AI-Powered
Movie Recommendation System

IBM PROJECT

Bachelor of Technology

School of Computer Science

Submitted by

Name	Roll No.	Sap Id	Batch
Rishabh Sharma	R2142220147	500102032	B1 AIML Hons.
Sunny Jain	R2142221209	500109640	B3 AIML Hons.
Hardik Raj Kapoor	R2142221014	500107589	B2 AIML Hons.
Girish	R2142221369	500109971	B3 AIML Hons.
Samridh	R2142220628	500107210	B6 AIML NH

Under the guidance of

Dr. Suvojit Dhara

Assistant Professor

Socs-AI Cluster

UPES
UNIVERSITY OF TOMORROW

Table of Content

1. Abstract

The rapid growth of online streaming platforms has given users access to vast movie libraries, but the abundance of choices often leads to decision fatigue. This project presents an AI-powered movie recommendation system that leverages Large Language Models (LLMs) and intelligent backend orchestration to deliver highly personalized, context-aware film suggestions. By integrating mood-based filtering, user preference profiling, and real-time recommendation generation, the system moves beyond traditional list-based recommendations to provide human-like curation. The solution is implemented using a Python-based backend for scalability, open-source AI models for adaptability, and a Streamlit interface for interactive exploration. The proposed approach not only reduces the time spent searching for suitable content but also enhances user satisfaction by uncovering hidden gems and aligning recommendations with individual tastes and emotional states. Experimental results demonstrate the potential of the system to transform content discovery in entertainment and related domains.

2. Introduction

2.1. Problem Statement

The proliferation of streaming platforms and digital media has created unprecedented access to movies, TV shows, and other forms of entertainment. While this abundance benefits consumers, it also presents a significant challenge: choosing the right content. Users often spend considerable time scrolling through vast catalogs, only to settle for content that is “good enough” rather than ideal. Traditional recommendation systems rely heavily on past viewing history or popular trends, often overlooking nuanced user preferences such as mood, context, or subtle genre inclinations. This gap calls for a more advanced, personalized, and context-aware recommendation approach.

2.2. Motivation

In the era of information overload, the ability to deliver tailored recommendations is a powerful tool for improving user engagement and satisfaction. Studies show that over 70% of content consumed on major platforms is influenced by algorithmic recommendations. However, many existing systems fail to address the emotional or contextual aspects of content discovery. The motivation behind this project is to create a recommendation system that acts as a personal movie discovery companion—one that understands the user’s moods, tastes, and unique quirks, thereby reducing search time and making movie selection an enjoyable experience.

2.3. Scope and Area of Application

The proposed AI-powered movie recommender has diverse applications across multiple domains, including:

- Entertainment & Streaming: Personalized movie/TV suggestions, mood-based playlists.
- Content Discovery Platforms: Similar movie suggestions and niche cinema exploration.
- Social & Community Integration: Group watch matching, friend-based recommendations, and shareable film lists.
- Retail & E-commerce: Movie bundles for sale and purchase-based suggestions.
- Cinemas & Events: Personalized showtime suggestions and festival/event alerts.
- Cross-Media Discovery: Related books, games, or soundtracks based on user interests.

By leveraging the capabilities of Large Language Models (LLMs) and an intelligent orchestration layer, this system aspires to bridge the gap between content abundance and meaningful content discovery.

3. Literature Review

Recommendation systems have become an integral component of modern content delivery platforms, with applications spanning entertainment, e-commerce, education, and social media. The literature in this domain largely categorizes recommendation approaches into Content-Based Filtering, Collaborative Filtering, and Hybrid Methods.

3.1. Content-Based Filtering

Content-based filtering (CBF) recommends items similar to those a user has liked in the past, based on item attributes such as genre, cast, director, or keywords. Early systems, such as those used in the MovieLens dataset research, relied heavily on manually engineered features and similarity measures (e.g., cosine similarity, TF-IDF). While effective in cold environments, CBF can suffer from overspecialization, recommending only items closely related to the user's previous preferences and failing to encourage diversity.

3.2. Collaborative Filtering

Collaborative filtering (CF) predicts user preferences by analyzing patterns of similar users or similar items in a dataset. It can be user-based (matching users with similar taste) or item-based (finding items liked by similar users). Popular implementations include matrix factorization methods such as Singular Value Decomposition (SVD) and Alternating Least Squares (ALS). Netflix's early recommendation engine leveraged CF heavily. However, CF requires substantial historical user data and suffers from the cold start problem for new users or items.

