Harsh Mankar

LLM PROJECTS AND EXPERIENCE

Certifications: Microsoft Azure Certified: DP-100 - Azure Data Scientist Associate (Issued: June 2025)

GridLLM: Agentic AI System for California Electricity Market Operations: Github

- I built a specialized agent system for the **California Energy Grid (CAISO)**. I used **LangGraph** to orchestrate a multi-agent system assigning distinct roles like Price Analysis, Incident Response, and Anomaly Detection because a single prompt simply couldn't handle the complexity of 400+ regulatory protocols.
- Generic models failed to understand energy jargon or numerical trends. I solved this with a two-step training process: **DAPT (Domain-Adaptive Pre-Training)**: I fine-tuned Llama 3.2-3B on a 32MB text corpus of technical PDFs and market logs using **QLoRA** to teach it the vocabulary. **Instruction Tuning:** I converted raw CSV numerical data into a **"Q&A"** format to fine-tune the agent's prompts, optimizing it to run efficiently on a single **free-tier T4 GPU**. I implemented a Hybrid Architecture because I didn't trust the LLM to do math. I ran traditional **Association Rule Mining** on 2 years of raw **time-series data** to mathematically discover hard pricing patterns (e.g., "If Load > 40GW, Price spikes"). I fed these discovered rules into the agents' reasoning loop via the **Model Context Protocol (MCP)**, giving the AI "ground truth" logic to reference and preventing hallucination.

The Application Pilot (Custom RPA For Greenhouse & Lever recruiting platform) : Github

- Built a Python automation engine for automated job application using just a URL & architected a full pipeline using **Playwright** because it handles dynamic JavaScript sites way better than Selenium. Relying purely on AI is slow and expensive. I actually tried building a **'Vision-Based' version** first using **Gemini 1.5 Pro** to **look at page screenshots and decide where to click** but the **latency was too high** and the free tier hit limits too fast. So, I pivoted to a hybrid architecture that is much faster: **Dumb Logic: For detection and H1B checks**, I use deterministic regex because it's instant. **Smart Logic:** I integrated **Groq API (Llama-3.3-70b-versatile) for decision-making**. I uploaded 6 different versions of my resume (Backend, AI, Data Science, etc.). The system scans the Job Description, matches keywords, and intelligently selects the single best PDF for that specific role. It even generates **custom answers for essay questions** like 'Why do you want to work here?' in real-time. Built a 'Stealth Mode' to bypass bot blockers and wrote custom input handlers for React forms, automatically switching to **DOM injection** if standard typing fails due to React state lag. My next step is to wrap this in a **Telegram API**, so I can just **text a job link to my server and the agent will apply for me while I'm away from my keyboard.**

AI ENGINEER **September 2025 - November 2025**
Lorelei Molinari Home of Mental Health Info & Support Groups **PA, USA**

- Real-Time Voice Architecture (RAG): I engineered Voice AI Agent using **Azure OpenAI** to handle crisis response queries. The core challenge was eliminating the standard 3-4 second latency of LLMs, which is unacceptable in emergency scenarios. I implemented token streaming, pushing text chunks to the Text-to-Speech (TTS) engine the moment they were generated, reducing response latency to <800ms. I integrated Twilio (telephony) with Azure Speech Services to handle the audio stream, ensuring the system could handle interruptions . Reliability: To prevent high-stakes hallucinations, I built a **RAG (Retrieval-Augmented Generation)** pipeline that grounded the model's responses in a validated database of 10,000+ emergency records all over the globe. I used **System-Level Prompt Engineering** as a logic gate, strictly constraining the agent's vocabulary during critical workflows.

BACKEND ENGINEER **November 2025 - Present**
Zenzie Corp **NJ, USA**

- Scalability & Latency Optimization: I architected the backend migration to Go microservices to handle high-concurrency user requests. The Mechanics: I focused on the data layer, implementing Redis caching to store frequently accessed user sessions, which reduced content delivery latency . I containerized these services using Docker on AWS (EC2), creating a CI/CD workflow that allowed for safe, iterative deployment of new features.

Reddit TIFU Summarization (BART & Pegasus): Github

- I fine-tuned Facebook/BART and Google/Pegasus models on a dataset of **124k records** to build an automated summarization pipeline. The engineering focus was on metric evaluation; I benchmarked models using ROUGE scores (Pegasus achieved ROUGE-L: 55.5%) to select the optimal architecture for summarizing long-form unstructured text into executive briefs summarizing long-form unstructured text into **concise executive briefs (TLDRs)**.

Cost-Efficient Sentiment Classifier (DistilBERT): Github

- Instead of using expensive LLMs for simple classification, I fine-tuned a lightweight **DistilBERT** model on 10k reviews, achieving **87.1% accuracy**. I implemented a custom **threshold tuning** layer in PyTorch to specifically reduce false positives by 12%, ensuring that only high-confidence negative feedback triggered alerts.