GenAI Developer Roadmap by Perplexity

Phase 1 – Data Foundation (Day 1–12)

Day 1 – AI Ecosystem 2026 + Setup

🎯 Goal: Understand today's AI landscape (reasoning models vs legacy LLMs) and set up your development environment.

📚 Learn (Links):

- Blog/video: "Reasoning models (o1-style) vs traditional LLMs (GPT, Gemini, etc.)".
- Overview of Google Gemini models and use-cases.

💻 Hands-on:

- Install Python (3.10+), VS Code, Git.
- Set up virtual environment + Poetry (or pip) and a devcontainer (optional).
- Create a mono-repo for the entire 90-day journey:
    - 01_data_foundation/
    - 02_nlp_agents/
    - 03_rag_systems/
    - 04_production/

🛠️ Project Work:

- Initialize GitHub repo: genai-engineer-90days.
- Add top-level README.md explaining your background, 4-year gap as "full-time upskilling", and this 90-day plan.

🧠 Notes:

- Write 5–7 bullet points summarizing differences between classic ML, old LLMs, and modern reasoning models.

📌 Daily Output:

- GitHub repo with initial structure + README.

⏱️ Time Spent:

- ~8 hours.

Day 2 – NumPy for Transformers

🎯 Goal: Refresh NumPy and connect matrix operations to attention.

📚 Learn (Links):

- NumPy quickstart (arrays, broadcasting, matrix multiplication).
- Article or video explaining dot product and softmax in attention.

💻 Hands-on:

- Implement:
  - Dot product between query and key vectors.
  - Softmax manually.
  - Simple scaled dot-product attention for tiny vectors.

🛠️ Project Work:

- Notebook: 01_data_foundation/attention_from_numpy.ipynb with examples and plots.

🧠 Notes:

- Write the intuition of softmax and why attention uses dot products.

📌 Daily Output:

- Notebook pushed to GitHub.

⏱️ Time Spent:

- ~8 hours.

Day 3 – Pandas for LLM Data

🎯 Goal: Become comfortable with CSV, JSONL, Parquet used in LLM workflows.

📚 Learn (Links):

- Pandas docs: read_csv, read_json, to_parquet.
- Short article on JSONL format for LLM training.

💻 Hands-on:

- Download any text dataset (reviews, QA, etc.).
- Load CSV → clean columns → save as JSONL and Parquet.

🛠️ Project Work:

- Script: 01_data_foundation/preprocess_dataset.py which:
  - Cleans raw CSV.
  - Outputs clean.jsonl and clean.parquet.

🧠 Notes:

- Write when you'd choose JSONL vs Parquet in a GenAI pipeline.

📌 Daily Output:

- Script + small report in Markdown.

⏱️ Time Spent:

- ~8 hours.

Day 4 – EDA & Data Quality for GenAI

🎯 Goal: Perform EDA with focus on bias, poisoning, and quality.

📚 Learn (Links):

- Article: "Data quality and bias in LLM datasets".
- Blog: "Data poisoning basics in AI".

💻 Hands-on:

- Use the dataset from Day 3.
- Check:
  - Class distribution, duplicates, missing values.
  - Simple toxicity or profanity rate (using any open library).

🛠️ Project Work:

- Notebook: 01_data_foundation/eda_for_llm.ipynb with charts and textual conclusions.

🧠 Notes:

- List possible risks (e.g., bias, leaking PII) and mitigation strategies.

📌 Daily Output:

- EDA notebook + short summary in README.

⏱️ Time Spent:

- ~8 hours.

Day 5 – Probability & Sampling (Temperature, Top-p)

🎯 Goal: Connect probability distributions to how LLMs sample tokens.

📚 Learn (Links):

- Short article on temperature/top-p/top-k sampling.
- Simple explanation of “stochastic parrots” idea.

💻 Hands-on:

- Implement:
  - Categorical sampling from a probability vector.
  - Temperature scaling on logits, plus top-k and top-p filtering.

🛠️ Project Work:

- Notebook: 01_data_foundation/sampling_strategies.ipynb comparing outputs for different settings.

🧠 Notes:

- Write 3 bullet rules: when to increase/decrease temperature and top-p.

📌 Daily Output:

- Notebook + function library sampling_utils.py.

⏱️ Time Spent:

- ~8 hours.

Day 6 – ML vs LLM (Cost vs Accuracy)

🎯 Goal: Understand when a standard ML model is better than an LLM.

📚 Learn (Links):

- Blog: “When to use XGBoost vs LLM in production”.
- Quick recap on RandomForest or XGBoost.

💻 Hands-on:

- Pick a tabular classification dataset.
- Train a baseline RandomForest/XGBoost model.
- Estimate cost if you used an LLM API instead.