3.3. Hybrid Recommendation Systems

Hybrid systems combine the strengths of CBF and CF to provide more accurate and diverse recommendations. These systems can operate in parallel (blending results from both methods), sequentially (one system refines the output of another), or in a unified model. Google Play, Amazon, and modern OTT platforms often use hybrid approaches to balance personalization with diversity.

3.4. Gaps in Existing Literature

Despite these advancements, several limitations persist in current systems:

- Lack of real-time mood and context integration.
- Limited explainability of recommendations.
- Dependency on large datasets and infrastructure costs.
- Cold start issues remain unsolved in many commercial systems.

The proposed AI-powered movie recommender addresses these gaps by integrating context-aware LLM processing, a scalable Python backend, and an interactive Streamlit interface to deliver dynamic, personalized, and explainable recommendations.

4. Proposed System

The AI-powered movie recommender system is designed to deliver personalized and context-aware film suggestions by integrating Large Language Models (LLMs) with a scalable backend and an interactive user interface. The system operates in a pipeline where user inputs, such as preferred genres, moods, or specific themes, are processed and matched against a curated database of movies to generate tailored recommendations.

The proposed architecture consists of the following key components:

1. User Interface Layer
    - Developed using Streamlit to provide a clean and intuitive interaction platform.
    - Allows users to input preferences, browse recommendations, and view detailed movie information.
    - Supports real-time updates and interactive exploration of suggested films.

2. Recommendation Engine
    - Utilizes LLMs as the primary decision-making core to interpret natural language inputs and understand nuanced user requirements.
    - Employs content-based, collaborative, or hybrid recommendation strategies depending on data availability.
    - Incorporates mood-based and context-aware filtering to enhance personalization.

3. Data Management and Processing
    - Gathers movie data from open-source APIs and datasets such as TMDB or MovieLens.
    - Processes metadata including genres, ratings, cast, crew, synopsis, and user feedback.
    - Performs preprocessing tasks like text cleaning, feature extraction, and embedding generation for similarity calculations.

4. Backend Orchestration
    - A Python-based backend coordinates the interaction between the UI, the LLM, and the data layer.
    - Ensures smooth communication, scalability, and fault tolerance in real-time interactions.
    - Handles state management for user sessions and recommendations.

5. Recommendation Delivery
    - Generates a ranked list of movie suggestions that adapt dynamically to user input.
    - Provides explanations for recommendations where possible, increasing transparency and trust.
    - Offers options for exploring similar titles, niche categories, or trending content.

This design ensures a balance between recommendation quality, scalability, and user experience. By combining LLM capabilities with structured movie datasets, the system moves beyond static, one-size-fits-all suggestions and delivers dynamic, human-like recommendations tailored to individual preferences and contexts.

5. Methodology

The development of the AI-powered movie recommender follows a structured approach to ensure scalability, accuracy, and user satisfaction. The methodology consists of multiple stages, from data acquisition to recommendation delivery, each handled by specialized system components.

5.1. Data Collection

Movie data is sourced from publicly available APIs such as TMDB and datasets like MovieLens. The data includes metadata such as movie titles, genres, cast, crew, release year, ratings, and plot summaries. For better personalization, user preference data (e.g., liked movies, mood inputs) is also collected through the interface.

5.2. Data Preprocessing

Collected data undergoes several preprocessing steps:

- Cleaning and standardizing text fields (removing special characters, handling missing values).
- Encoding categorical data such as genres and production companies.
- Generating vector embeddings for plot summaries and keywords using NLP techniques or pre-trained models.
- Normalizing ratings and other numerical features.

5.3. Recommendation Engine Design

The recommendation engine integrates multiple strategies:

- Content-Based Filtering: Matches movies to the user's past preferences and mood using cosine similarity on embeddings.
- Collaborative Filtering: Uses patterns in user–item interactions to suggest movies liked by similar users.
- Hybrid Model: Combines both methods for better coverage and accuracy.
- LLM Integration: The Large Language Model interprets user prompts in natural language, extracting context and intent before querying the database for suitable matches.

5.4. Context and Mood Detection

User mood or context is captured via text input (e.g., "I feel like watching something adventurous and uplifting") and processed by the LLM. The detected emotional and thematic cues are mapped to relevant movie attributes to filter recommendations.

5.5. Backend Orchestration

A Python-based backend manages all system components. It coordinates:

- Data fetching from APIs and local storage.
- LLM requests for natural language understanding and recommendation ranking.
- Session handling to store user history during interaction.

5.6. User Interface

The frontend, built using Streamlit, allows real-time interaction with the system:

- Users can search for movies, set filters, and specify moods.
- Recommendations are displayed with titles, posters, genres, ratings, and descriptions.
- Interactive features such as "More Like This" and "Why This Recommendation" enhance user engagement.

