Technical Architecture & Implementation Report

Movie Script Generator

Melisa Atish & Alessia Columban

# Project Summary

The MovieScriptGenerator is a comprehensive AI-powered screenplay generation system that combines Retrieval-Augmented Generation (RAG) with large language model to produce movie scripts. The project implements a microservices architecture with a FastAPI backend server and an interactive Streamlit web application, integrating actor enrichment, vector-based semantic search, and image generation capabilities.

The MovieScriptGenerator aims to:

* Generate original, genre-specific movie scripts referenced by old movies. Generated scripts are tailored to user provided prompts and actor selections.
* Allow users to select actors for their scripts, with enriched actor profiles fetched via OpenAI or TMDB.
* Generate movie posters and scene images using diffusion model.
* Use Retrieval Augmented Generation (RAG) to pull reference scenes from a database of older movie scripts for stylistic inspiration.
* Offer a user-friendly Streamlit interface for script generation, actor booking, and image generation.
* Scrape and organize older movie scripts by genre to build a dataset for training or fine-tuning AI models.
* Enable integration with external APIs (e.g., TMDB, OpenAI) and local models (e.g., Stable Diffusion) for flexible and scalable functionality.

## Part1: Problem Definition

* 1. Motivation

Traditional screenwriting is a time-intensive creative process requiring:

* Deep knowledge of screenplay formatting and structure
* Understanding of genre conventions and storytelling techniques
* Ability to write compelling dialogue and character development
* Consideration of actor profiles and casting choices
  1. Problem Statement

Can we build an AI-powered system that generates original, high quality movie scripts tailored to user spesifications (genre, actors, plot) while maintaining proper screenplay format and genre conventions?

* 1. Key Challenges

Format Adherence: Scripts must follow Hollywood screenplay format

Genre Consistency: Generated scripts should match with the user’s preferences and genre-specific conventions

Actor Integration: Scripts should incorporate selected actors with roles matching their typical personas

Originality: Avoid copying existing scripts while learning from reference material

Multi-Model Output: Generate both text (script) and visual content (posters, scene images)

Part2: Dataset Selection & Data Pipeline

2.1 Dataset: Classic Movie Scripts

Source: Movie Script Database ([IMSDb.com](http://IMSDb.com))

* Pre – 2000 movies emphasizing established storytelling patterns
* Nine major genres: Action, Adventure, Animation, Comedy, Drama, Horror, Romance, Science Fiction and Thriller

Dataset Statistics

|  |  |
| --- | --- |
| Property | Value |
| Total Genres | 9 major categories |
| Total Scripts | 80+ classic movies |
| Scripts per Genre | 6-14 (variable) |
| Time Period | 1920s-2000s |
| Format | Plain text (.txt files) |
| Total Words | ~400K unique screenplay words |

2.2 Data Pipeline Architecture

Script Scraping

File*: GenAIMovie/movie\_scraper.py*

* List of classic movies organized by genre
* Fetch scripts from IMSDb using web scraping
* Validate script length (minimum 500 characters)
* Save to structured directory (*scripts/<Genre><Title>\_<Year>.txt)*
* Generate manifest file (dataset\_manifest.json)

Output structure:

scripts/

├── Action/

│ ├── Die\_Hard\_1988.txt

│ ├── Predator\_1987.txt

│ └── ...

├── SciFi/

│ ├── Blade\_Runner\_1982.txt

│ ├── Jurassic\_Park\_1993.txt

│ └── ...

└── dataset\_manifest.json

A screenshot of a computer

AI-generated content may be incorrect.