🛠️ Project Work:

- Markdown note: compare model metrics, latency, and cost vs hypothetical LLM calls.

🧠 Notes:

- Write 3 real-world examples where classic ML wins over LLM.

📌 Daily Output:

- Notebook + comparison note.

⏱️ Time Spent:

- ~8 hours.

Day 7 – Scikit-learn Router for Intents

🎯 Goal: Build an intent router that decides when to call which pipeline.

📚 Learn (Links):

- scikit-learn text classification docs.

💻 Hands-on:

- Build an intent classifier (e.g., "support", "billing", "general", "chit-chat").
- Evaluate with precision, recall, F1.

🛠️ Project Work:

- Module: 01_data_foundation/router/intent_router.py exposing route_intent(text).

🧠 Notes:

- Describe how this router will sit in front of your LLM API.

📌 Daily Output:

- Code + saved model + metrics.

⏱️ Time Spent:

- ~8 hours.

Day 8 – Text Feature Engineering

🎯 Goal: Convert text to features beyond simple bag-of-words.

📚 Learn (Links):

- Articles on TF-IDF, n-grams, character n-grams, tokenization basics.

💻 Hands-on:

- Using your intent dataset:
  - Compare BoW vs TF-IDF vs character n-gram.
  - Train models and log metrics.

🛠️ Project Work:

- Notebook: 01_data_foundation/text_features_experiments.ipynb.

🧠 Notes:

- Note which representation gave best balance of performance and speed.

📌 Daily Output:

- Notebook + updated router using best representation.

⏱️ Time Spent:

- ~8 hours.

Day 9 – Evaluation: Precision/Recall vs Perplexity

🎯 Goal: Understand metrics for classification vs language modeling.

📚 Learn (Links):

- Video/article on precision/recall/F1.
- Article on perplexity for language models.

💻 Hands-on:

- Compute PR/F1 for your router.
- Train a simple n-gram language model and compute perplexity.

🛠️ Project Work:

- Short doc: "When to use PR/F1 vs Perplexity in GenAI systems".

🧠 Notes:

- Write 3 Q&A style notes you can reuse in interviews.

📌 Daily Output:

- Metrics notebook + markdown explanation.

⏱️ Time Spent:

- ~8 hours.

Day 10 – Hyperparameter Tuning

🎯 Goal: Improve your intent router using systematic tuning.

📚 Learn (Links):

- GridSearchCV / RandomizedSearchCV documentation or equivalent.

💻 Hands-on:

- Tune hyperparameters for your best model.

- Log performance gains and training time.

🛠️ Project Work:

- Script: 01_data_foundation/train_router.py using config file for reproducibility.

🧠 Notes:

- Note which hyperparameters were most sensitive and why.

📌 Daily Output:

- Updated router + config + training logs.

⏱️ Time Spent:

- ~8 hours.

Day 11 – Explainability & Hallucination Intuition

🎯 Goal: Connect attention and explanation to hallucinations.

📚 Learn (Links):

- Visual explanation of attention maps.
- Blog about hallucinations and causes.

💻 Hands-on:

- Take a small transformer model (or prebuilt example) and visualize attention for a few inputs.

🛠️ Project Work:

- Notebook: 01_data_foundation/attention_visualization.ipynb.
- Note 3 patterns from the visualization.

🧠 Notes:

- Draft a short explanation: “Why do LLMs hallucinate?” in your own words.

📌 Daily Output:

- Notebook + short write-up.

⏱️ Time Spent:

- ~8 hours.

Day 12 – Project 1: Intelligent Lead Scraper & Classifier

🎯 Goal: Build an end-to-end data pipeline that classifies leads before any LLM step.

📚 Learn (Links):

- Tutorial on ethical web scraping (or use mock/synthetic data).
- Example of ML pipeline in production.

💻 Hands-on:

- Acquire 10k+ sample "lead" records (or generate synthetic).
- Clean, featurize, and classify leads as hot/warm/cold using tuned model.

🛠️ Project Work:

- End-to-end structure:
  - data/, src/, models/, scripts/.
  - CLI to run pipeline: ingest → clean → classify → export.
- Add docs/architecture.md explaining flow and decisions.

🧠 Notes:

- Write a "Business story": how this pipeline could save time/cost before LLM processing.

📌 Daily Output:

- GitHub repo section 01_data_foundation/project_lead_classifier/.
- Resume line example:
  - "Engineered a data pipeline that classifies 10k+ leads with >90% accuracy before LLM processing, reducing downstream model calls and cost."

⏱️ Time Spent:

- ~8 hours.

Phase 2 – NLP to Agentic Thinking (Day 13–25)

Day 13 – Tokenization & Cost Awareness

🎯 Goal: Understand tokens, model limits, and cost.

