🚀 Agentic AI Consultant: Kickstarter Exercises

These exercises are designed to build foundational skills in LLM interaction, code manipulation, web data extraction (crucial for RAG/Agentic systems), and stateful agent implementation using LangGraph. They are **highly relevant** and appropriately challenging for an aspiring **IT/Business Consultant in the domain of Enterprise Agentic AI (LLM/GenAI)**.

Prerequisites

Area	**Requirement**	**Contextual Hint**
Environment	**Python 3.10** or above.	Python is the current industry standard for GenAI development due to rich libraries. The exercises can be adapted to **JavaScript/TypeScript** or **Java** (see Exercise #1).
Libraries	langchain, langgraph, langchain-openai, langchain-google-genai, crawl4ai (or similar web automation/scraping tool).	These are the core building blocks for connecting LLMs to tools and creating stateful agents.
Containerization	Docker image: mcr.microsoft.com/playwright/python:1.55.0-release	This image is essential for web automation exercises (like Exercise #8) as it includes Playwright's necessary browser dependencies.
APIs	An API key for **OpenAI**, **Gemini**, or **Claude**.	**Actionable Tip:** Gemini 2.5 Pro is often excellent for complex

Area	Requirement	Contextual Hint
		reasoning and structured output, making it a strong choice for initial development and proof-of-concepts (POCs).
IDE/Tools	A robust IDE (VSCode, IntelliJ, Eclipse).	**Consultant Insight:** Using tools like Cursor, GitHub Copilot, or even the web interface of an LLM is encouraged. However, **deeply understanding** and **refactoring** the generated code is a non-negotiable skill for a professional consultant.

Phase 1: Environment Setup & Core LLM Interaction

1. Tooling Equivalence Mapping

Find and document the equivalent development environment toolset (base libraries, key LLM wrappers, orchestration frameworks) for the following languages: **JavaScript/TypeScript**, **Java**, and **.NET (C#)**.

- **Key Learning Area (KLA):** Cross-Language Familiarity, Ecosystem Awareness.
- **Evaluation for Consultant: Crucial.** A consultant must understand the full enterprise landscape and articulate why a client should choose Python, Java, or Node.js for a given project.

2. Basic API Consumption and Concepts

- **Task:** Send a simple prompt (e.g., "Hey, what's up?") and a complex, opinionated prompt (e.g., "Ready to shoot the breeze on which LLM API is better—Gemini or OpenAI?") to an LLM and print the response.
- **2.1:** Clearly explain the difference between the legacy Completions endpoint and the modern **Chat/Completions** endpoint and why the latter is preferred.
- **2.2:** Implement the chat interaction using the **OpenAI Python SDK** but point it to the **Gemini API** (using the langchain-google-genai wrapper or similar). Explain the utility of this API standardization.
- **2.3:** Implement the same chat interaction using the **Gemini native SDK** to contrast the implementation.
- **Key Learning Area (KLA):** LLM API Fundamentals, SDK Usage, API Wrapper Utility, Conceptual Clarity.
- **Evaluation for Consultant: Foundational.** Essential to debug and justify architectural choices (e.g., "Why use a standardized wrapper vs. native SDK?").

3. LLM for Code Translation

Use an LLM (via its API or web interface) to translate the code from Exercise #2 (e.g., Python) into a different enterprise language (e.g., Java).

- **Key Learning Area (KLA):** Prompt Engineering for Code, Cross-Language Synthesis, Output Review.
- **Evaluation for Consultant: High Value.** Demonstrates the use of GenAI for accelerating development and prototyping in polyglot environments.

4. LLM for Code Refactoring and Analysis

Use a web-based LLM (Gemini, ChatGPT, or Claude) to analyze the code from Exercise #2. Request a specific, non-trivial refactoring (e.g., encapsulate the API call logic into a reusable class method).

- **Key Learning Area (KLA):** Code Analysis, Refactoring Skills, Critical Review of LLM Output.
- **Evaluation for Consultant: Essential.** A consultant must be a senior reviewer of LLM-generated code, ensuring best practices and maintainability.

5. LLM for Code Modification and Comparison

- **Task:** Provide a single code file (Python, Java, etc.) to an LLM.

- **Modification:** Request a simple change (e.g., adding explicit null/None checks) and a complex refactoring (e.g., break a >50 line method into cohesive smaller methods, ≤ 10 lines each).
- **Review:** Use a *different* LLM to compare and contrast the original and refactored code, focusing on clarity, adherence to the request, and potential side effects.
- **Key Learning Area (KLA):** Advanced Prompt Engineering, Code Review Automation, Verification Strategy.
- **Evaluation for Consultant: Core Consultant Skill.** Using an LLM to *verify* the output of another LLM introduces crucial double-checking for quality assurance.

