8-Day Training Plans Overview

8-Day Python Fullstack Training Plan

Complete Fullstack Development in 24 Hours
Duration: 8 Days × 3 hours daily (1 hour theory + 2 hours practical) | Project: Job Application Tracking Dashboard

Day	Focus Area	Theory (1 Hour)	Practical (2 Hours)	Project & Homework
1	WSL/Ubuntu Setup	WSL architecture, Ubuntu benefits, file systems, APT package management	Install WSL/Ubuntu, update packages, navigate directories, init Git repo	Repo setup for Job Dashboard; Practice commands on sample files
2	Linux Admin	User management, processes, networking, security, scripting basics	Configure permissions, monitor processes, set up firewall/SSH, write backup scripts	Secure project dirs and automate backups; Create cron job for data sync
3	Python & UV	Python versions, virtual environments, UV for dependencies, basic syntax	Install Python 3.12, create UV venv, add packages, write scripts	Initialize UV for dashboard deps, build CLI for job input; Add error handling
4	Python OOP & FastAPI	OOP concepts, modules, FastAPI setup, path operations, models	Define classes, install FastAPI, build hello world API with params	Create Job class and root endpoint; Add query params to routes
5	FastAPI Auth & Testing	Query/body validation, JWT auth, dependencies, pytest testing	Implement GET/POST routes, basic auth, write/run tests	Secure job CRUD endpoints; Test user registration
6	FastAPI & Frontend	Background tasks, middleware, Jinja2 templates for UI basics	Add async tasks, render templates, integrate simple HTML/CSS	Handle applications async, build listings UI; Implement search in UI
7	PostgreSQL & ORM	PostgreSQL basics, SQL queries, SQLAlchemy ORM, migrations	Install DB, create schemas, connect via psycopg2, define models	Set up jobs tables, integrate with API; Build search queries

Continued on next page

Table 1 – Continued from previous page

Day	Focus Area	Theory (1 Hour)	Practical (2 Hours)	Project & Homework
8	Project Assembly & Deployment	End-to-end CRUD, analytics, Docker/NGINX deployment best practices	Complete flows, add stats, containerize and deploy locally	Finalize and present dashboard; Explore cloud deployment options

8-Day Training Timeline

- Days 1-2: Environment & Database - WSL2, Ubuntu, Python, PostgreSQL, FastAPI, SQLAlchemy, Project setup
- Days 3-4: Authentication & API - JWT security, User management, CRUD operations, Data validation
- Days 5-6: Frontend Development - HTML5, CSS3, JavaScript ES6+, Responsive design, API integration
- Days 7-8: Advanced & Deployment - Testing, Analytics, Search/Filter, Docker, CI/CD, Production setup

Technology Stack

Backend	Frontend
Python 3.12, FastAPI, SQLAlchemy, PostgreSQL, JWT	HTML5, CSS3, JavaScript ES6+, Respons

Project Progression

Day 1	Day 2	Day 3
Development environment ready	Database schema designed, API skeleton	User authentication sys
Day 5	**Day 6**	**Day 7**
Responsive dashboard UI	Frontend-backend integration	Advanced features & a

Learning Outcomes

- Built a complete fullstack Job Dashboard application
- Mastered modern development technologies and workflows
- Implemented secure authentication and authorization
- Created responsive frontend interfaces
- Deployed a production-ready application
- Gained experience with database design and management
- Learned containerization with Docker
- Developed comprehensive testing strategies