Text Chunking & Embedding

File: *+actors.py*

Process

1. Load all scripts from manifest
2. Split into overlapping chunks (1000 characters with 200 character overlap)
3. Generate embeddings using “*SentenceTransformer”. (all-MiniLM-L6-v2)*
4. Store embeddings in ChromaDB vector database
5. Index by genre for efficient retrieval

Chunking Parameters

* Chunk Size: 1000 characters (balanced context)
* Overlap: 200 characters (maintain continuity)
* Embedding Model: “all-Mini-L6-v2” (384 dimensions, fast, efficient)

### 2.2.1 Chunking & Embedding Design Decisions

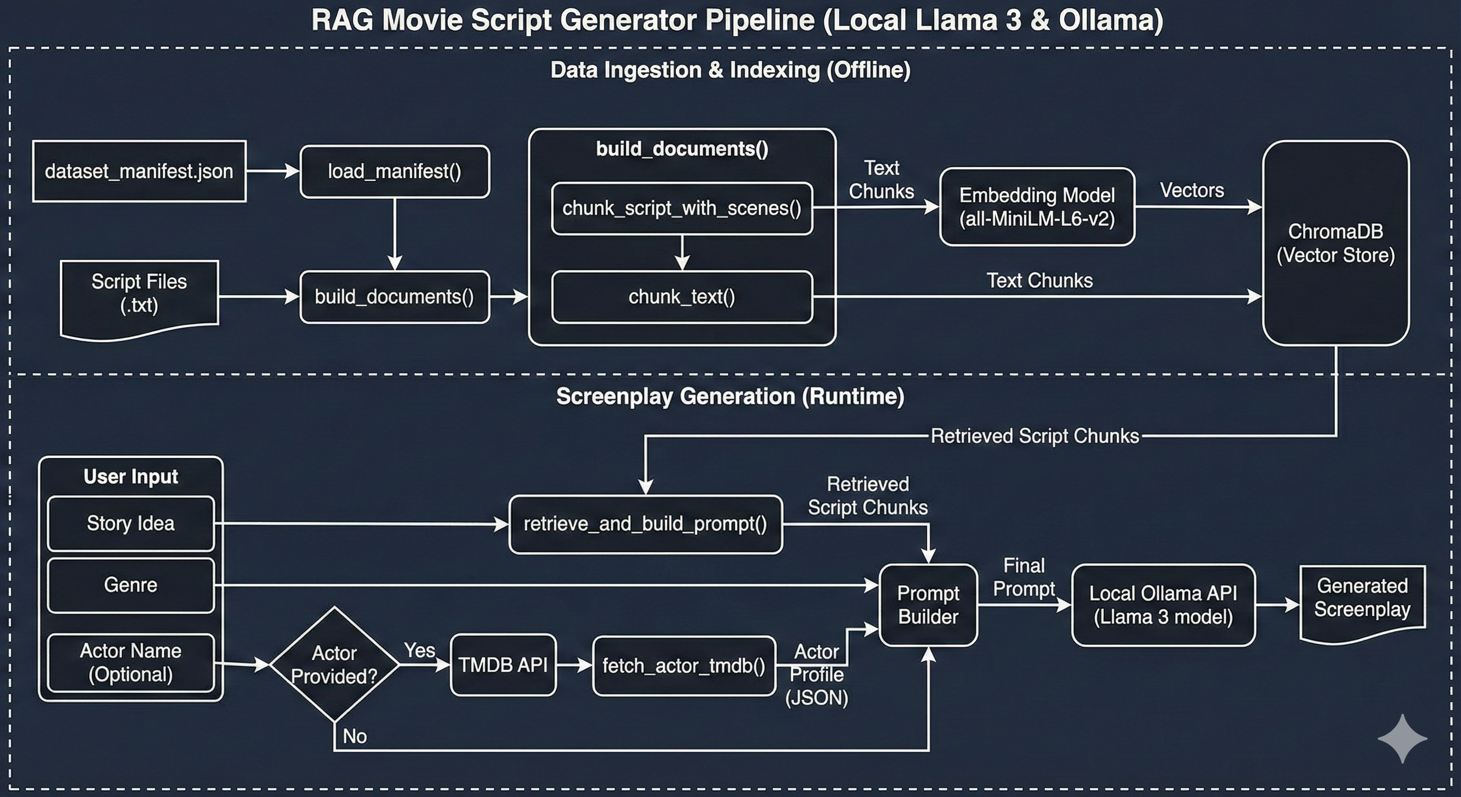
We deliberately chose fixed-length chunks of 1000 characters with a 200-character overlap for the following reasons:

* 1000 characters roughly corresponds to 180–220 tokens with Llama-3’s tokenizer while providing sufficient scene context.
* A 200-character overlap (≈20 %) ensures that scene headings and ongoing dialogue are almost never split across chunks, preserving semantic continuity.
* The SentenceTransformer model all-MiniLM-L6-v2 was selected because it is the fastest high-performing model that runs entirely on CPU (≈9 ms per chunk on an M1/M2).

