Linux, Prometheus, Grafana

Certifications & Supplemental Coursework

Cloud Computing Foundations	IBM (edX)
Containers, Kubernetes, and OpenShift	IBM (edX)
Microservices, Serverless Architecture, and OpenShift	IBM (edX)
DevOps Fundamentals and Agile Practices	IBM (edX)
Machine Learning with Apache Spark	IBM (edX)
Python for Data Science and AI Development	IBM (edX)
Red Hat Enterprise Linux Fundamentals	Red Hat (Coursera)