14-Day "KYC Bot" Agent Capstone Plan

Project: "KYC Bot" (Know Your Customer) Agent **Track:** Concierge Agents **Deadline:** 14 Days

This is an aggressive sprint plan optimized to score maximum points by focusing on the official evaluation criteria. It is built around the specific Codelabs and concepts from your 5-day intensive course.

The 100-Point Strategy

- **Week 1 (Days 1-7): The Code Sprint.** Goal: Secure the 70-point "Implementation" category (50 for code, 20 for docs). We will build the complete, functional agent and add clear comments.
- **Week 2 (Days 8-14): The Polish.** Goal: Secure the 30-point "Pitch" and 20-point "Bonus" categories. This week is dedicated *only* to the writeup, the video, and the submission.

The Pitch (To score 30 points)

- **Problem:** "Manual 'Know Your Customer' (KYC) checks are slow, repetitive, and prone to human error. Compliance officers spend 1-2 hours per new customer manually searching for adverse news, checking watchlists, and collating data, creating a costly bottleneck."
- **Solution:** "I am building the 'KYC Bot,' an autonomous agent that, given a customer's name, performs a multi-step investigation. It searches for adverse media, checks simulated watchlists, and analyzes the data to produce a risk report, reducing a 2-hour task to a 5-minute automated review."

Key Concepts We Will Target (To score 50+ points)

We will implement 3+ concepts, mapping them directly to your course Codelabs:

1. **Multi-agent system (Sequential):** A "Manager" agent will pass the task to a team of 3 specialist agents.
   - **Course Codelab:** `Day 3: Build an agentic ordering system in LangGraph.`
2. **Tools (Built-in):** Our `SearchAgent` will use Google Search.
   - **Course Codelab:** `Day 4: Use Google Search data in generation.`
3. **Tools (Custom):** Our `WatchlistAgent` will call a custom Python function.
   - **Course Codelab:** `Day 3: Talk to a database with function calling.`
4. **Sessions & Memory (State Management):** LangGraph's core state object will be used to pass data (the customer's file) between agents.
   - **Course Codelab:** `Day 3: Build an agentic ordering system in LangGraph.`
5. **Bonus: Use Gemini:** Our `AnalysisAgent` will use a Gemini model.
   - **Course Codelab:** `Day 1/Day 4 Codelabs.`

6. **Bonus: Deployment:** We will write a deployment strategy in our README.
    - **Course Codelab:** `Day 5: MLOps for Generative AI (Agent Starter Pack).`

Week 1: The Code Sprint (Goal: Secure 70 Points)

Day 1 (Mon, Nov 17): Project Setup & Pitch

- **Goal:** Set up your project and lock in the "Pitch" (15 pts).
- **Actions:**
    1. Create your public **GitHub Repository** (or Kaggle Notebook).
    2. Set up your local Python environment (e.g., `venv`) and install `LangGraph`, `langchain-google-genai`, etc.
    3. Create your `README.md` file.
    4. **CRITICAL:** Write the first draft of your "Pitch" (Problem, Solution, Value) directly into the `README.md`.

Day 2 (Tue, Nov 18): Agent 1 - The `SearchAgent`

- **Goal:** Build the first agent and use a built-in tool.
- **Actions:**
    1. Create your `SearchAgent` node for your LangGraph.
    2. **Task:** Implement this using the concepts from your **Day 4 Codelab** (`Use Google Search data in generation`).
    3. This agent's job: Take a name (e.g., "John Doe") and a Gemini model to generate 3-5 search queries (e.g., "John Doe fraud," "John Doe sanctions"), then execute those queries using the Google Search tool.
    4. It should return a list of search snippets.

Day 3 (Wed, Nov 19): Agent 2 - The `WatchlistAgent`

- **Goal:** Build the second agent and a custom tool.
- **Actions:**
    1. Create a simple Python function `check_watchlist(name: str) -> bool`.
    2. Inside this function, have a hardcoded list: `RISK_LIST = ["BadGuy Inc", "Shady Corp"]`. The function returns `True` if the name is in the list.
    3. **Task:** Implement the `WatchlistAgent` node using concepts from the **Day 3 Codelab** (`Talk to a database with function calling`) to register your `check_watchlist` function as a **Custom Tool**.

