

SIDHARTH

Generative AI Engineer — LLM / ML Engineer

+91 9468168946 — jisidharthjain@gmail.com — linkedin.com/in/jisidharthjain — github.com/JAINSID02

Paonta Sahib, Sirmaur, Himachal Pradesh, India — Open to Relocation

Professional Summary

B.Tech Computer Science student at Jaypee University of Information Technology (JUIT) and IBM Certified Generative AI Engineer (16-course Professional Certificate, 2026) with hands-on, from-scratch experience across the full modern LLM stack: transformer architecture design, pretraining, supervised fine-tuning, reward modeling, and RLHF (PPO), alongside production Agentic AI systems using Self-RAG and multi-agent orchestration. Proficient in Python, PyTorch, LangChain, LangGraph, Mistral AI, transformer fine-tuning (LoRA, PEFT, RLHF), and Retrieval-Augmented Generation (RAG). Strong foundation in building, debugging, and deploying end-to-end Generative AI systems from first principles rather than only via pre-built APIs.

Technical Skills

- **Languages:** Python, SQL
- **Deep Learning Frameworks:** PyTorch, TensorFlow, Keras, Scikit-learn
- **AI , ML & DL Concepts:** Machine Learning, Deep Learning, Transfer Learning, Neural Networks
- **LLM Architecture (built from scratch):** Multi-Head Self-Attention, RoPE, RMSNorm, SwiGLU, KV-Caching, Sliding-Window Attention with Attention Sink, Mixture-of-Experts (Top- k Gating, Load-Balancing Loss), Custom BPE Tokenization
- **LLM Training & Alignment:** Base Pretraining, Supervised Fine-Tuning (SFT), Reward Modeling (Bradley–Terry Loss), RLHF (PPO), LoRA, QLoRA, PEFT, Mixed-Precision Training, Gradient Accumulation, LR Warmup Scheduling
- **LLM Ecosystem:** Hugging Face (Transformers, Datasets), Prompt Engineering, LangChain, LangGraph, LangSmith, Retrieval-Augmented Generation (RAG), Multi-Agent Systems
- **Data Science:** NumPy, Pandas, Matplotlib, Seaborn, Optuna
- **Engineering Practices:** Unit Testing (pytest), Experiment Tracking (TensorBoard), Checkpointing & Resume Support, Model Debugging & Failure-Mode Diagnosis
- **Tools & DevOps:** Docker, Git, GitHub, Streamlit, FastAPI, Pydantic
- **Databases:** ChromaDB (Vector DB), FAISS (Vector DB)

Projects

LLM From Scratch — Full Pretraining-to-RLHF Pipeline | Python, PyTorch, Custom BPE Tokenizer, Streamlit
July 2026

- Built a complete decoder-only transformer LLM pipeline from first principles in PyTorch spanning architecture, tokenization, base pretraining, supervised fine-tuning, reward modeling, and RLHF, with no dependence on Hugging Face Trainer or other pre-built training frameworks; validated with a unit-test suite (pytest) covering every module.
- Implemented a modern transformer architecture (RoPE positional embeddings, RMSNorm, SwiGLU activation, KV-caching, sliding-window attention with attention sink) and trained a custom Byte-Pair Encoding tokenizer (8,000-token vocabulary) for a 6-layer, 8-head, 384-dim model with a 256-token context window.
- Designed and implemented a Mixture-of-Experts (MoE) layer with top- k token routing and Switch-style load-balancing auxiliary loss, integrated into a hybrid dense/MoE architecture as an extension beyond the core dense model.
- Pretrained the base model on a custom ~5M-token corpus using mixed-precision training, gradient accumulation, and LR warmup with checkpointing/resume support; diagnosed a verbatim-memorization bug by tracing repeated corpus exposure across training steps and fixed it via corrected epoch-step calibration and dropout regularization.
- Fine-tuned the base model on instruction and custom Q&A data (causal LM loss with prompt-masked labels), trained a transformer-based reward model using Bradley–Terry pairwise preference loss, and aligned the policy via clipped PPO objective with a value head, KL-penalty, entropy regularization, reward normalization, and gradient clipping.
- Diagnosed and resolved a PPO mode-collapse failure where the policy converged to a single generic response — a classic reward-hacking failure mode — by isolating the cause to low-temperature rollout sampling and restoring response diversity via entropy bonuses and retuned KL constraints.
- Deployed a live Streamlit chat demo of the fine-tuned model answering questions about the author's background and projects, plus a terminal chat interface (`chat.py`) with token-by-token streaming generation across checkpoints.
- GitHub: github.com/JAINSID02/ADVANCED_LLM

