Project Title: Intelligent Multi-Modal Media Crawler & Curation Agent

Duration: 4 Months | **Role:** AI Engineer | **Stack:** LangGraph, Python, GCP

1. Executive Summary

The objective is to build a scalable, autonomous agentic system capable of navigating complex, dynamic websites (e.g., Pinterest, Instagram, ArtStation), extracting media (Image/Video/Audio), and curating this content based on complex natural language prompts.

Unlike a standard scraper, this system must utilize **LangGraph** to orchestrate stateful agents that can reason about navigation, handle anti-bot measures, and perform multi-stage semantic filtering to ensure high precision with optimized costs. The final solution must be deployed as a production-ready microservice architecture on **Google Cloud Platform (GCP)**.

2. Core Business Use Case (The "North Star")

The Scenario: A user provides a target URL (e.g., a Pinterest board) and a complex natural language filter: *"Find all tattoo designs featuring Japanese Irezumi style that specifically contain Maple leaves or Cherry Blossoms, but exclude koi fish."*

The Goal: The agent must autonomously traverse the site, handle infinite scrolling, extract media, and return a structured, clean dataset that matches the prompt with >90% precision.

3. Technical Requirements & Architecture

A. The Agentic Core (LangGraph)

The candidate must demonstrate seniority by architecting a **Stateful Multi-Agent System** rather than a linear script.

- **Navigation Agent:** Responsible for DOM parsing, identifying "next page" buttons or infinite scroll triggers, and managing the crawl frontier. Must handle dynamic rendering (Selenium/Playwright).
- **Extraction Agent:** Identifies media assets, standardizes formats, and manages download queues.
- **Curator Agent (The Brain):** The core reasoning engine.
  - **Level 1 Filter (Fast/Cheap):** Uses CLIP or lightweight embedding models to discard obviously irrelevant content.

    - **Level 2 Filter (Slow/Accurate):** Uses Large Vision Models (e.g., GPT-4o, Gemini 1.5 Pro) for nuanced understanding (e.g., distinguishing "Maple leaf" from "Cannabis leaf" or "Japanese style" from "American Traditional").
- **Supervisor Node:** Manages the state, retries, and hands off tasks between agents.

B. Infrastructure & Deployment (GCP)

- **Compute:** Deploy agents on **Cloud Run** (Serverless) or **GKE** (if heavy persistence is needed).
- **Storage:**
    - **GCS (Google Cloud Storage):** For raw media assets.
    - **Firestore:** For storing agent state, crawl history, and metadata.
    - **Vector Database (Vertex AI Vector Search):** Store embeddings of the downloaded images to allow for future semantic search capabilities.
- **Event-Driven Architecture:** Use **Pub/Sub** to decouple the crawling (fast) from the analysis (slow/expensive).

4. Extensions (Success Criteria)

To prove capability, the candidate must implement the following "Production-Grade" features:

1. **Cost-Aware Intelligence:** The system must not send *every* image to a costly VLM (Vision Language Model). Implementation of a "Waterfall Filtering" mechanism is required (Metadata check -> Embedding Similarity -> VLM Verification).
2. **Anti-Fragility:** Handling IP bans and DOM changes. Integration of rotating proxies and user-agent rotation. The system should degrade gracefully (e.g., if the VLM API fails, pause and retry, don't crash).
3. **Human-in-the-Loop (HITL):** Using LangGraph's checkpointing features, implement a pause state where the agent asks for human clarification if confidence scores are borderline (e.g., *"I found a tattoo with a leaf, but I am unsure if it is a Maple leaf. Proceed?"*).
4. **Observability:** Full integration with **LangSmith** or **Phoenix** to trace agent reasoning steps, token usage, and latency per node.

5. Implementation Roadmap

Month 1: The Skeleton & LangGraph Logic

- **Goal:** A locally running LangGraph prototype that can navigate a static page and use a VLM to classify an image.
- **Deliverables:**
    - Design Document (Architecture Diagram).
    - LangGraph State definition (Nodes/Edges).
    - Basic Scraper tool definition.

Month 2: The Infrastructure & GCP Deployment

- **Goal:** Moving to the cloud. Decoupling crawling from processing.
- **Deliverables:**
    - Terraform/IaC scripts for GCP environment.
    - Deployment of agents to Cloud Run.
    - Integration of Pub/Sub for asynchronous processing of images.

Month 3: Optimization & "The Senior Touch"

- **Goal:** Making it smart and cost-effective.
- **Deliverables:**
    - Implementation of the Waterfall Filtering (CLIP + Gemini/GPT).
    - Vector Database integration for duplicate detection and semantic search.
    - Anti-bot handling (Headless browser hardening).

Month 4: Robustness, UI & Handover

- **Goal:** Production readiness and User Experience.
- **Deliverables:**
    - A simple Streamlit or React frontend to input URL and Prompt, and view results.
    - **Evaluation Pipeline:** A script that runs the agent against a "Golden Dataset" to calculate Precision/Recall metrics.
    - Final Documentation and Code Handoff.

6. Evaluation Metrics

The project will be judged on:

1. **Architecture:** Is the LangGraph state managed cleanly? Is the GCP architecture scalable?
2. **Code Quality:** Type hinting, modularity, unit tests, and docstrings.
3. **Accuracy:** Does it actually filter "Japanese style" correctly versus generic tattoos?
4. **Engineering Maturity:** How does the system handle failures? (e.g., Network timeouts, API limits).

Specific Challenge for the Candidate:

*"In your design, explicitly demonstrate how you handle the trade-off between **Scraping Speed** and **Rate Limits**. If Pinterest blocks your IP after 50 requests, how does your LangGraph agent handle the exception, rotate the proxy, and resume the graph exactly where it left off without losing the state?"*