6. LLM for Documentation/Explanation

Ask an LLM to generate professional documentation (e.g., Python pydoc, Java Javadoc, or C# documentation comments) for a given piece of code. Compare the quality, style, and completeness of the explanations from all three major LLMs (Gemini, Claude, and OpenAI).

- **Key Learning Area (KLA):** Documentation Standards, Comparative LLM Performance.
- **Evaluation for Consultant: Required.** Explaining code and generating consistent documentation is necessary for client handover and maintenance.

7. API Response Structure and Tokenization

- **Task:** Repeat Exercise #2, but this time, specifically track and print the **token consumption** for both the input prompt and the output response.
- **Exploration:** Analyze the full API response structure, identifying and explaining the purpose of key elements like choices, reasoning/logprobs (if available), and other **cost/latency metrics**.
- **Key Learning Area (KLA):** Cost Management, API Deep Dive, Performance Metrics.
- **Evaluation for Consultant: Critical for Business.** Consultants must accurately estimate and manage operational costs ($` \text{cost per call} = f(\text{tokens consumed})$$) and justify model choices based on latency and feature availability.

Phase 2: Web Scraping and Data Extraction (Tool Usage)

8. Headless Browser Automation with Playwright/Crawl4ai

Use crawl4ai (which leverages Playwright) to navigate to a target URL. Implement the following variations:

- Wait for the full page to load.
- Take a screenshot and save it as a PNG file.
- Save the complete page content as a PDF file.

Explore variations: **headless=True**, **headless=False** (visual check), using a **proxy** to load the URL, and using a **different browser engine** (e.g., Firefox instead of default Chromium).

- **Key Learning Area (KLA):** Web Automation, Headless vs. Headful, Network Configuration (Proxies), Tool/Library Integration.
- **Evaluation for Consultant: Essential for RAG.** Agents often need to *interact* with complex web applications, not just static HTML. This builds the foundation for tool use.

9. Targeted Web Crawling for External Links

Crawl a specific HTML webpage and extract **all external links** (e.g., links pointing to a domain *different* from the current page's domain). This requires robust link parsing and domain comparison.

- **Key Learning Area (KLA):** Data Pre-processing, URL Parsing, Information Filtering.
- **Evaluation for Consultant: Essential for RAG.** Needed to determine a knowledge source's boundaries and identify reliable, external sources for deep-dive analysis.

10. Traditional vs. LLM-Based Extraction

Implement Exercise #9 **without using an LLM** (e.g., using a traditional library like BeautifulSoup or Playwright's native DOM methods). Compare and contrast the effort, complexity, and final output with the LLM-based solution (if Exercise #9 used an LLM for extraction).

- **Key Learning Area (KLA):** Traditional Programming vs. LLM-Assisted, Efficiency Analysis.
- **Evaluation for Consultant: High Value.** A consultant must articulate the **Return on Investment (ROI)** of using an LLM for tasks that can be done traditionally.

Phase 3: Structured Data, Modality & Business Use Cases

11. Structured Data Extraction (JSON Schema)

Given a paragraph of unstructured text (e.g., a customer complaint about a service, a billing query), define a clear **JSON schema** for the desired output entity. Use an LLM to accurately extract the relevant data from the text into this specified JSON structure.

- **Key Learning Area (KLA):** Pydantic/JSON Schema, Structured Output, Reliability.
- **Evaluation for Consultant: Core Business Application.** This is a frequent requirement in enterprise automation (e.g., turning emails into structured CRM tickets).

12. Multi-Modal Input (Voice-to-Text)

Convert Exercise #11 to accept a **voice prompt** instead of a text prompt. Implement a **Speech-to-Text (STT)** interface (e.g., Whisper, Gemini Audio, or Azure Speech) to transcribe the audio, and then feed the resulting text into the LLM for JSON extraction.

- **Key Learning Area (KLA):** Multi-Modal Integration, API Chaining, Modality Layering.
- **Evaluation for Consultant: Crucial.** Many business processes start with non-textual input (calls, videos, etc.). Understanding STT/TTS is vital for comprehensive solutions.

13. Competitive Analysis from Web Data (RAG)

Load key offering details from the public websites of **two competitor offerings**. Compare and summarize the information based on defined key terms and benefits. **Assumption:** All necessary data is available on a single, easily crawlable page for each competitor.

- **Key Learning Area (KLA):** Prompting for Synthesis, RAG Preparation, Business Intelligence.
- **Evaluation for Consultant: Essential Consulting Skill.** A common requirement is rapid market/competitor analysis using internal/external knowledge bases.

14. Integrated Competitive Positioning

Extend Exercise #13 by adding data about **your own offering**. Perform a comparative analysis, highlighting your unique selling propositions (USPs) against the two competitors. **Assumption:** All data is still available on a single page per offering.

- **Key Learning Area (KLA):** Strategic Analysis, Comparative Prompting, Summarization.

- **Evaluation for Consultant: Direct Business Value.** This exercise validates the ability to turn raw RAG data into actionable business insights.

