aiVOLUTION 2025 - Idea Submission Template

Title

🎬**MediaGraphAI: A Multimodal Knowledge Graph for Movies**

Overview (max 300 words)

Today, entertainment discovery is limited to keywords, genres, or ratings (e.g., “action movies” or “movies by Tolkien”). But people often think in richer, story-driven terms:

> *“Show me movies where the hero wears green pants and is betrayed by their best friend.”*

MediaGraphAI solves this by building an **AI-powered multimodal knowledge graph** that ingests movies to extract **fine-grained entities** (characters, objects, themes, emotions, scenes) and their relationships. Using **AI models for text, vision, and audio**, we convert raw scripts, subtitles, and clips into structured graph data.

Core features:

- Natural language search across movies.
- Scene-level tagging (actions, moods, objects).
- Semantic similarity queries (e.g., “movies that feel like Spirited Away”).
- Knowledge graph visualization for exploration.

The result is **a smarter IMDb + Wikipedia mashup**—a tool for discovery, analysis, and creative research.

AI/LLM Use Case

AI powers every layer of the system:

- **Text (Scripts/Books):** spaCy + GPT-4/gemini for entity/relation extraction.
- **Vision:** CLIP + YOLO for scene/object tagging.
- **Audio:** Whisper for ASR to extract dialogues.
- **LLM Querying:** GPT/gemini translates natural language into graph queries (Cypher/GraphQL).
- **Embeddings:** Vector search for “vibe” or thematic similarity.

LLMs act as the “semantic glue” connecting symbolic graph data with fuzzy, human-style queries.

Feasibility Plan

Defined Scope:

We will focus only on the smallest working slice of the project that still demonstrates the core idea:

1. **Data Ingestion**
    - Top 50 movies (via TMDB + OpenSubtitles).
2. **Entity Extraction**
    - Use **spaCy + GPT/Gemini API** to extract characters, locations, events, add themes from subtitles and book text.
3. **Graph Storage**
    - Store entities and relationships in **Neo4j** with a simple schema.
    - Query via Cypher/GraphQl.
4. **Minimal UI**
    - React frontend with a **search box** and **graph visualization** (Neo4j Browser integration or lightweight D3.js).
5. **Demo Queries**
    - “Find movies about betrayal.”
    - “Books where Alice is the protagonist.”

Outcome: A working prototype (text-first Knowledge Graph + query interface) that is demo-ready.

Why Achievable in 8–10 Hours:

- **Tightly Scoped**: We limit to *text-only pipeline* (no video/audio analysis in MVP).
- **Use of Existing APIs**: TMDB, OpenSubtitles, Project Gutenberg → no manual dataset prep.
- **Pre-built NLP Models**: SpaCy + GPT/Gemini for NER and summaries → no training required.
- **Off-the-shelf Graph DB**: Neo4j has quick setup and built-in visualization → avoids custom infra.
- **Minimal Frontend**: Only search + graph visualization, no complex UI features in MVP.

Team Allocation (3 Members):

- **Frontend + Integration (1 member)**
- **Backend + AI/LLM (2 members)**

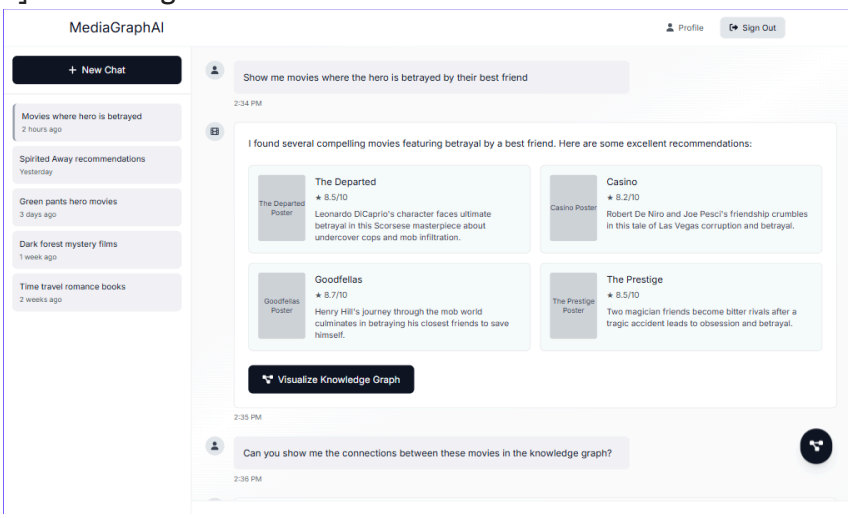
Constraints & Risk Management:

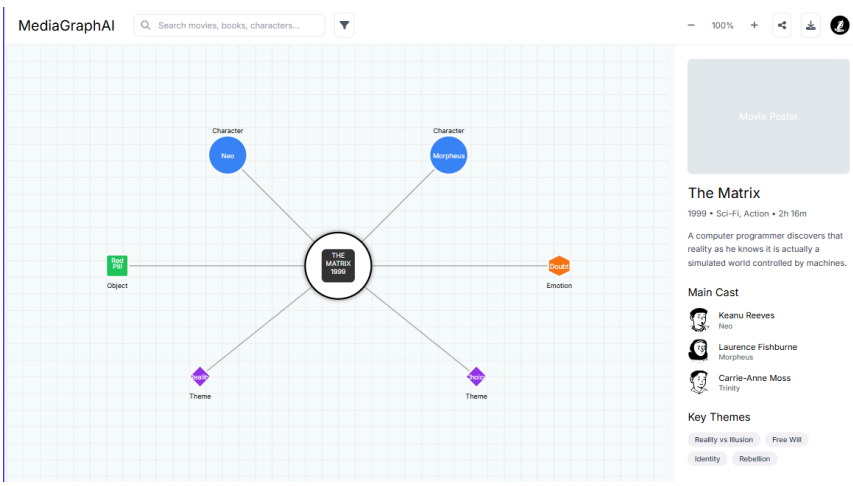
- **Time Constraint**: 8–10 hours → only core features, leave advanced multimodal analysis for Iterations 2 & 3.
- **API Dependence**: Rely on public/free APIs (TMDB, OpenSubtitles, Gutenberg). If API limits hit → fall back to cached samples.
- **LLM Cost/Latency**: Keep GPT calls limited to short summaries/NER, not entire scripts.

Conclusion: With scoped-down objectives, proven APIs, and a clear team split, our MVP is **realistically achievable in 8–10 hours**, with optional Iterations 2 & 3 for polish and wow-factor if time allows.

Final Product UX

1] UI Designs:

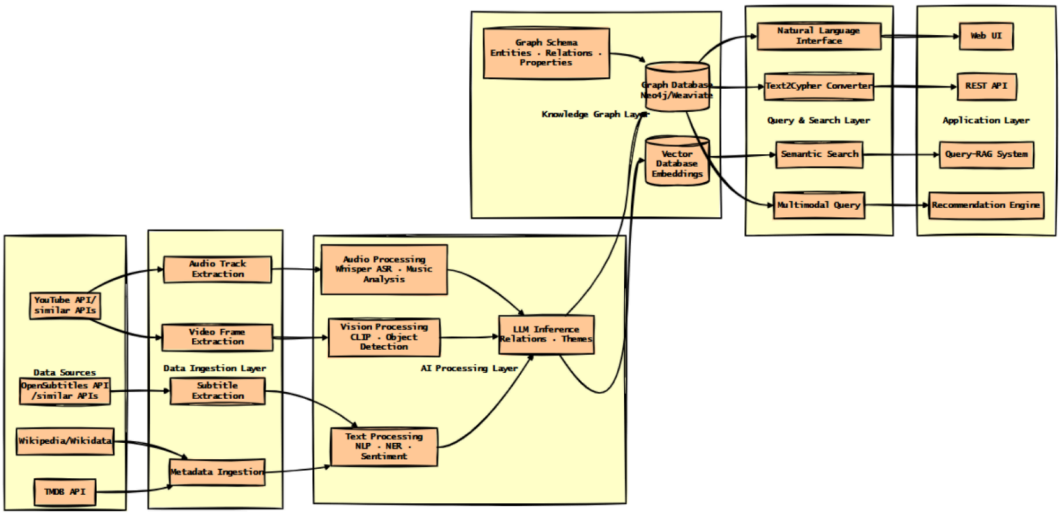




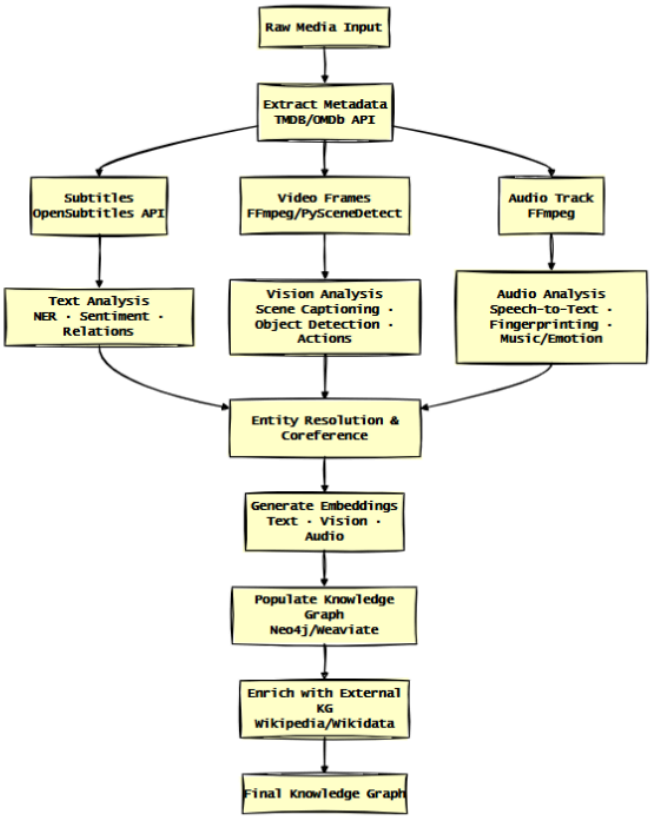
2]Do check our Full UI Collection (Clickable Prototype), made using UX Pilot: [Complete Clickable Prototype]

3]System Flowcharts:

I] System Architecture-



II] Data ingestion flowchart-



Minimum Viable Product (MVP) Definition

Core Functionalities (Hackathon Priority – 8–10 hrs)

1. **Data Ingestion**
    - Import top 50 movies (TMDB + OpenSubtitles).
2. **Entity Extraction**
    - Use spaCy + GPT/Gemini API to extract key entities: characters, locations, events, and themes.
3. **Graph Storage & Query**
    - Store extracted entities and relationships in Neo4j.
    - Enable querying through Cypher.
4. **Minimal UI**
    - A simple React search box for natural language queries.
    - Basic graph visualization to display entity relationships.

Outcome: A functional knowledge graph where users can run queries like *"Movies with betrayal theme"* or *"movies where Alice is the protagonist."*

Additional Features (If Time Allows)

Iteration 2 – Enrichment & Polish (+2–3 hrs)

- **Data:** Add trailers for 1–2 movies via YouTube API.
- **Extraction:** Include emotional themes (sentiment analysis/GPT tags); run Whisper on a short clip to generate transcripts.
- **Graph:** Add scene-level nodes for 1–2 movies (chapters/scenes).
- **UI:** Pre-built query buttons (e.g., "Find betrayals," "Find sad scenes"); show results as cards with posters/snippets.
- **Demo Queries:** "Show me sad fight scenes" or "Which movies feel similar to Spirited Away?"

Outcome: A richer, more polished graph with emotional themes, scene-level detail, and improved usability.

Iteration 3 – Stretch / Wow Features (+2–3 hrs)

- **Vision:** Apply CLIP on frames to auto-tag objects/clothing; detect 1–2 iconic fight scenes via pretrained action classifiers.
- **Audio:** Prototype "Shazam for movies" by matching audio clips with Whisper transcripts.
- **UI Polish:** Add scene timeline with clickable nodes; allow screenshot upload → return closest scene.
- **Demo Queries:** "Find movies with rainy fight scenes" or "Upload an image to find the matching scene."

Outcome: A multimodal demo combining text, audio, and vision for a strong wow factor.