Day 4 (Thu, Nov 20): Agent 3 - The `AnalysisAgent`

- **Goal:** Build the reasoning/summary agent.
- **Actions:**

1. Create the `AnalysisAgent` node.
2. Its job: Take the search snippets and the watchlist status as input.
3. **Task:** Use concepts from the **Day 1 Codelab (** `Evaluation and structured data` **)** to prompt the Gemini model to return a structured JSON output: `{"risk_score": "Low/Medium/High", "summary": "..."}` .

Day 5 (Fri, Nov 21): The `Manager` - Connecting the Team

- **Goal:** Get all three agents working together in a sequence.
- **Actions:**
  1. **Task:** This is the most important day. Use the **Day 3 Codelab (** `Build an agentic ordering system in LangGraph` **)** as your primary template.
  2. Define your graph's `AgentState` (e.g., a `TypedDict` that holds `customer_name` , `search_results` , `watchlist_status` , and `final_report`).
  3. Define the graph, add your 3 agent nodes, and set the `sequential` edges to make them run in order.

Day 6 (Sat, Nov 22): Testing & Code Cleanup

- **Goal:** Ensure the code works and is readable for the judges.
- **Actions:**
  1. Create 3-5 test cases (e.g., a "clean" name, a "high-risk" name). Run them.
  2. **CRITICAL:** Add code comments! (Part of the 50 pts). Explain *why* you're doing things, especially where you are using the Key Concepts (e.g., `"# This is our Custom Tool, as learned in Day 3"`).

Day 7 (Sun, Nov 23): Buffer Day & Deployment "Writeup"

- **Goal:** Finish all coding and get the "Deployment" bonus.
- **Actions:**
  1. Catch up on any unfinished code from Week 1.
  2. **(Bonus: Agent Deployment - 5 pts):** In your `README.md` , write a new section titled "Deployment Strategy."
  3. **Task:** Reference the **Day 5 MLOps Codelab**. Explain *how* you would deploy this agent (e.g., "This agent is designed to be deployed on Google Cloud Run, packaged as a Docker container. We would use the 'Agent Starter Pack' as a template to manage the API server..."). This gets you the 5 bonus points without any complex setup.

Week 2: The Polish (Goal: Secure 50+ Points)

Day 8 (Mon, Nov 24): README - Architecture

- **Goal:** Fulfill the `Writeup` (15 pts) and `Documentation` (20 pts) criteria.

- **Actions:**
    1. Refine your "Pitch" from Day 1. Make it perfect.
    2. Create a simple architecture diagram (you can use draw.io or a text-based one).
    3. Add the diagram to your `README.md` and write the "Architecture" section explaining your LangGraph flow.

Day 9 (Tue, Nov 25): README - Setup & Key Concepts

- **Goal:** Make it effortless for the judge to give you full points.
- **Actions:**
    1. Create your `requirements.txt` file (from `pip freeze > requirements.txt`).
    2. Write crystal-clear "How to Run" instructions in the `README.md`.
    3. **CRITICAL:** Add a section to your README titled "**Key Concepts Used**" and explicitly list them with links to your code (e.g., "1. **Multi-Agent System:** See `main.py`, line 42, where our LangGraph state machine is defined...").

Day 10 (Wed, Nov 26): Bonus Video - Script & Record

- **Goal:** Get the 10-point video bonus.
- **Actions:**
    1. Write a simple 2-minute script covering the 5 required points (Problem, Why Agents, Architecture, Demo, Build).
    2. Record your screen (use OBS or QuickTime). Just run the agent on your test cases and show the final report.

Day 11 (Thu, Nov 27): Bonus Video - Edit & Upload

- **Goal:** Finish the video. (Light task day).
- **Actions:**
    1. Use a simple editor (Clipchamp, iMovie) to trim your video.
    2. Upload to YouTube as "Unlisted".
    3. Add the video link to your `README.md` and get it ready for the submission form.

Day 12 (Fri, Nov 28): Final Review

- **Goal:** Pretend to be the judge.
- **Actions:**
    1. Read the *entire* Kaggle prompt again, line by line.
    2. Read your `README.md` from top to bottom. Does it check every single box?
    3. Make sure your GitHub repo is clean and all code is pushed.

Day 13 (Sat, Nov 29): SUBMIT DAY (Early)

- **Goal:** Submit *before* the deadline. **Do not wait until the last day.**

- **Actions:**
    1. Fill out the Kaggle submission form.
    2. Add your GitHub repository link.
    3. Add your YouTube video link.

Day 14 (Sun, Nov 30): Relax

- **Goal:** You're done. You've submitted a high-quality project with time to spare.