GenAI Intensive Training Plan

8 Days, 3 Hours Daily, Complete GenAI Coverage

Day	Theory Concepts	Practical Concepts	Tools & Technologies
1	What is Generative AI? LLM architecture basics, Key buzzwords explained, AI ecosystem overview	AI Tools Setup	Google AI Studio, Notebook LM, ChatPDF, Julius AI, DeepWiki
2	Transformer architecture, Attention mechanisms, Model parameters & scaling, GPT vs Llama vs Qwen	OpenRouter Platform	OpenRouter API, Multiple LLM models, Cost analysis tools
3	Local vs cloud tradeoffs, Hardware requirements, Model quantization, Performance optimization	Local LLM Setup	Ollama, LM Studio, Local models, Performance profiling
4	API integration patterns, Frontend-backend communication, Real-time messaging, Error handling	Build Chat Application	FastAPI, JavaScript, HTML/CSS, Dual-model system
5	Prompt structuring techniques, Few-shot learning, System message crafting, Temperature tuning	Advanced Prompting	Prompt templates, Testing frameworks, Quality metrics
6	Model hub exploration, Pipeline usage, Custom model loading, Community models	Framework Integration	HuggingFace Transformers, PyTorch, TensorFlow, Datasets

Continued on next page

Table 2 – Continued from previous page

Day	Theory Concepts	Practical Concepts	Tools & Technologies
7	Pre-training vs fine-tuning, LoRA (Low-Rank Adaptation), QLoRA for memory efficiency, Training data requirements	Unsloth Fine-Tuning	Unsloth, LoRA, QLoRA, Training scripts, Evaluation tools
8	Multi-model systems, Performance optimization, Cost management, Future trends	Project Completion	Full stack development, Deployment tools, Documentation

Daily Learning Outcomes

- Day 1-4: AI tools proficiency, GenAI fundamentals, OpenRouter mastery, Model comparison, Local deployment, Performance optimization
- Day 5-8: Fullstack development, API integration, Prompt engineering, Quality optimization, HuggingFace ecosystem, Framework integration, Fine-tuning expertise, Unsloth optimization, System integration, Production readiness

Technology Stack Mastery

- AI Tools & Platforms: Google AI Studio, Notebook LM, ChatPDF, Julius AI, DeepWiki, OpenRouter, HuggingFace, Ollama, LM Studio, Unsloth, LoRA/QLoRA
- Development Frameworks: PyTorch, TensorFlow, FastAPI, JavaScript, HTML/CSS
- Core Concepts: LLM architectures, Transformer networks, Prompt engineering, Fine-tuning, Local deployment, Cloud integration, Performance optimization, Cost management

Final Project: Smart Chat Application

- Features Delivered: Dual-model support, Advanced prompt engineering, Custom fine-tuned models with Unsloth, Professional user interface, Error handling & performance optimization, Complete documentation
- Skills Demonstrated: AI tool evaluation and integration, Fullstack application development, Model training and optimization, Production deployment readiness, Cost-performance tradeoff analysis

Progressive Learning Path

- Week 1 (Days 1-4): Foundation & Building - Tools, Architectures, Deployment, Integration

- Week 2 (Days 5-8): Enhancement & Mastery - Prompting, Customization, Fine-tuning, Production

8-Day RAG Systems Training Plan

Daily Schedule: 3 hours per day (1 hour theory, 2 hours practicals)

Day	Focus Area	Theory Component	Practical Component	Milestone
1	RAG Fundamentals & Setup	RAG architecture, benefits over pure LLMs; intro to vectorization and embeddings	Install LangChain/LlamaIndex; load sample text data, generate basic embeddings	Initialize repo for multimodal Q&A app and read RAG guide
2	Embeddings & Vectorization Tools	Semantic vectors, models (OpenAI, Hugging Face); similarity metrics	Embed text, compute cosine similarities using FAISS	Embed sample documents and compare embedding models
3	Chunking & Indexing Strategies	Chunking methods (fixed, semantic); indexing with ANN in vector DBs	Implement chunkers in LangChain, build/index FAISS store	Chunk and index project docs and test chunk overlaps
4	Memory Concepts in RAG	Context persistence, hybrid search; memory frameworks like MemoRAG	Add conversation memory to basic pipeline	Integrate memory for multi-turn queries and explore hybrid search
5	RAG Frameworks Overview	LangChain vs. LlamaIndex; orchestration for retrieval/generation	Build simple RAG chain in both frameworks	Adapt project to LlamaIndex and compare framework pros/cons
6	Advanced RAG Tools & Optimization	Vector DBs (Pinecone, Qdrant); evaluation metrics, re-ranking	Tune pipelines, evaluate with metrics like MRR	Optimize retrieval in app and add re-ranking module
7	Multimodal RAG Concepts	Handling images/text; vision models (CLIP, Granite); multimodal embeddings	Embed images/text, build multimodal retriever with Docling/Granite	Add image handling to project and test on mixed-media docs
8	Multimodal RAG Integration & Deployment	Scaling multimodal systems; real-world applications, deployment basics	Assemble full pipeline, deploy locally with Streamlit	Complete multimodal Q&A app demo and extend to audio if time allows