- Built an end-to-end AI assistant processing uploaded video/audio files and YouTube URLs, normalizing audio to 16kHz mono and chunking recordings into 10-minute segments before transcription.
- Implemented dual-engine transcription: local OpenAI Whisper for English audio and Sarvam AI for Hinglish audio (auto-translated to English), with automatic sub-chunking into 25-second pieces to satisfy Sarvam's 30-second API limit.
- Built a map-reduce summarization pipeline (3,000-character chunks, 200-character overlap) using Mistral AI to generate a meeting title, concise summary, and structured action items, key decisions, and open questions from long-form transcripts.
- Built a transcript-aware RAG Q&A system using LangChain, ChromaDB, and MiniLM (all-MiniLM-L6-v2) embeddings, retrieving the top-4 most relevant 500-character chunks (50-char overlap) per query to ground responses in the source transcript.
- Delivered an interactive Streamlit frontend enabling users to upload files or paste YouTube links and query video content conversationally.
- GitHub: github.com/JAINSID02/AI-VIDEO-ASSISTANT

MAARS — Multi-Agent AI Research System | Python, LangChain, Mistral AI, Tavily, BeautifulSoup June 2026

- Designed a fully autonomous multi-agent research pipeline via LangChain: a Search Agent (Tavily API, top-5 results) and a Reader Agent that scrapes and cleans the most relevant source (BeautifulSoup, 8s timeout, HTML stripped of scripts/nav/footers) for deeper context.
- Engineered an LLM-based Writer Agent that synthesizes search results and scraped content into a structured report (Introduction, minimum 3 explained Key Findings, Conclusion, cited Sources) using Mistral AI.
- Built a Critic Agent that independently scores the generated report on a 0–10 scale with structured strengths, weaknesses, and a one-line verdict, enabling quality evaluation separate from generation.
- Delivered a monochrome, editorial-style Streamlit frontend with persisted research history, compressing a multi-step manual research workflow into a single autonomous query-to-report pipeline.
- GitHub: github.com/JAINSID02/MULTI-AGENT-AI-RESEARCH-SYSTEM-MAARS

Self-RAG AI Agent | Python, LangGraph, LangChain, Mistral AI, FAISS

June 2026

- Built a Self-RAG inspired document Q&A system with conditional LangGraph workflows choosing between internal reasoning and semantic PDF retrieval (FAISS, `mistral-embed`, top-4 retrieval) across a multi-document PDF knowledge base.
- Implemented 5 structured LLM-as-a-Judge nodes (retrieval necessity, relevance grading, 3-tier groundedness, usefulness, query rewriting) with two bounded self-correction loops (10-retry groundedness fix, 3-try query rewrite).
- GitHub: github.com/JAINSID02/self_rag

Education**Jaypee University of Information Technology**

Bachelor of Technology in Computer Science and Engineering

Solan, Himachal Pradesh

Aug 2023 – July 2027

- CGPA: 7.28 / 10
- Relevant Coursework: Machine Learning, Deep Learning, Data Structures & Algorithms, Database Management Systems, Operating Systems, Computer Networks

Guru Nanak Mission Public School

Senior Secondary (Class XII) — Science Stream

Sirmaur, Himachal Pradesh

Completed May 2023

Certifications**IBM Generative AI Engineering Professional Certificate** | IBM / Coursera

July 2026

- Completed a rigorous 16-course professional certification covering the full Generative AI engineering stack: foundational LLM theory, transformer architectures, fine-tuning, and production deployment.
- Key areas covered: LLM Architecture & Data Preparation, Language Modeling with Transformers, Fine-Tuning & PEFT/LoRA, Advanced LLM Fine-Tuning, AI Agents with RAG & LangChain, Deep Learning with Keras, Building Generative AI Applications with Python.
- Credential: [Verify Credential Here](#)