📚 Learn (Links):

- Docs for tokenization in Gemini or any mainstream LLM.
- Short tutorial on token counting and pricing.

💻 Hands-on:

- Write a token_counter.py script that:
  - Counts tokens for different prompts.

- Estimates cost across several models.

🛠️ Project Work:

- Notebook: 02_nlp_agents/tokenization_and_cost.ipynb with examples.

🧠 Notes:

- Note patterns that make prompts shorter but still effective.

📌 Daily Output:

- Token counter script + notebook.

⏱️ Time Spent:

- ~8 hours.

Day 14 – Modern Embeddings (Dense vs Late Interaction)

🎯 Goal: Learn how embeddings work and where late interaction (ColBERT-style) fits.

📚 Learn (Links):

- Intro to sentence embeddings.
- Simple explanation of ColBERT / late interaction.

💻 Hands-on:

- Use any open embedding model to embed a small corpus.
- Compute cosine similarities and examine nearest neighbors.

🛠️ Project Work:

- Notebook: 02_nlp_agents/embedding_basics.ipynb.

🧠 Notes:

- Write pros/cons of dense vs late interaction for RAG.

📌 Daily Output:

- Embedding experiments notebook.

⏱️ Time Spent:

- ~8 hours.

Day 15 – Sentiment Analysis 2.0

🎯 Goal: Build a sentiment model that handles nuance and sarcasm better than a naive baseline.

📚 Learn (Links):

- Article on advanced sentiment (aspect-based, sarcasm handling).

💻 Hands-on:

- Use a sentiment dataset (e.g., movie reviews, tweets).
- Train/finetune a model or use an existing transformer for analysis.

🛠️ Project Work:

- Evaluate on tricky sarcastic samples.
- Save misclassified examples for future prompting experiments.

🧠 Notes:

- Write 3 ways LLMs can help improve sentiment analysis.

📌 Daily Output:

- Sentiment classification notebook + small report.

⏱️ Time Spent:

- ~8 hours.

Day 16 – Vector Theory: Cosine vs Euclidean

🎯 Goal: Deepen intuition about similarity metrics in embedding spaces.

📚 Learn (Links):

- Explanation of cosine similarity vs Euclidean distance.

💻 Hands-on:

- Generate vectors and compare pairwise metrics in 2D/3D.
- Visualize with plots.

🛠️ Project Work:

- Notebook: 02_nlp_agents/vector_metrics.ipynb with visualizations.

🧠 Notes:

- Decide which metric you'll default to for retrieval and why.

📌 Daily Output:

- Notebook + short conclusion.

⏱️ Time Spent:

- ~8 hours.

Day 17 – Transformer Internals (Self vs Cross-Attention)

🎯 Goal: Understand the internal building blocks of transformers.

📚 Learn (Links):

- "Illustrated Transformer" style blog or video.
- Short explanation of self-attention and cross-attention.

💻 Hands-on:

- Implement a minimal transformer encoder block in PyTorch.

🛠️ Project Work:

- Notebook: 02_nlp_agents/mini_transformer.ipynb with code and explanations.

🧠 Notes:

- Connect Day 2 NumPy attention to this PyTorch implementation.

📌 Daily Output:

- Working mini transformer block in notebook.

⏱️ Time Spent:

- ~8 hours.

Day 18 – PyTorch & VRAM Basics

🎯 Goal: Manage tensors and GPU memory.

📚 Learn (Links):

- PyTorch "getting started" and memory best practices.

💻 Hands-on:

- Run small model on CPU and GPU (Colab/Kaggle).
- Log VRAM usage for different batch sizes.

🛠️ Project Work:

- Script: 02_nlp_agents/pytorch_vram_demo.py logging usage.

🧠 Notes:

- Note safe batch sizes for your typical hardware.

📌 Daily Output:

- Script + short VRAM usage table.

⏱️ Time Spent:

- ~8 hours.

Day 19 – Text Classifier from Scratch

🎯 Goal: Build a neural text classifier end-to-end.

📚 Learn (Links):

- Tutorial: text classification with PyTorch.

💻 Hands-on:

- Create a simple dataset (e.g., intents or topics).
- Build embedding + LSTM/Transformer classifier.

🛠️ Project Work:

- Save model, training plots, and evaluation metrics.

🧠 Notes:

- Compare this to your classical ML router.

📌 Daily Output:

- Notebook & saved model.

⏱️ Time Spent:

- ~8 hours.

Day 20 – Transformer Paper Deep Dive

🎯 Goal: Read and "MCA-style" summarize the original Attention paper.

📚 Learn (Links):

- "Attention Is All You Need" summary blogs.

