

FLM's AI Mastery Course (From Python to Agentic AI)

Project Details

There will be 2 mini projects and 2 major project and for each concept we will be having a Real time Use Case scenario for better understanding of the Concept

Minor Projects:

Project 1: SkillGap AI - Intelligent Career Readiness & Personalized Upskilling Roadmap Generator.

Description:

An AI-powered system that helps students and early-career professionals bridge the gap between where they are and where they want to be — career-wise. The user inputs their current skill set or resume text alongside a target job description, and the system performs a deep comparative analysis. It identifies missing skills, maps them to priority levels based on job market demand, and generates a structured 30-60-90 day upskilling roadmap with specific, free or low-cost learning resources for each gap.

The project addresses one of the most frustrating realities for job seekers: knowing they lack skills but not knowing exactly which ones to build, in what order, or how.

Key Value Proposition:

Removes the guesswork from career preparation by delivering a clear, prioritized, AI-generated action plan that connects current skills to target job requirements — so learners spend time building what actually matters for their specific goal.

Scope:

MVP focuses on text-based inputs (resume paste + job description paste) and outputs a structured gap analysis with a prioritized learning roadmap.

Future scope includes integration with LinkedIn profiles, live job posting APIs, and course platform recommendations.

Assumptions:

Users provide honest descriptions of their current skills, projects, and background in text form, along with a real job description they are targeting.

Example Use Case

A student inputs:

- Their resume: Python basics, Excel, one data project, MBA coursework
- Target JD: Data Analyst role at a fintech company requiring SQL, Tableau, Power BI, business analysis, and stakeholder communication

The system:

- Identifies SQL and Tableau as critical high-priority gaps
- Flags stakeholder communication as a soft skill gap
- Assigns a 30-60-90 day roadmap with specific free resources (e.g., Mode SQL Tutorial, Tableau Public courses)
- Scores their current readiness as a percentage
- Highlights which existing skills are already strong matches

Project 2: ContractScan AI — Intelligent Vendor Contract Risk Spotter for Non-Legal Users

Description:

An AI-powered system that reads vendor contracts, service agreements, or freelance contracts and flags hidden risk clauses in plain, easy-to-understand language. The system identifies problematic areas such as one-sided liability clauses, auto-renewal traps, vague IP ownership terms, unreasonable payment conditions, unfair termination penalties, and non-compete overreach — then provides an overall risk score and specific renegotiation suggestions.

The project solves a critical real-world problem: small business owners, startup

founders, and freelancers routinely sign contracts they do not fully understand, exposing themselves to financial and legal risks that could have been caught early with basic AI analysis.

Target Users:

Freelancers, startup founders, solopreneurs, procurement managers at small businesses, and early-stage company employees who deal with vendor or client contracts but do not have access to in-house legal counsel.

Key Value Proposition:

Democratizes contract awareness by giving non-legal users a fast AI-powered safety net before signing — surfacing the 20% of clauses that cause 80% of post-signing regrets, without requiring legal expertise.

Scope:

MVP focuses on pasted or uploaded text-based contract content, analyzing and flagging risk clauses with structured explanations and a risk summary.

Future scope includes clause-by-clause comparison against industry standards and integration with e-signature platforms.

Assumptions:

Users provide the full text of a contract or paste the specific sections they want analyzed.

Example Use Case:

A startup founder uploads a vendor SaaS contract and the system:

- Flags a 12-month auto-renewal clause with 90-day cancellation window buried in Section 8
- Identifies an IP assignment clause that transfers ownership of customizations to the vendor
- Highlights a liability cap set at 1x monthly fee even for vendor negligence
- Spots a data usage clause allowing the vendor to use customer data for product improvement

- Delivers an overall risk score of 72/100 (High Risk)
- Suggests three specific renegotiation points with sample counter-language

MAJOR PROJECTS

Project 1: MeetBrain AI — Intelligent Meeting Transcript Processor & Follow-Up Automation Engine

Description:

A Python-based web application that transforms raw meeting transcripts — pasted as text or uploaded as files — into structured, actionable intelligence using AI. The system processes the transcript and generates a comprehensive Meeting Intelligence Report: a crisp summary, a list of decisions made, clearly extracted action items with owner and deadline assignments, a log of open questions, and a ready-to-send follow-up email draft for participants. The entire post-meeting workflow that typically takes 30–60 minutes of manual effort is compressed into under 60 seconds.