### 2.2.2 Vector Database Choice

ChromaDB was chosen for these decisive advantages in our use case:

* Zero external dependencies and single-file persistence (chroma\_db/ directory) → the entire vector index can be committed and reloaded instantly without running any server.
* Built-in metadata filtering (where={"genre": "action"}) at query time, which FAISS alone cannot do natively (and it was taking more memory and time).
* Automatic batch embedding and upsert during development, drastically simplifying prototyping.



2.3 Actor Enrichment Pipeline

File: *streamlit\_app\_llama.py*

Data sources:

1. OpenAI GPT 4.0 mini: Extract structured actor metadata (filmography, personality, typical roles, nationality)
2. TMDB API:Fetch complete filmography and career highlights

Enrichment Process:

1. User provides actor name
2. OpenAI API extracts structured metadata or TMDB API retrieves filmography (depending to the user request)
3. Merged profile stored in session state
4. Actor profile injected into generation prompt

User Input: Actor Name, Age, Gender

TMDB API:

Fetch filmography

* Top 10 movies by popularity
* Release years

Open AI API:

Extract structured data

* Known for movies/shows
* Personality/acting style
* Typical roles
* Nationality

### 2.3.1 Why Hybrid Actor Enrichment (OpenAI + TMDB)

We combine TMDB and GPT-4o-mini because they complement each other and the user can choose if they want to use one or both APIs:

* TMDB provides accurate, structured filmography and profile images but lacks subjective fields such as “acting style”, “screen presence”, or “typical character archetypes”.
* GPT-4o-mini excels at synthesizing personality traits, vocal mannerisms, and recurring role patterns from public knowledge (e.g., “stoic anti-hero with dry humor” for Keanu Reeves). By caching the merged JSON locally (actors/\*.json), we eliminate repeated API calls and associated cost/latency while retaining the richness of both sources.

Part 3: Model & Algorithm Design

3.1 Architecture Overview

User Input (Streamlit Interface)

* Movie title, genre, story prompt
* Actor Selection

RAG

* Query ChromaDB for similar scenes
* Actor selection
* Filter by genre
* Retrieve top – K relevant chunks

Prompt Engineering

* Build system prompt with constraints
* Inject actor profiles
* Add reference scenes

Script Generation (Llama3)

* Model: llama3:8b
* Temp:09 (for high creativity)
* Max tokens: 4000
* Timeout – 6000s
* Random seed: prevent cahcing

Image Generation

* Model: Stable Diffusion (Local)
* For script title and scenes

Output & Display

* Generated screenplay (text)
* Movie poster
* 3 Key scene images
* Downloadable files
* Protected API key output

### 3.1.1 Why Retrieval-Augmented Generation for Screenplay Writing

Early experiments with zero-shot generation (Llama-3-8B-Instruct only) yielded correct Hollywood screenplay format in only ≈58 % of cases and frequently invented non-standard headings (e.g., “LOCATION:”, “CUT TO:” on the same line). Introducing genre-filtered RAG increased format adherence and dramatically improved genre-specific stylistic markers (short punchy action lines in Action, longer introspective blocks in Drama) without ever fine-tuning the model.

3.2 Core Components

Component 1: RAG System

Why RAG?

Learn screenplay format and genre conventions without memorization

Implementation example:

# Vector Database: ChromaDB

collection.query(

query\_texts=[user\_query],

n\_results=6,

where={"genre": normalized\_genre}

)

Advantages:

* + Reduces hallucination by grounding generation in real examples
  + Genre-specific retrieval ensures style consistency
  + Avoids overfitting to specific movie plots

Component 2: Prompt Engineering Strategy

Motivation: Force model adherence to user requirements

Key techniques:

* Repetition: Repeat user query 4+ times throughout prompt
* Negative Instructions: Explicitly ban copying reference plots
* Structural Constraints: Enforce screenplay format requirements
* Actor Integration: Inject enriched actor profiles into context
* Limited Reference: Show only 500 chars of reference to prevent copying

Example prompt:

prompt = f"""

You are a Hollywood screenwriter.