💻 Hands-on:

- Map sections of the paper to your mini transformer implementation.

🛠️ Project Work:

- 2–3 page 02_nlp_agents/transformer_mini_survey.pdf:
    - Include original Transformer vs modern variants.

🧠 Notes:

- Extract 3 concepts that you can explain in interviews.

📌 Daily Output:

- PDF summary + notes.

⏱️ Time Spent:

- ~8 hours.

Day 21 – SLMs vs Big LLMs (Llama/Gemma/etc.)

🎯 Goal: Understand small language models and trade-offs.

📚 Learn (Links):

- Blog posts describing small, efficient models (Llama 3, Gemma, Phi).

💻 Hands-on:

- Run a small quantized open model locally using any standard tool.

🛠️ Project Work:

- Benchmark latency vs one cloud LLM using the same prompts.

🧠 Notes:

- Write when you'd choose SLMs over large closed models.

📌 Daily Output:

- Benchmark notebook + findings.

⏱️ Time Spent:

- ~8 hours.

Day 22 – Hugging Face & Inference

🎯 Goal: Use HF Hub and basic inference endpoints.

📚 Learn (Links):

- Hugging Face basic course or quickstart.

💻 Hands-on:

- Load a text generation or classification model from HF.
- Run inference on your own sample texts.

🛠️ Project Work:

- Script: 02_nlp_agents/hf_inference_demo.py.

🧠 Notes:

- Note how HF models can plug into your future RAG pipelines.

📌 Daily Output:

- Script + sample outputs in README.

⏱️ Time Spent:

- ~8 hours.

Day 23 – Quantization (GGUF/EXL2)

🎯 Goal: Run larger models on modest hardware.

📚 Learn (Links):

- Introduction to quantization for LLMs (GGUF/EXL2).

💻 Hands-on:

- Compare memory use and speed for quantized vs full-precision model.

🛠️ Project Work:

- 02_nlp_agents/quantization_experiments.md with numbers.

🧠 Notes:

- Decide what quantization level you'll default to for local experiments.

📌 Daily Output:

- Notes + scripts used.

⏱️ Time Spent:

- ~8 hours.

Day 24 – Model Merging

🎯 Goal: Understand when and how to merge models or LoRA adapters.

📚 Learn (Links):

- Blog/tutorial on merging LoRA adapters or models.

💻 Hands-on:

- Perform a simple adapter merge experiment for a classification task.

🛠️ Project Work:

- Record performance before/after merging.

🧠 Notes:

- Write when model merging is dangerous vs useful.

📌 Daily Output:

- Notebook with metrics.

⏱️ Time Spent:

- ~8 hours.

Day 25 – Project 2: Local Privacy-First Support Bot

🎯 Goal: Build a private support bot running locally.

📚 Learn (Links):

- Tutorial for local chatbots with a small model.

💻 Hands-on:

- Deploy a quantized SLM locally.
- Ingest a small set of support docs (text/Markdown).
- Answer questions without external APIs.

🛠️ Project Work:

- Package as a small app (CLI or simple web UI).
- Add docs/architecture.md and "Business Impact" section.

🧠 Notes:

- Prepare a resume bullet highlighting privacy and cost savings.

📌 Daily Output:

- GitHub subfolder 02_nlp_agents/project_local_support_bot/.

⏱️ Time Spent:

- ~8 hours.

Phase 3 – GenAI Architecture & RAG (Day 26–55)

Day 26 – Reasoning Models & Scratchpads

🎯 Goal: Practice Chain-of-Thought and hidden scratchpads.

📚 Learn (Links):

- Materials explaining CoT and reasoning prompts.

💻 Hands-on:

- Compare answers with CoT vs without on math and reasoning tasks.

🛠️ Project Work:

- Notebook: 03_rag_systems/reasoning_prompts.ipynb.

🧠 Notes:

- Note which prompt patterns help the most.

📌 Daily Output:

- Prompt examples + observations.

⏱️ Time Spent:

- ~8 hours.

Day 27 – Prompt Engineering Fundamentals

🎯 Goal: System prompt, few-shot examples, formatting.

📚 Learn (Links):

- Prompt engineering guide from any major provider.

💻 Hands-on:

- Build templates for classification, extraction, summarization, rewriting.

🛠️ Project Work:

- 03_rag_systems/prompts/ folder with reusable templates.

🧠 Notes:

- Keep a "prompt cookbook" for later reuse.

📌 Daily Output:

- Prompt library committed.

⏱️ Time Spent:

- ~8 hours.

Day 28 – Advanced Prompt Patterns

🎯 Goal: Implement advanced patterns like ReAct, Skeleton-of-Thought, Tree-of-Thought.

