Project Kinetograph: Autonomous Multi-Agent Video Orchestration Engine

Target Event: Mistral Worldwide Hackathon – New York Edition **Primary Objective:** Mistral Global Winner / Enterprise Tooling

1. Executive Summary

Project Kinetograph is not a video generator; it is an enterprise-grade autonomous post-production studio. Traditional AI video tools lack deterministic control and fail in professional workflows. Kinetograph solves this by deploying a **Stateful, Event-Driven Multi-Agent Swarm** powered by Mistral's ecosystem.

By separating the cognitive tasks of video editing (ingestion, scripting, cutting, and QA) into specialized AI agents that communicate via a central message broker, the system can ingest raw A-roll/B-roll, construct a narrative, and programmatically export a fully edited, timeline-ready video file without human intervention—save for a single, high-leverage narrative approval step.

2. Core Architecture: The Event-Driven State Machine

Kinetograph abandons linear, brittle scripts in favor of a **Message Broker architecture** (e.g., LangGraph or a local Redis instance).

- **The Master State:** A central `Project_State.json` file holds the entire context of the video (transcripts, tags, paper edit, timeline math).
- **Event-Driven Execution:** Agents do not call each other directly. They subscribe to the message broker. When the `Project_State` updates, the broker wakes the specific agent required for the next phase. If an agent fails, the state does not corrupt; the system catches the exception and attempts a new approach.

3. How it Works: The Human-Swarm Workflow

Project Kinetograph operates as a "Human-in-the-Loop" system, ensuring the human acts as the Creative Executive while the AI swarm handles the manual labor.

Here is the step-by-step operational flow:

Step 1: The Asset Drop (Human Action) The user begins by dragging and dropping folders of raw media—A-roll (talking heads) and B-roll (scenery, action shots)—into a designated local directory.

Step 2: The Deep Parse (Autonomous) The **Archivist Agent** instantly wakes up. Without any prompting, it uses Python to strip the audio and pull 1-frame-per-second visual samples from the raw files. It silently passes these assets to the Mistral transcription and Pixtral vision models, generating a master JSON index that maps exactly what happens, and what is said, at every single millisecond of the footage.

Step 3: The Creative Prompt (Human Action) With the footage indexed, the user enters a single natural language command into the terminal or a lightweight web UI.

- *Example:* "Create a fast-paced, 60-second hype reel about AI startups. Cut out all the dead air, and overlay B-roll of the city whenever the speaker mentions 'infrastructure'."

Step 4: The Paper Edit (Autonomous) The **Scripter Agent** (powered by Mistral Large 3) takes that prompt and analyzes the massive JSON index. Instead of rendering a video blindly, it outputs a "Paper Edit"—a structured text timeline of the proposed video, detailing exactly which clip will play at which second.

Step 5: The Greenlight (Human-in-the-Loop) The swarm pauses. The user reviews the generated Paper Edit on their screen. If the AI chose a B-roll clip the user doesn't like, they simply delete that line of text or swap it out. Once satisfied with the narrative structure, the user clicks **"Approve Execution."**

Step 6: Asset Synthesis (Autonomous) If the approved script requires a shot that wasn't in the original raw folders, the **Synthesizer Agent** triggers a background script. It utilizes `axios` to execute clean, promise-based HTTP requests to external stock footage APIs (like Pexels), downloading the missing contextual B-roll directly into the project folder.

Step 7: The Mathematical Render (Autonomous) The **Director Agent** and the **Sound Engineer Agent** take over. They translate the approved text script into deterministic Python commands (`MoviePy` / `FFmpeg`). The system mathematically slices the clips, stitches them together, tracks the faces to keep them centered, and ducks the background music whenever someone is speaking.

Step 8: The Final Output Within minutes, the system outputs a fully rendered, polished `.mp4` video file, alongside an exported XML timeline file that can be dragged directly into Premiere Pro or DaVinci Resolve for manual color grading if desired.

---

4. The 8-Agent "Hollywood Swarm" Pipeline

Phase 1: Pre-Production (Data Ingestion & Indexing)

- **Agent 1: The Archivist**
  - **Function:** Ingests raw video/audio files. Uses Python `FFmpeg` wrappers to extract the audio track and pull frames at 1-second intervals.
  - **AI Integration:** Sends audio to Mistral's transcription endpoints and frames to Pixtral (vision).
  - **Output:** Generates a mathematically precise JSON index mapping exact timestamps to spoken words and visual context.

Phase 2: Production (The Narrative Engine)

- **Agent 2: The Scripter**
  - **Function:** Reads the user prompt (e.g., *"Create a 60-second fast-paced hype reel"*) and the Archivist's index.
  - **AI Integration:** Mistral Large 3 drafts the story arc, outputting a highly structured "Paper Edit" (a JSON Directed Acyclic Graph dictating the sequence).
- **Agent 3: The Producer (Human-in-the-Loop)**
  - **Function:** The swarm pauses execution. The user reviews the generated JSON Paper Edit on a lightweight dashboard, tweaks any necessary lines, and clicks "Approve."
- **Agent 4: The Synthesizer**
  - **Function:** Identifies visual gaps in the approved script where user-provided B-roll is insufficient.
  - **Execution:** Utilizes `axios` within isolated Node.js microservices or testing scripts to execute clean, promise-based HTTP requests to external stock footage APIs (like Pexels) to autonomously pull the missing assets.

Phase 3: Post-Production (The Assembly)

- **Agent 5: The Director**
  - **Function:** Translates the approved script and pulled assets into hard math.
  - **Execution:** Relies on Python libraries like `MoviePy` to deterministically trim the video files, stitch the A-roll and B-roll, and align the visual cuts to the audio transcript timestamps.
- **Agent 6: The Motion Grapher**
  - **Function:** Adds kinetic typography.
  - **Execution:** Uses `OpenCV` to track the speaker's face and generates dynamic, word-by-word highlighted text overlays that never block the subject.

Phase 4: Mastering (Polish & Self-Correction)

- **Agent 7: The Sound Engineer**
  - **Function:** Automates the audio mix. Normalizes dialogue to standard LUFS and programmatically calculates "Audio Ducking"—lowering the background music volume exactly when the speaker is talking.
- **Agent 8: The QA Lead**
  - **Function:** The autonomous critic. It feeds the rendered `mp4` output back into Pixtral and the audio models to grade it against the original prompt. If text is cut off or the music is too loud, it flags the `Project_State` with an error and routes the task back to the Director.

5. Technical Stack & Prerequisites

- **Language:** Python (Primary Backend / ML Orchestration).
- **Orchestration Framework:** LangGraph (for multi-agent state management).
- **AI Models:** Mistral Large 3 (Logic/Director), Pixtral (Vision/QA), Ministral Edge (Local parsing), Mistral Vibe 2.0 (Terminal automation).
- **Media Manipulation:** `FFmpeg-python`, `MoviePy`, `OpenCV`.
- **API Interactions:** `axios` (Strictly used for any external HTTP requests to ensure reliable promise resolution and interceptor management).