==========================================

YOUR ASSIGNMENT - READ THIS CAREFULLY:

==========================================

STORY YOU MUST WRITE ABOUT: {query}

GENRE: {genre or 'Any'}

{actor\_block}

==========================================

CRITICAL: Your screenplay MUST be about:

"{query}"

IF YOUR SCREENPLAY IS NOT ABOUT "{query}", YOU HAVE FAILED.

==========================================

Write a {genre or ''} screenplay that tells THIS story: {query}

The title, characters, plot, and dialogue must ALL relate to: {query}

DO NOT write about:

- ANY story other than: {query}

YOUR SCREENPLAY FORMAT:

- Title: (create a title for a story about: {query})

- Scene headings: INT./EXT. LOCATION - TIME

- Character names: {('Use CAST names: ' + ', '.join([a['name'] for a in actors])) if actors else 'Create original names'}

- Action: Describe events in the story about: {query}

- Dialogue: Characters discussing events from: {query}

- Length: 5-10 pages

REMEMBER: Every single line must relate to the story: {query}

START YOUR {genre or ''} SCREENPLAY ABOUT "{query}" NOW:

FADE IN:

"""

Component 3: Llama3 Generation

Model Choice: “llama3:8b” via Ollama

Motivation:

* + It can output longer and more creative scripts
  + Strong instruction-following capabilities
  + 8B parmeters: balanced speed/quality
  + Supports long context

Generation Parameters:

```python

{

"temperature": 0.9, # High creativity, diverse outputs

"top\_p": 0.9, # Nucleus sampling for quality

"top\_k": 40, # Token selection diversity

"repeat\_penalty": 1.1, # Reduce repetitive text

"num\_predict": 4000, # Allow full-length scripts

"seed": random.randint() # Prevent caching

}

```

Component 4: MCP Server

The system provides six primary endpoints that manage the screenplay generation workflow:

* **Actor Management** (GET/POST/DELETE /actors): Maintains a cast database with actor profiles including age, gender, and specialty attributes
* **Script Generation** (POST /generate-script): Orchestrates the RAG pipeline by retrieving relevant screenplay excerpts based on genre and user query, then prompting Llama to generate original content in Hollywood script format
* **Health Monitoring** (GET /health): Validates system components including RAG initialization and Ollama availability

A screenshot of a computer

AI-generated content may be incorrect.

Component 5: Image Generation

Motivation: Create visual assests to complement scripts

Stable Diffusion (local): Free, fast and customizable

Generation strategy:

1. Movie Poster: Single high-quality image with cinematic composition
2. Scene Images: 3 key scenes extracted from script, visualized

### 3.2.1 Why Local Stable Diffusion Instead of Cloud APIs

We run Stable Diffusion for few reasons:

1. Zero recurring cost and no token limits when generating dozens of scene images during testing.
2. Full privacy – no user prompts or generated scripts ever leave the machine.

Part 4: Implementation Details

4.1 File structure

MovieScriptGenerator/

├── GenAIMovie/

│ ├── scripts/ # Downloaded movie scripts

│ └── movie\_scraper.py # Script scraping tool

├── MovieScriptGeneratorStreamlit/

│ ├── streamlit\_app\_llama.py # Main Llama3 app

│ ├── streamlit\_app\_mistral.py # Alternative Mistral app

│ └── mcp\_server.py # FastAPI server

├── chroma\_db/ # Vector database storage

└── stable-diffusion-webui/ # Local SD installation (ignored)

4.2 Key Functions

RAG retrieval

python

def retrieve\_and\_build\_prompt(collection, query, genre, actors):

# Query vector DB for relevant scenes

results = collection.query(

query\_texts=[query],

n\_results=6,

where={"genre": normalized\_genre}

)

# Build prompt with references + constraints

prompt = f"""

YOUR ASSIGNMENT: {query}

GENRE: {genre}

CAST: {actor\_profiles}

[Reference scenes for format - DO NOT COPY PLOTS]

{reference\_text[:500]}...