📚 Learn (Links):

- Blogs on ReAct and ToT/SoT prompts.

💻 Hands-on:

- Apply two patterns to your local support bot and measure quality changes.

🛠️ Project Work:

- Notebook: 03_rag_systems/advanced_prompt_patterns.ipynb.

🧠 Notes:

- Record trade-offs between latency and answer quality.

📌 Daily Output:

- Before/after comparison.

⏱️ Time Spent:

- ~8 hours.

Day 29 – LLM APIs: Rate Limits & Async

🎯 Goal: Build robust async client for LLM APIs.

📚 Learn (Links):

- API documentation of any LLM provider.

💻 Hands-on:

- Implement async batching, retries with backoff, and error handling.

🛠️ Project Work:

- Module: 03_rag_systems/llm_client.py.

🧠 Notes:

- Note patterns for keeping costs and rate-limit issues under control.

📌 Daily Output:

- Client module + sample usage.

⏱️ Time Spent:

- ~8 hours.

Day 30 – Chatbot Memory

🎯 Goal: Implement buffer and summary memory.

📚 Learn (Links):

- Library docs on chat memory (e.g., LangChain).

💻 Hands-on:

- Build a simple chat app with:
    - Sliding window memory.
    - Summarized memory.

🛠️ Project Work:

- Notebook: 03_rag_systems/chat_memory_experiments.ipynb.

🧠 Notes:

- Decide which memory strategy suits long conversations.

📌 Daily Output:

- Chat memory demo.

⏱️ Time Spent:

- ~8 hours.

Day 31 – LangChain vs LlamaIndex

🎯 Goal: Compare two RAG frameworks.

📚 Learn (Links):

- Intro resources for LangChain and LlamaIndex.

💻 Hands-on:

- Build minimal RAG over the same small corpus in both frameworks.

🛠️ Project Work:

- Document pros/cons in 03_rag_systems/framework_comparison.md.

🧠 Notes:

- Choose one primary framework for your future projects.

📌 Daily Output:

- Two mini RAG demos + comparison.

⏱️ Time Spent:

- ~8 hours.

Day 32 – LangGraph & Stateful Workflows

🎯 Goal: Build multi-step, stateful workflows.

📚 Learn (Links):

- LangGraph or similar graph-based workflow tutorial.

💻 Hands-on:

- Implement "research → synthesize → refine" workflow.

🛠️ Project Work:

- Graph definition saved with a simple visual diagram.

🧠 Notes:

- List future flows you want to convert into graphs.

📌 Daily Output:

- Graph-based workflow demo.

⏱️ Time Spent:

- ~8 hours.

Day 33 – Vector Databases (Pinecone/Weaviate/pgvector)

🎯 Goal: Understand and experiment with vector DBs.

📚 Learn (Links):

- Intro guidelines for Pinecone, Weaviate, pgvector.

💻 Hands-on:

- Index a small corpus in pgvector (Postgres) and in one hosted DB.

🛠️ Project Work:

- Notebook: 03_rag_systems/vector_db_benchmarks.ipynb.

🧠 Notes:

- Choose one stack for your capstone RAG.

📌 Daily Output:

- Benchmarks with latency and storage.

⏱️ Time Spent:

- ~8 hours.

Day 34 – Hybrid Search (BM25 + Semantic)

🎯 Goal: Combine keyword search and embeddings.

📚 Learn (Links):

- Blog on hybrid search.

💻 Hands-on:

- Use BM25 (Elasticsearch/Meilisearch) + embeddings and merge results.

🛠️ Project Work:

- Evaluate recall@k for pure BM25, pure semantic, hybrid.

🧠 Notes:

- Decide default hybrid strategy for your RAG systems.

📌 Daily Output:

- Evaluation notebook.

⏱️ Time Spent:

- ~8 hours.

Day 35 – RAG Architecture (Naive vs Advanced)

🎯 Goal: Design robust RAG architectures.

📚 Learn (Links):

- RAG architecture overviews (naive vs advanced).

💻 Hands-on:

- Draw architecture diagrams for:
  - Naive RAG (chunk + embed + retrieve + answer).
  - Advanced RAG (query rewrite, rerank, self-correction, feedback loop).

🛠️ Project Work:

- Document in 03_rag_systems/rag_architectures.md.

🧠 Notes:

- Define v1 vs v2 goals for your upcoming RAG project.

📌 Daily Output:

- Architecture diagrams and doc.

⏱️ Time Spent:

- ~8 hours.

Days 36–40 – Project 3: "Agentic" Enterprise Knowledge Base

🎯 Goal: Build a serious RAG system with self-correction and evaluation.

📚 Learn (Links):

- RAG evaluation frameworks (e.g., RAGAS).