The project addresses one of the most costly productivity drains in modern organizations: meetings that end without clear accountability. Studies consistently show that over 60% of meeting action items are forgotten or ambiguous within 24 hours, leading to delayed deliverables, missed commitments, and repeated discussions.

Target Users:

Managers, team leads, project managers, startup founders, scrum masters, and remote-first teams who hold regular meetings and struggle to maintain structured follow-through and accountability across participants.

Key Value Proposition:

Converts passive meeting records into active accountability systems — giving teams instant clarity on who owns what, what was decided, and what still needs resolution, without manual note-taking or post-meeting effort.

Scope:

MVP focuses on transcript text input (copy-paste or plain text upload) with structured AI output including summary, decisions, action items, open questions, and an email draft.

Future scope includes audio file input via Whisper transcription, Slack/email integration for direct delivery, and calendar-linked action item tracking.

Assumptions:

Users provide the meeting transcript as text — either pasted directly or uploaded as a .txt or .pdf file.

Example Use Case

A project manager pastes the transcript of a 45-minute sprint planning meeting.

MeetBrain AI:

Produces a 5-sentence executive summary of what the meeting covered

Extracts 3 decisions made:

- Sprint 12 scope finalized with 18 story points
- Launch date moved to March 28
- Performance testing to be skipped for this sprint

Generates 6 action items:

- Arjun to update the backlog in Jira by Friday
- Priya to notify the design team about the new timeline by EOD
- Dev team to begin API integration work on Monday

Flags 2 open questions:

- Staging environment access still unconfirmed
- QA resource availability for next sprint unclear

Drafts a follow-up email ready to send to all attendees with all the above structured clearly

1. Project Title & Overview — (covered above)

2. Target Users — (covered above)

3. Key Value Proposition — (covered above)

4. Scope — (covered above)

5. Assumptions — (covered above)

6. Core System Design

The system should be AI-first, meaning transcript understanding, entity extraction, reasoning about ownership and timelines, and follow-up generation should all be driven by LLM capabilities — not hard-coded rules.

Input: Raw meeting transcript text (minimum 500 words for reliable extraction)

Processing: Multi-step AI pipeline using prompt chaining or LangChain/LangGraph

Output:

- Meeting Summary (5–7 sentences)
- Decisions Log (bulleted, clear, attributable where possible)
- Action Items Table (Item | Owner | Deadline | Priority)
- Open Questions List (with context)
- Follow-Up Email Draft (formatted, professional)

7. Minimum Build Requirements

A) One clear entry point

Web UI built with Streamlit or FastAPI + simple frontend.

User pastes or uploads transcript and clicks "Process Meeting."

B) End-to-end AI workflow

Transcript Input → Chunking & Context Setup → AI Extraction Pipeline → Structured Output Rendering → Email Draft Generation
At least 4 logical AI-driven steps in the pipeline.

C) Agentic behavior

The system should reason about who owns action items even when not explicitly stated (e.g., "Dev team should handle this" → assigns to Dev team).

It should distinguish between a decision (final) and an open question (unresolved).

It should handle ambiguous or messy transcripts gracefully with fallback outputs.