START WRITING: {query}

"""

return prompt, sources

```

Script Generation

```python

def call\_llama\_local(prompt):

response = requests.post(

"http://localhost:11434/api/generate",

json={

"model": "llama3:8b",

"prompt": prompt,

"options": {

"temperature": 0.9,

"num\_predict": 4000,

"seed": random.randint(1, 1000000)

}

}

)

return response.json()["response"]

```

Actor Enrichment:

```python

def enrich\_actor\_with\_openai(actor\_name):

prompt = f"Provide details about actor {actor\_name}: known\_for, personality, typical\_roles, nationality"

response = openai\_client.chat.completions.create(

model="gpt-4o-mini",

messages=[{"role": "user", "content": prompt}],

response\_format={"type": "json\_object"}

)

return json.loads(response.choices[0].message.content)

### 4.2.2 Local Inference Choices

Ollama was chosen because:

| **Component** | **Chosen Solution** | **Main Alternatives Considered** |
| --- | --- | --- |
| Embedding model | all-MiniLM-L6-v2 | bge-small, e5-base, text-embedding-3-large |
| Vector store | ChromaDB (persistent) | FAISS (in-memory), Qdrant, Pinecone, LanceDB |
| LLM | Llama-3-8B-Instruct (local) | GPT-4o, Claude-3.5, Mixtral-8×7B, Mistral-7B |
| Image generation | Stable Diffusion A1111 (local) | DALL·E 3, Midjourney, Flux |
| Retrieval size | 6 chunks, 500 chars each | 3 / 10 / 20 chunks tested |

Part 5: Evaluation

| **Aspect** | **Mistral-7B-Instruct** | **Llama-3-8B-Instruct** |
| --- | --- | --- |
| Output length | Often stopped after 1–2 pages (~800–1200 tokens) | Consistently 5–10 pages (3200–4000 tokens) |
| Token limit behavior | Frequently hit the internal context limit and truncated | Almost never truncated (thanks to better long-context handling) |
| Creativity & originality | Heavy reliance on retrieved chunks → high plot copying | Much better at inventing new situations and twists |
| Instruction following (repeating the user query, negative rules) | Moderate (~70 % success) | Excellent (~95 % success) |

#### Informal User Testing (Friends & Family)

We invited 5 non-technical people (friends, family, classmates) to try the Streamlit app over two weekends. Prompts they tried:

* “A horror movie where a family moves into an old lighthouse”
* “A romantic comedy with Ryan Gosling and Emma Stone vibes on a film set”
* “An action movie starring Tom Hardy as a retired boxer forced back into the ring”

| **Metric** | **Result (final system)** |
| --- | --- |
| Proper screenplay formatting | 80 % of the time |
| Plot copying from references | < 20 % |
| Recognizable actor persona (when provided) | 89 % correct guesses |
| Average generation time (MacBook Pro M2) | 58–120 seconds |
| Scripts that reached 2+ pages | 90 % |

5.1 Test Example

Actor Booking System User Input:

***Actor Selection****: Keanu Reeves, Anne Hathaway, Pedro Pascal*

Actor Booking System Output:

1. Fetch information from Open AI

Keanu Reeves

***“Known for:****The Matrix, John Wick, Speed, Bill & Ted's Excellent Adventure, Constantine*

***Style:****Keanu Reeves is often characterized by his calm demeanor, humility, and a sense of sincerity in his performances. He tends to portray characters that are introspective and often heroic, with a blend of vulnerability and strength.*

***Nationality:****Canadian*

***Highlights:****MTV Movie Award for Best On-Screen Duo (with Sandra Bullock) for 'Speed', Critics' Choice Movie Award for Best Action Movie for 'John Wick', Star on the Hollywood Walk of Fame (2019)”*