💻 Hands-on (spread across 5 days):

- Day 36:
    - Prepare domain docs (product guides, policies, FAQs).
    - Implement ingestion and chunking (semantic/recursive).
- Day 37:
    - Build retrieval pipeline with hybrid search.
- Day 38:
    - Create answer generation chain with self-critique step.
- Day 39:
    - Integrate RAG evaluation (RAGAS-style metrics).
- Day 40:
    - Clean up, write docs, prepare demo script.

🛠️ Project Work:

- Full project under 03_rag_systems/project_enterprise_kb/.
- Include:
    - docs/architecture.md
    - deployment.md
    - evaluation_results.md

🧠 Notes:

- Draft resume bullet emphasizing hallucination reduction and reliability.

📌 Daily Output:

- Production-style RAG repo.

⏱️ Time Spent:

- ~8 hours/day.

Day 41 – Fine-Tuning (PEFT/LoRA) – When & Why

🎯 Goal: Understand when fine-tuning adds value.

📚 Learn (Links):

- Intro to PEFT/LoRA on a small model.

💻 Hands-on:

- Fine-tune a small model for a narrow task (e.g., classification or style).

🛠️ Project Work:

- 03_rag_systems/fine_tuning_experiment.md with hypothesis, setup, results.

🧠 Notes:

- Decision checklist: Prompt vs RAG vs Fine-tune.

📌 Daily Output:

- Experiment notebook + summary doc.

⏱️ Time Spent:

- ~8 hours.

Day 42 – Tool/Function Calling

🎯 Goal: Let LLM call tools (calculator, DB, web).

📚 Learn (Links):

- Tool/function calling docs from any major LLM provider.

💻 Hands-on:

- Wire at least 2 tools:
    - Calculator.
    - Simple DB query.

🛠️ Project Work:

- Module: 03_rag_systems/tools/ with tool definitions and an example orchestrator.

🧠 Notes:

- Note security and validation considerations for tools.

📌 Daily Output:

- Function calling demo.

⏱️ Time Spent:

- ~8 hours.

Day 43 – AI Agents (ReAct)

🎯 Goal: Implement ReAct agent (reason + act loop).

📚 Learn (Links):

- ReAct pattern explanation.

💻 Hands-on:

- Build an agent that plans, calls tools, and responds.

🛠️ Project Work:

- Trace logs of reasoning steps for debugging.

🧠 Notes:

- Note how ReAct compares to simple prompting.

📌 Daily Output:

- ReAct agent script/notebook.

⏱️ Time Spent:

- ~8 hours.

Day 44 – Multi-Agent Systems (CrewAI/Autogen)

🎯 Goal: Orchestrate multiple specialized agents.

📚 Learn (Links):

- Multi-agent framework documentation.

💻 Hands-on:

- Two agents: Researcher + Writer collaborating to produce a blog or report.

🛠️ Project Work:

- Keep conversation logs for later analysis.

🧠 Notes:

- Identify at least 2 real use-cases for multi-agent setups.

📌 Daily Output:

- Multi-agent demo.

⏱️ Time Spent:

- ~8 hours.

Day 45 – Researcher & Writer Agent Team

🎯 Goal: Productionize the Researcher–Writer pattern.

📚 Learn (Links):

- Article describing research pipelines with agents.

💻 Hands-on:

- Feed agents tasks from GitHub issues or technology topics.

🛠️ Project Work:

- Create pipeline that:
  - Collects information.
  - Produces structured report or draft PR description.

🧠 Notes:

- Record one "case study" you can show in interviews.

📌 Daily Output:

- Agent pipeline with documentation.

⏱️ Time Spent:

- ~8 hours.

Day 46 – Long-Term Memory (Mem0/Zep)

🎯 Goal: Add persistent memory for agents.

📚 Learn (Links):

- Docs of one memory framework (Mem0/Zep/etc.).

💻 Hands-on:

- Integrate into an existing agent system so it remembers preferences.

🛠️ Project Work:

- Demo script: run two sessions and show persistence.

🧠 Notes:

- Note privacy/storage considerations for long-term memory.

📌 Daily Output:

- Memory-enhanced agent demo.

⏱️ Time Spent:

- ~8 hours.

Day 47 – API Integration (Slack/Jira/GitHub)

🎯 Goal: Connect agents to real productivity tools.

📚 Learn (Links):

- API docs for Slack, Jira, or GitHub.

💻 Hands-on:

- Agent that reads GitHub issues and posts a summary comment or Slack message.

🛠️ Project Work:

- 03_rag_systems/integrations/ with simple connectors.

🧠 Notes:

- Think about access control and secure tokens.

📌 Daily Output:

- Integration demo.