D) Reliability layer

Structured JSON output for each extraction task (action items, decisions, questions)

Fallback message if transcript is too short or unstructured

Guardrails to prevent hallucinating names or deadlines not present in the transcript

E) Working deployed demo

Deployed on Render, Railway, or Hugging Face Spaces

Clear run instructions and a sample transcript for testing

Reproducible end-to-end demo

8. Recommended Tech Stack

Core:

- Python (FastAPI or Flask for backend)
- LLM: OpenAI GPT-4o / Gemini 1.5 Pro / Claude Haiku
- LangChain or LangGraph for prompt chaining
- Pydantic for structured output schemas

UI:

- Streamlit (recommended for speed) or basic HTML/CSS frontend

Deployment:

- Render / Railway / Hugging Face Spaces

- Docker (optional)

Optional Enhancements:

- Whisper API for audio-to-text transcription
- SMTP integration for actual email delivery
- Notion or Google Docs output formatting

9. Applying Course Learnings

Your build should visibly apply:

- Python fundamentals and API integration
- Prompt engineering and structured output extraction
- LLM selection and understanding of context windows
- LangChain / LangGraph pipeline construction
- Safety and hallucination guardrails
- GitHub repo structure with README
- Deployment to a cloud platform
- Evaluation thinking: how do you verify the action items extracted are accurate?

10. Existing Solutions to Research Before Building

Before finalizing your solution, study these tools and identify your differentiation:

- Otter.ai (transcription + basic notes)
- Fireflies.ai (meeting recording + summaries)
- Notion AI (document summarization)
- ChatGPT for meeting notes (generic, no structured output pipeline)

Your angle: Deeper structured extraction, customizable output format, transparent AI reasoning, and no vendor lock-in.

11. Final Deliverables

- Working deployed web application

- GitHub repository with clear README and run instructions
- Sample transcript test case with documented output
- Demo walkthrough (screen recording or live demo)
- Brief report explaining design decisions, challenges, and next steps

12. Review & Assessment Criteria

- AI-first design: Is the core value driven by AI, not just formatting?
- Extraction accuracy: Are action items, decisions, and questions correctly identified?
- Working demo: Can a reviewer test it with their own transcript?
- Structured output quality: Is the output clean, actionable, and professional?
- Business relevance: Does it solve a real, recognizable problem clearly?

Project 2: Enterprise AI Capstone Project

Description:

This capstone is designed to make you **job-ready for AI Engineer roles**. Its not just a theory, it will be your Project get job you AI and Job ready

In real companies, AI engineers do not just experiment with models.

They **identify real problems**, design **AI-first systems**, build **working solutions**, and deploy them so others can use them.

In this project, you will:

1. Select a real product from a real company{You can select any product from any company of your choice, or enterprise level problem from any domain of your choice}
2. Identify a real enterprise-level problem
3. Design an **AI-first (GenAI / Agentic)** solution
4. **Build and deploy** a working system{You will code and will deploy}
5. Validate it and explain its business value{Testing phase and will explain in document}

You are encouraged to use **GenAI tools for development** (ChatGPT, Codex-style coding, vibe coding, copilots..... etc).

What matters is not manual coding speed — but **system thinking, correctness, and clarity**.

This capstone is structured this way to:

- Simulate **real enterprise AI work**
- Help you apply **everything you learned in the course**
- Give you a **strong portfolio-grade project**
- Make interviews and reviews easier to handle

Step 1: Product Selection

Choose **one real product** from a real company

(e.g., Microsoft, Google, Uber, Meta, PhonePe, ShareChat, Front Line Media, etc.)

[{Click Here for companies listed based on domain categorization}](#)

Note : You can choose any company from any domain based on below:

1. Which domain do you want to target for your next job ?
2. Select any company from that domain [Companies list click here](#)
3. Select a product or enterprise problem you want to target from your filtered company

Focus on:

- What the product does
- Who uses it
- Where complexity or scale exists

👉 Why this matters

AI engineers always work **inside a product context**, not in isolation.

Step 2: Enterprise Problem Identification

Identify **one clear, specific problem** related to the product.

The problem should:

- Affect users or internal teams
- Exist at scale
- Be meaningful enough to justify an AI solution

Avoid vague problems like “improve UX” or “use AI to automate everything.”

👉 Why this matters

Strong AI systems start with **well-defined problems**, not tools.