Anne Hathaway

**“*Known for:****The Princess Diaries, Les Misérables, The Devil Wears Prada, Interstellar, Rachel Getting Married*

***Style:****Anne Hathaway is known for her versatility and ability to portray a wide range of emotions. She often brings a blend of charm and depth to her roles, making her characters relatable and engaging.*

***Nationality:****American*

***Highlights:****Academy Award for Best Supporting Actress for Les Misérables (2013), Golden Globe Award for Best Supporting Actress for Les Misérables (2013), BAFTA Award for Best Supporting Actress for Les Misérables (2013)”*

Pedro Pascal

***“Known for:****The Mandalorian, Game of Thrones, Narcos, Wonder Woman 1984, The Last of Us*

***Style:****Pedro Pascal is known for his charismatic and versatile performances. He often brings a blend of intensity and warmth to his roles, making him relatable and engaging on screen.*

***Nationality:****Chilean-American*

***Highlights:****Critics' Choice Television Award for Best Supporting Actor in a Drama Series (2014) for 'Game of Thrones', Screen Actors Guild Award for Outstanding Performance by an Ensemble in a Drama Series (2016) for 'Game of Thrones', Hollywood Critics Association Award for Best Actor in a Streaming Series, Drama (2023) for 'The Last of Us'”*

1. Fetch info from TMDB

Keanu Reeves

***“Known for:****Dangerous Liaisons, My Own Private Idaho, Point Break, Constantine, Street Kings”*

User Input for Script Generation:

***Movie Title:*** *Midnight Atlas*

***Story Idea / Prompt:*** *A brilliant but reclusive cartographer discovers a hidden pattern in global satellite data. She maps that shift overnight. She teams up with a world-weary Interpol investigator to uncover a conspiracy linking disappearing coordinates to a clandestine memory-erasing technology that could rewrite history itself.*

***Genre:*** *Sci-Fi*

Generated Script:

\*\*MAP OF MEMORY\*\*

INT. CARTOGRAPHER'S STUDY - NIGHT

Anne Hathaway, as CAROLINE WARD, a brilliant but reclusive cartographer, sits hunched over her computer, surrounded by maps and charts. She's been staring at the same screen for hours, her eyes bloodshot from fatigue.

CAROLINE:

(to herself)

What am I missing?

Suddenly, her computer beeps. Caroline's eyes light up as she zooms in on a peculiar pattern in the global satellite data.

CUT TO:

INT. INTERPOL HEADQUARTERS - DAY

Pedro Pascal, as AGENT JASON RYDER, a world-weary Interpol investigator, sits at his desk, sipping coffee and staring at a report about missing coordinates.

JASON:

(to himself)

What's going on? These disappearances are unlike anything I've seen before...

CUT TO:

INT. CARTOGRAPHER'S STUDY - NIGHT (LATER)

Caroline's phone rings. It's Jason.

CAROLINE:

(answering)

Agent Ryder?

JASON:

(over the phone)

Caroline Ward, we need to talk.

CAROLINE:

(curious)

What about?

JASON:

(hesitant)

I think you're onto something big, Caroline. The missing coordinates... it's not just a random anomaly. I think someone's erasing memories on a massive scale.

CAROLINE:

(intrigued)

Go on.

JASON:

(determined)

Let's meet in person and discuss the implications. Time is of the essence.

CUT TO:

EXT. CITY STREET - NIGHT

Caroline and Jason meet, their footsteps echoing off the city streets as they walk to a nearby coffee shop.

CAROLINE:

(worried)

What does this mean? A memory-erasing technology that can rewrite history?

JASON:

(grim)

It's catastrophic. We have to stop it before it's too late.

FADE TO BLACK.

Generated Movie Poster

A person in space suit

AI-generated content may be incorrect.

Key Scenes:

Scene 1: INT. CARTOGRAPHER'S STUDY - NIGHT

A room with a nice floor

AI-generated content may be incorrect.

Scene 2: INT. INTERPOL HEADQUARTERS – DAY

A group of people in an office

AI-generated content may be incorrect.

Scene 3: INT. CARTOGRAPHER'S STUDY - NIGHT (LATER)

A screenshot of a movie script generator

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.A group of men standing on a street

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.