⏱️ Time Spent:

- ~8 hours.

Day 48 – Guardrails & Safety

🎯 Goal: Add safety filters and guardrails.

📚 Learn (Links):

- Intro to guardrail frameworks and prompt injection defenses.

💻 Hands-on:

- Implement content filters (regex + model-based) on inputs and outputs.

🛠️ Project Work:

- 03_rag_systems/guardrails.md documenting your rules.

🧠 Notes:

- Prepare 2–3 examples of blocked vs allowed prompts.

📌 Daily Output:

- Guardrail layer wired into one app.

⏱️ Time Spent:

- ~8 hours.

Day 49 – LLM-as-a-Judge

🎯 Goal: Use an LLM to evaluate other LLM outputs.

📚 Learn (Links):

- Articles on LLM-as-judge and evaluation frameworks.

💻 Hands-on:

- Ask model to rate your RAG answers for correctness, completeness, style.

🛠️ Project Work:

- Store judgments in a structured format for later analysis.

🧠 Notes:

- Think about bias risks in model-based evaluation.

📌 Daily Output:

- Evaluation notebook + results.

⏱️ Time Spent:

- ~8 hours.

Day 50 – Observability (LangSmith/Arize-style)

🎯 Goal: Add tracing, logging, and metrics.

📚 Learn (Links):

- Observability tutorials for LLM applications.

💻 Hands-on:

- Instrument one RAG app with:
    - Prompt/response logging.
    - Latency and error metrics.

🛠️ Project Work:

- Screenshots or exports of key dashboards.

🧠 Notes:

- Note which signals matter most in production.

📌 Daily Output:

- Observability layer integrated.

⏱️ Time Spent:

- ~8 hours.

Days 51–55 – Project 4: Autonomous DevOps Agent

🎯 Goal: Build multi-agent DevOps assistant with human-in-the-loop.

📚 Learn (Links):

- Examples of GitHub-aware coding agents.

💻 Hands-on (spread across 5 days):

- Day 51:
  - Agent monitors GitHub issues.
- Day 52:
  - Agent proposes fixes (files and diffs).
- Day 53:
  - Agent drafts PRs, with clear summaries.
- Day 54:
  - Add human approval workflow.
- Day 55:
  - Polish, document, record demo.

🛠️ Project Work:

- Project directory 03_rag_systems/project_devops_agent/.
- Full docs (architecture, assumptions, limitations).

🧠 Notes:

- Prepare a story showing impact (time saved, automation percentage).

📌 Daily Output:

- Complete DevOps agent project.

⏱️ Time Spent:

- ~8 hours/day.

Phase 4 – Production & Job Ready (Day 56–90)

Day 56 – FastAPI with Streaming

🎯 Goal: Serve chat responses with streaming.

📚 Learn (Links):

- FastAPI streaming response examples.

💻 Hands-on:

- Wrap one chatbot/RAG app with a streaming endpoint.

🛠️ Project Work:

- Add basic logging and error handling.

🧠 Notes:

- Note advantages of streaming in UX and perceived latency.

📌 Daily Output:

- FastAPI app with streaming.

⏱️ Time Spent:

- ~8 hours.

Day 57 – Caching with Redis

🎯 Goal: Cache expensive LLM calls.

📚 Learn (Links):

- Redis caching tutorials.

💻 Hands-on:

- Cache responses keyed by prompt and model settings.

🛠️ Project Work:

- Track cache hit rate in your logs.

🧠 Notes:

- Note patterns where caching helps most.

📌 Daily Output:

- Redis-integrated service.

⏱️ Time Spent:

- ~8 hours.

Day 58 – Security: Keys & PII

🎯 Goal: Handle secrets and user data securely.

📚 Learn (Links):

- OWASP API basics.
- Basics of PII and redaction.

💻 Hands-on:

- Move all keys to environment variables or secrets.
- Implement basic PII redaction before sending to models.

🛠️ Project Work:

- Security checklist for your apps.

🧠 Notes:

- Write 3 interview-ready points on AI security.

📌 Daily Output:

- Hardened configuration and middleware.

⏱️ Time Spent:

- ~8 hours.

Day 59 – Dockerizing AI Apps

🎯 Goal: Containerize LLM/RAG services.

📚 Learn (Links):

- Docker basics for Python and AI.

💻 Hands-on:

- Dockerize one RAG app with dependencies.
- Add docker-compose for app + Redis + DB.

🛠️ Project Work:

- Push image to Docker Hub (optional).

🧠 Notes:

- Note image size and build time and how to optimize.

📌 Daily Output:

- Docker configs committed.

⏱️ Time Spent:

- ~8 hours.

