MOHIT RAJ

India | mohitraj3697@gmail.com | +91 9334307945 | github.com/mohitraj3697 | mohitrajdev.in

SUMMARY

Software Engineer and AI Developer focused on LLM systems — LangGraph agent pipelines, RAG architectures, PyTorch model training, and full-stack AI applications. Builds from scratch when needed (tokenizers, attention, training loops) and ships production-ready systems when it counts.

TECHNICAL SKILLS

Languages: Python, JavaScript, TypeScript, HTML, CSS

Frameworks and Libraries: PyTorch, FastAPI, LangGraph, LangChain, Hugging Face Transformers, Next.js, React (Vite), Streamlit, Pydantic

AI and ML: LLMs, Generative AI, RAG, Transformer Architecture, Pretraining, Supervised Fine-Tuning (SFT), Transfer Learning, Agentic AI, HITL, Prompt Injection Defense

Tools and Databases: FAISS, SQLite, SqliteSaver, Groq API, OpenAI Embeddings (text-embedding-3-small), MCP, Uvicorn, Vite, ExchangeRate-API

PROJECTS

Coding LLM From Scratch | *Python, PyTorch*
https://github.com/mohitraj3697/Coding_LLM_End2End

- Implemented every component independently — custom BPE tokenizer, DataLoader, multi-head self-attention with causal masking, full transformer block (LayerNorm, FFN, residuals, positional embeddings), and pretraining loop. Zero HuggingFace Trainer or pre-built blocks.

LangGraph Pipelines | *Python, LangGraph, Streamlit*
https://github.com/mohitraj3697/langgraph_pipelines

- Reference library of 8+ LangGraph patterns (sequential, conditional, parallel, subgraph, persistence, RAG, tool-use) with fully typed state management and a Streamlit frontend — built to understand the full design space, not just one workflow.

NLP to App Compiler | *Python, LangGraph, FastAPI, React (Vite), Groq, Pydantic*
https://github.com/mohitraj3697/nlp_to_app_compiler

- 5-node StateGraph (intent → design → schema → validate → repair) where every node reads/writes a shared AppState TypedDict — no global state, no side effects between stages.
- AppSchema(BaseModel) catches exact field-level ValidationError; add_conditional_edges routes to repair where the model sees its own broken output plus the error — not a blind retry, a surgical fix. clean_json_output() strips markdown fences before json.loads().
- FastAPI backend (POST /api/generate) + React/Vite frontend with four tabbed JSON views. Groq openai/gpt-oss-20b at temperature 0.5.

Cognitive Routing RAG | *Python, LangGraph, FAISS, LangChain, OpenAI Embeddings, Groq*
https://github.com/mohitraj3697/cognitive_routing_rag

- Phase 1 (phase1_router.py): three bot personas embedded at startup into a FAISS in-memory index via text-embedding-3-small; cosine similarity computed with numpy — more reliable than raw FAISS L2. Only bots above threshold engage.
- Phase 2 (phase2_content_engine.py): 3-node LangGraph state machine (decide_search_node → web_search_node → draft_post_node) generating 280-char opinionated posts as strict JSON {bot_id, topic, post_content}.
- Phase 3 (phase3_combat_engine.py): full-thread RAG context (PARENT POST → COMMENT blocks → HUMAN INPUT) with system-level prompt injection defense — injection keywords trigger intensified in-persona responses, not compliance. Checkpointing via SqliteSaver to chatbot.db.

Pretraining Gemma-3 SLM | *Python, PyTorch*

- Translated Google's Gemma-3 technical paper into a working PyTorch implementation: Grouped-Query Attention (GQA), Rotary Positional Embeddings (RoPE), GeGLU activations, pre-normalization, full pretraining loop.

NeuroChat Engine | *Python, LangGraph, SQLite, FAISS, Streamlit, MCP*
https://github.com/mohitraj3697/neuroChat_engine

- Production-style chatbot: LangGraph orchestration, SQLite thread persistence, dual-layer memory (short-term context + long-term FAISS semantic store), MCP tool-calling, streaming via Streamlit, HITL checkpoints.

Instruction Training Pipeline | *Python, PyTorch*
https://github.com/mohitraj3697/instructiontraining_pipeline

- End-to-end SFT pipeline: instruction-response pair formatting, response-token-only loss masking, custom training loop with LR scheduling and gradient accumulation, checkpoint saving and evaluation.

Text Classification Training Pipeline | *Python, PyTorch*
https://github.com/mohitraj3697/classification_training_pipeline

- Transfer learning on a pretrained transformer: replaced LM head with a classification head, built custom DataLoaders and training loop with cross-entropy loss and per-epoch accuracy tracking — no Trainer APIs.

Ask YouTube — RAG Application | *Python, Next.js, React 19, LangChain, FAISS, Groq*
https://github.com/mohitraj3697/ask_youtube

- Ingests YouTube transcripts, chunks and embeds via BAAI/bge-small-en into FAISS, answers questions via a LangChain RAG chain backed by Groq (Llama-3). Next.js 16 + React 19 frontend.

fxgenie — LLM-Powered Currency Converter | *Python, FastAPI, React (Vite), LangChain, Groq*
https://github.com/mohitraj3697/fxgenie

- Natural language input ('10 USD to INR') → Groq Llama-3.3-70b extracts structured intent → ExchangeRate-API fetch → result to React/Vite UI via FastAPI. LLM tool-use in a real product.

GenAI Systems with LangChain | *Python, LangChain*
https://github.com/mohitraj3697/genai_systems_langchain

- Modular LLM systems using LCEL: versioned prompt templates, Pydantic output parsers, tool registration for agent use, conversation memory injection — composable components, not one-off scripts.

Custom Runnable Pipeline | *Python*
https://github.com/mohitraj3697/custom_runnable_pipeline

- Rebuilt LangChain's Runnable abstraction from scratch — base Runnable with invoke() and pipe (|) operator, plus subclasses for prompt templates, LLM callers, and output parsers. Pure Python, zero dependencies.

HuggingFace GPT Weight Mapping | *Python, PyTorch*
https://github.com/mohitraj3697/huggingface-gpt-weight-mapping

- Manually mapped HuggingFace GPT state dict into a custom architecture via name-based mapping with shape verification; added temperature scaling and top-k sampling for inference control.

Quicksync — Real-Time Collaboration App | *TypeScript, Next.js, Node.js*
https://github.com/mohitraj3697/quicksync

- Real-time multi-user sync app — Next.js/TypeScript frontend, dedicated Node.js server with event-driven architecture for instant state propagation across clients.

CERTIFICATIONS

Oracle Cloud Infrastructure 2025 Certified Generative AI Professional — Oracle
Credential: Oracle Badge Verification Link

EDUCATION

Bachelor of Technology (B.Tech), Computer Science and Engineering
Birla Institute of Technology (BIT) Mesra, Ranchi | Expected 2028

- Currently in 4th Semester (2nd Year), undertaking research and working on an academic thesis.