Step 3: Target Users & Target Buyers

Clearly define:

- Who directly faces the problem (users)
- Who benefits from solving it
- Who would approve or pay for the solution in a company

👉 Why this matters

Enterprise AI solutions must satisfy **both users and decision-makers**.

Step 4: Use Case Definition

Describe how your solution will be used:

- When the system is triggered
- What input it receives
- What output it produces
- How users interact with it

Think in **flows**, not features.

👉 Why this matters

This converts a problem into a **clear system workflow**.

Step 5: Why Now?

Explain **why this problem needs an AI / GenAI solution today**:

- Why this problem can no longer be solved effectively using humans or rule-based systems
- What value the business gains by solving this problem with AI now (speed, cost, scale, quality)
- What value users gain immediately from an AI-powered solution (ease, accuracy, personalization, accessibility)

👉 Why this matters

AI is applied **when it creates clear value for both business and users at the right time**, not just because the technology exists.

Step 6: AI Solution Overview

Describe your solution at a high level:

- What the AI system does
- Where AI is used
- Where non-AI logic is used
- Where humans are involved (if any)

No deep tech yet — just clarity.

👉 Why this matters

AI engineers must justify **why and where AI is applied**.

Step 7: What You Need to Develop (Core Build Section — AI-First)

This is the **most important section** of the capstone.

Your solution must be **AI-first**, meaning the core value comes from **Generative AI and/or agentic workflows**, not from a normal app with a small AI feature.

7.1 What “AI-First” means

Your system must do at least one of these as its **core capability**:

- Understand messy human inputs (text, docs, logs, screenshots)
- Reason through tasks and decide next steps
- Retrieve relevant context before responding
- Generate structured outputs that drive actions
- Use tools/APIs as part of a workflow
- Coordinate multi-step or multi-agent behavior

If your app still works without AI, then AI is not core.

7.2 Minimum Build Requirements

Your working solution must include:

A) One clear entry point

- Web UI, API, CLI, or simple interface
Reviewer must be able to try it.

B) End-to-end AI workflow

- Input → AI processing → Output
At least **3 logical steps** in the workflow.

C) Agentic behavior (required)

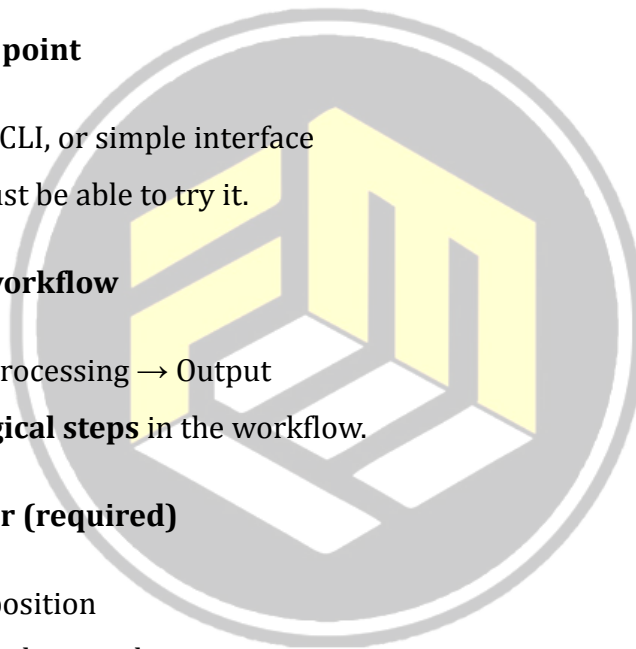
- Task decomposition
- Tool usage or decision logic
- Retry / clarification / reasoning steps

D) Reliability layer

- Guardrails or constraints
- Fallback behavior
- Structured outputs (JSON / schema-style)

E) Working deployed demo

- Cloud or local deployment
- Clear run instructions



- Reproducible demo

7.3 Applying Course Learnings (Required)

Your build should visibly apply **most** of the following:

- Python fundamentals & APIs
- Prompt engineering & reusable prompts
- Safety, ethics, guardrails
- LLM selection & limitations
- GitHub repo structure & versioning
- Hosted or local LLM usage
- RAG / contextual retrieval (if relevant)
- LangChain / LangGraph (or equivalent)
- Integrations or automation (if relevant)
- Evaluation thinking
- Deployment

7.4 Recommended Tech Stack (Flexible)

Core

- Python (FastAPI / Flask)
- LLMs (OpenAI / Gemini / Claude / Ollama)
- LangChain / LangGraph
- Vector DB (Chroma / FAISS if needed)
- GitHub

UI (optional)

- Streamlit or simple frontend

Deployment

- Render / Railway / Vercel
- AWS / GCP (optional)

- Docker (optional)

You choose and justify — no forced stack.

Step 8: Existing Solutions Scan (Required)

Before finalizing your solution, you must check:

- Are there existing products or tools solving this?
- What do they do well?
- Where do they fall short?
- What **specific gap** does your AI solution address?

This does not need heavy research.

Even **3–5 comparable tools** with short notes is enough.

👉 Why this matters

Enterprise AI solutions must clearly explain **how they differ**.

Step 9: AI System Architecture

Create and explain:

- Architecture diagram
- Data flow
- AI components
- Integration points

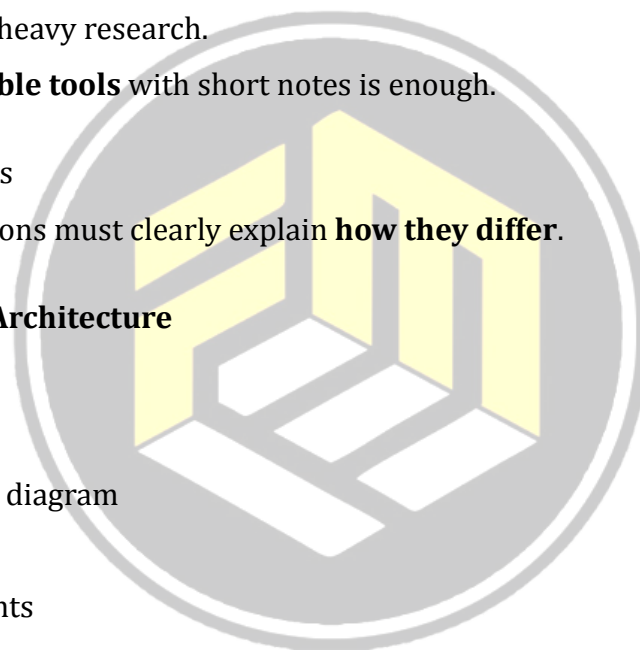
👉 Why this matters

Architecture thinking is a **core AI engineering skill**.

Step 10: Technology Stack & Tools

List:

- Languages
- Models
- Frameworks
- Tools



Explain choices briefly and practically.

Step 11: Integrations & External Systems

Explain any:

- APIs used
- Platforms integrated
- Data sources
- Output channels

Step 12: Development & Implementation

Explain:

- Core modules
- AI reasoning logic
- Error handling
- Edge cases

Step 13: Deployment Strategy

Explain:

- Where it's deployed
- How it's accessed
- Runtime environment

Step 14: Validation & Evaluation

Explain:

- How you tested
- How you validated outputs
- Known limitations

Step 15: Business Value & Impact(Optional)

Explain:



- What improves
 - Who benefits
 - How success is measured
- Step 16: Future Scope & Trajectory**

Explain:

- What you'd build next
- How it could scale
- Long-term vision

Step 17: Final Deliverables

- Working AI solution
- Architecture diagram
- Documentation / README
- Demo walkthrough
- Final report

Step 18: Review & Assessment Criteria

Evaluation based on:

- AI-first design
- Engineering depth
- Working demo
- Reliability & reasoning
- Business relevance

Final note for students

You are not expected to be perfect.

You are expected to be **clear, thoughtful, and hands-on.**

If you can confidently explain:

the problem → the AI system → the demo → the value

you are already operating at an **AI Engineer level.**