Day 60 – Cloud Deployment (e.g., Cloud Run/EC2)

🎯 Goal: Deploy one app to the cloud.

📚 Learn (Links):

- Quickstart for deployment on any major cloud provider.

💻 Hands-on:

- Deploy FastAPI + RAG app with HTTPS.

🛠️ Project Work:

- Public URL and instructions in deployment.md.

🧠 Notes:

- Note cost estimates and scaling options.

📌 Daily Output:

- Live demo link.

⏱️ Time Spent:

- ~8 hours.

Days 61–65 – Capstone: AI Customer Support Agent

🎯 Goal: Build your flagship production-style project.

📚 Learn (Links):

- Case studies of AI customer support/RAG systems.

💻 Hands-on (spread across 5 days):

- Ingest FAQ, product docs, and chat logs (anonymized).
- RAG pipeline with hybrid search and guardrails.
- Agent with tools (ticket lookup, order status).
- Dashboard (simple UI or metrics page) with latency, usage, satisfaction.
- Full documentation and demo video.

🛠️ Project Work:

- Project directory 04_production/project_customer_support_agent/.

🧠 Notes:

- Treat this as "mini-startup product" in your narrative.

📌 Daily Output:

- Capstone repo + demo.

⏱️ Time Spent:

- ~8 hours/day.

Days 66–70 – Portfolio & GitHub Polish

🎯 Goal: Turn projects into "experience".

📚 Learn (Links):

- Articles on software portfolio best practices.

💻 Hands-on:

- For each project:
  - Clean README, add screenshots and diagrams.
  - Add basic CI (lint/tests).

🛠️ Project Work:

- Create portfolio site (static or simple React) listing your 5 main projects.

🧠 Notes:

- Write 1–2 lines explaining how each project connects to business value.

📌 Daily Output:

- Polished repos + portfolio site.

⏱️ Time Spent:

- ~8 hours/day.

Days 71–75 – System Design & Gap Narrative

🎯 Goal: Prepare for system design and career-gap questions.

📚 Learn (Links):

- System design articles focused on AI/LLM systems.

💻 Hands-on:

- Write design docs for:

        - Enterprise RAG system.
        - DevOps Agent.
        - Customer Support Agent.
    - Draft 3 key answers:
        - About your 4-year gap.
        - Why GenAI now.
        - Walkthrough of your most complex project.

🛠️ Project Work:

- Store docs in 04_production/system_design/.

🧠 Notes:

- Refine your elevator pitch.

📌 Daily Output:

- Design docs + written answers.

⏱️ Time Spent:

- ~8 hours/day.

Days 76–80 – Resume, LinkedIn & Case Studies

🎯 Goal: Present yourself as "MCA GenAI Engineer restarting with strong projects".

📚 Learn (Links):

- Guides on writing tech resumes and LinkedIn profiles.

💻 Hands-on:

- Rewrite resume using STAR bullets for each project.
- Update LinkedIn headline, About, and Experience sections.
- Write 2–3 case-study style blog posts (Medium/Hashnode/GitHub Pages).

🛠️ Project Work:

- Create a single PDF portfolio with project summaries and links.

🧠 Notes:

- Ensure your gap is clearly explained as structured upskilling.

📌 Daily Output:

- Finalized resume + LinkedIn.

⏱️ Time Spent:

- ~8 hours/day.

Days 81–85 – Interview Drills (Tech + Behavioral)

🎯 Goal: Practice explaining concepts and your story.

📚 Learn (Links):

- Common GenAI/ML interview question lists.

💻 Hands-on:

- Daily:
  - 3–5 GenAI/ML concept questions.
  - 3–5 behavioral/gap-related questions.
- Use an AI assistant to simulate interviews and critique your answers.

🛠️ Project Work:

- Maintain 04_production/interview_qa.md with your best answers.

🧠 Notes:

- Note questions you struggle with and revisit them.

📌 Daily Output:

- Q&A document growing daily.

⏱️ Time Spent:

- ~8 hours/day.

Days 86–90 – Job Hunt Sprint

🎯 Goal: Apply strategically and network.

📚 Learn (Links):

- Guides on networking and targeted applications in tech.

💻 Hands-on:

- Prepare 2–3 personalized application templates.
- Apply to roles that match: "GenAI Engineer", "AI Engineer (Python)", "LLM/RAG Engineer".
- Reach out to hiring managers/engineers with short messages + portfolio link.

- Post at least 3 LinkedIn updates showcasing your projects and learnings.

🛠️ Project Work:

- Track applications in a simple Notion or spreadsheet board.

🧠 Notes:

- Reflect daily on what is working in your outreach and adjust.

📌 Daily Output:

- Application log + networking messages.

⏱️ Time Spent:

- ~8 hours/day.