Phase 4: Stateful Agents with LangGraph

15. AgentState & StateGraph Conceptual Deep Dive

- **Task:** Study the **LangGraph** approach to implementing stateful agents.
- **Explanation:** Clearly explain how the three components—**AgentState**, the **Graph** (nodes and edges), and the **LLM** (as a decision maker/tool user)—work together to maintain state and execute a complex, multi-step process.
- **Key Learning Area (KLA):** State Management, Finite State Machines, Agent Orchestration.
- **Evaluation for Consultant: Advanced Agentic AI.** Understanding the **state machine pattern** is the distinction between simple LLM wrappers and robust, multi-step agents.

16. Fixed-Category Classification Agent

Create a State object and a LangGraph agent that runs until a specific termination prompt is received.

The graph must take a URL from the prompt, visit the page, and classify the content into one of four fixed categories: Object-Oriented Programming, C Programming, Go Programming, or MBA Degree Course. If none apply, the answer is "Oopsie!".

- **Key Learning Area (KLA):** Agent State Implementation, Conditional Routing (Edges), Tool Use (Web Crawling), Fixed Classification.
- **Evaluation for Consultant: Practical Agent Use Case.** This implements a content router—a common pattern in enterprise information flow.

17. Dynamic Taxonomy Creation Agent

Continue with Exercise #16, but remove the fixed categories. The task is now to analyze the webpage content and **dynamically generate a category/taxonomy label** that is consistent across different similar URLs. This mimics an **LLM expert** creating a coherent knowledge repository structure. The agent should also demonstrate an ability to **periodically re-analyze and update** the category based on content changes (simulating a change detection loop).

- **Key Learning Area (KLA):** Advanced Agent Reasoning, Dynamic Schema Generation, Consistency and Taxonomy Building, Asynchronous/Periodic Tasks (Conceptual).
- **Evaluation for Consultant: Expert-Level Agentic AI.** This is a highly valuable, real-world scenario for building self-organizing knowledge bases—a key deliverable for Agentic AI consultants.

18. Content Update

Take any existing material – a status report, a response to an RFQ, a presentation on a project, etc – and then enhance/improve through use of an LLM – spell check, converting from semi conversational or causal tone to a formal tone, segregating bigger chunks of data into smaller coherent and semantically meaningful paragraphs, bullets, etc.

- **Key Learning Area (KLA):** Effective use of major LLMs – Copilot, Chatgpt, Gemini, Claude, etc; upskilling self on conversion of data between various formats and prompt engineering to get the desired output. And explore niche LLMs and GenAI tools that focus on such tasks and compare and contrast vis a vis the major LLMs
- **Evaluation for Consultant: Expert-Level LLM User.** This is a highly valuable, real-world scenario for massaging existing material into context sensitive coherent whole.

19. Content Update - continued

Same as above but get the output to be an Executive Summary, 1 page highlight, etc.

- **Key Learning Area (KLA):** Effective use of major LLMs – Copilot, Chatgpt, Gemini, Claude, etc; upskilling self on conversion of data between various formats and prompt engineering to get the desired output
- **Evaluation for Consultant: Expert-Level LLM User.** This is a highly valuable, real-world scenario for compacting long material into key highlights – very usable for QBR, HBR, Senior Leadership Reviews, Elevator pitch, etc.

20. Generic Chatbot

Here it's a culmination of all the exercises above – a generic Chatbot that can handle any query – be it IT related or sports related or document update or code refactoring or generating a visual from a given prompt or reading an image and then producing the text contained within that image – and provide desired output in the expected format (both structured and unstructured)

- **Key Learning Area (KLA):** Combining various modes and abilities of an LLM/GenAI in multimodal input out
- **Evaluation for Consultant: Expert-Level LLM User.** This is a highly desirable, even must have skill for a consultant operating at the top tier on the abilities, and more importantly the limitations, of an LLM and API provider in GenAI space.

21. A local LLM

All of the above assumed that one has a remote LLM or API. The next step is to set up a local LLM that runs on one's laptop/desktop and use them over similar API. The local llm can be gpt-oss:20b or gemma3:4b or gemma3:12b etc. The higher the RAM CPU cores and preferably GPUs too, the better the performance of LLMs will be.

The best setup would be to have docker-ce on WSL (or) Docker Desktop (not on Infinite machines as that has to be licensed) and then try to run these models using ollama.

- **Key Learning Area (KLA):** Setting up an LLM in a laptop/desktop
- **Evaluation for Consultant: Expert-Level LLM User.** This is highly desirable as this eliminates needs for an API key; eliminates dependency on being online to consume those APIs; helps in writing functional code without depending on API before migrating to an API for higher parameter models and more advanced capabilities of those models.

22. With Embedding

An extensionof the 21 to use a model for chat/q&a and also to use an embeddingmodel to use RAG. Embedding models such as embeddginggemma