Key Considerations for Implementation

- Research suggests RAG systems can improve AI accuracy by 20-30% in domains like document analysis

- Requires careful handling of data modalities to avoid mismatches
- Start with text-focused days before multimodal to build foundations
- Use open-source tools to keep it accessible (e.g., Hugging Face for embeddings)

Expected Outcomes and Adaptations

- By Day 8, participants should deploy a functional app querying text and images
- For faster learners: incorporate audio in extensions
- For slower pace: extend with additional tutorials

8-Day Training Plan: Building Multi-Agent Systems with Agentic AI

A comprehensive guide to developing advanced AI agent systems

Day	Focus Area	Theory (1 Hour)	Practical (2 Hours)	Milestone & Homework
1	Concepts & Architecture	Agentic AI basics, architectures (single/multi-agent)	Set up environment, build simple agent	Init repo for multi-agent system; Explore architectures
2	Tools & Frameworks	LangChain, AutoGen, CrewAI overviews	Implement basic agents in frameworks	Add Planning Agent; Compare frameworks
3	Memory Management	Short/mid/long-term memory, MemoryOS	Integrate memory in agents using Redis/LangGraph	Add memory to Planning Agent; Test persistence
4	Multimodal Design Patterns	Reflection, Tool Use, multimodal patterns	Build multimodal agent with image/text handling	Integrate Image Agent; Apply patterns to examples
5	Agentic AI RAG	RAG evolution, agentic integrations	Build Agentic RAG pipeline	Add RAG to Web Search Agent; Optimize retrieval
6	Reinforcement Learning	RL for agent training, MDPs, Agent Lightning	Train agent with RL simulations	Apply RL to Coding Agent; Tune rewards
7	Multi-Agent Orchestration	Coordination, examples (Data Analytics, Text-to-Image)	Orchestrate agents in CrewAI/AutoGen	Build full orchestration; Add Reflection Agent
8	Advanced Agents & Deployment	Additional agents (Tool Use), scaling	Deploy system, test multimodal/RL features	Final demo; Explore extensions

Specialized Agents

Agent Type	Description	Key Frameworks/Tools
Planning Agent	Decomposes goals into steps	LangGraph, AutoGen
Web Search Agent	Retrieves online data	LangChain integrations
Data Analytics Agent	Processes and analyzes data	CrewAI, LlamaIndex
Image Agent	Handles visual data	Granite models, CLIP
Text-to-Image Agent	Generates visuals from text	Stable Diffusion via Hugging Face
Coding/Programming Agent	Writes and executes code	AutoGen, OpenAI SDK
Reflection Agent	Self-evaluates performance	Semantic Kernel
Tool Use Agent	Invokes external APIs	Agno, PydanticAI

Next Steps & Resources

- Explore advanced agent architectures and patterns
- Implement real-world multi-agent applications
- Contribute to open-source agent frameworks
- Stay updated with latest research in Agentic AI

Additional Resources:

- LangChain Documentation: https://python.langchain.com/
- AutoGen GitHub: https://github.com/microsoft/autogen
- CrewAI Documentation: https://docs.crewai.com/
- Redis for AI Agents: https://redis.io/solutions/use-cases/ai/