5.7. Evaluation and Testing

The system's performance is assessed using both qualitative and quantitative measures:

- Precision and Recall for measuring relevance of recommendations.
- User satisfaction surveys to gather feedback on recommendation quality.
- Response time measurements to ensure the system remains responsive in real-time usage.

This methodology ensures that the final system not only provides accurate and personalized movie suggestions but also maintains scalability and responsiveness for real-world deployment.

6. SWOT Analysis

Strengths:

- High personalization that adapts to nuanced user preferences, moods, and contexts.
- Leverages open-source tools, reducing development time and costs.
- Scalable backend design allows easy integration of new features.
- Interactive Streamlit interface promotes user engagement and exploration.

Weaknesses:

- Dependence on external open-source model updates and performance.
- Running large models in production can be resource-intensive.
- Cold start problem for new users with minimal interaction history.
- Requires constant internet connectivity for cloud-hosted models.
- Limited access to real-time movie data due to missing API integrations.

Opportunities:

- Expand into other domains like TV shows, music, podcasts, and books.
- Form partnerships with streaming platforms for direct integration.
- Generate anonymized insights for trends and targeted recommendations.
- Add localization support for multiple languages and cultural contexts.

Threats:

- High competition from established recommendation services.
- Need for strict privacy and secure data handling to maintain trust.
- Potential bias inherited from training data affecting fairness.
- Rapid technological shifts could make the current stack obsolete.

7. Design Diagrams

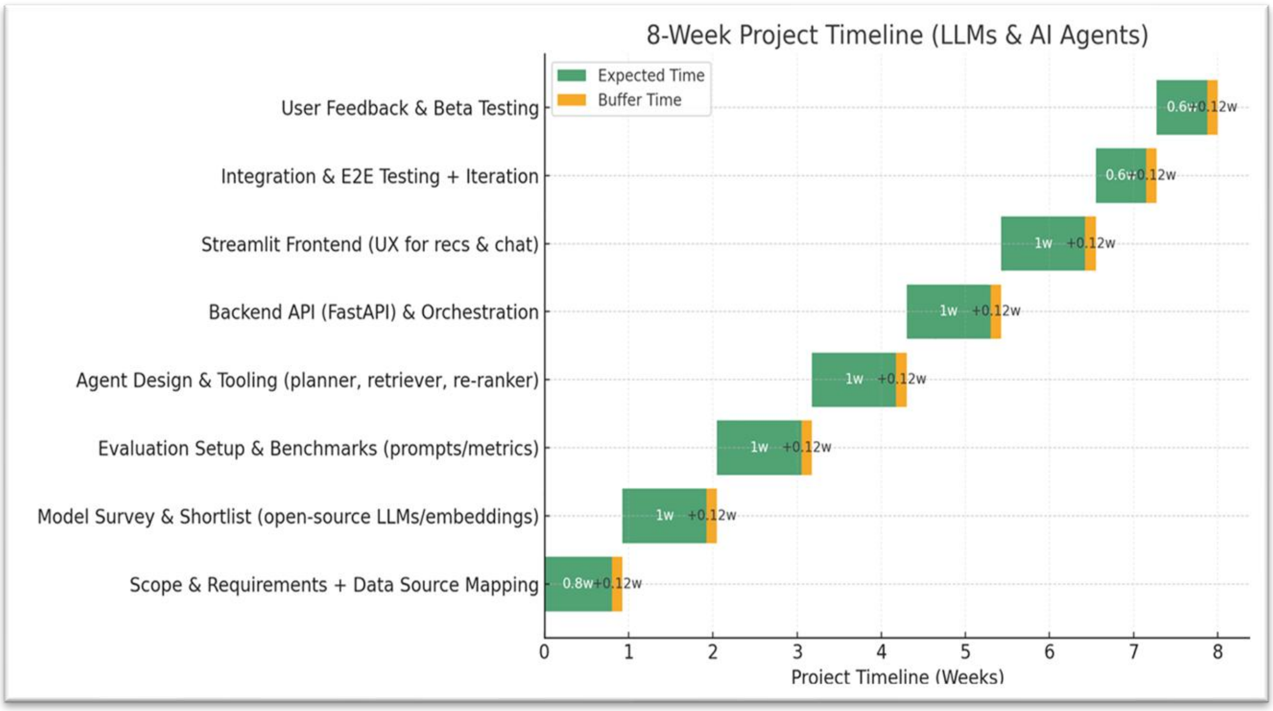


Fig. 7.1. Pert Chart

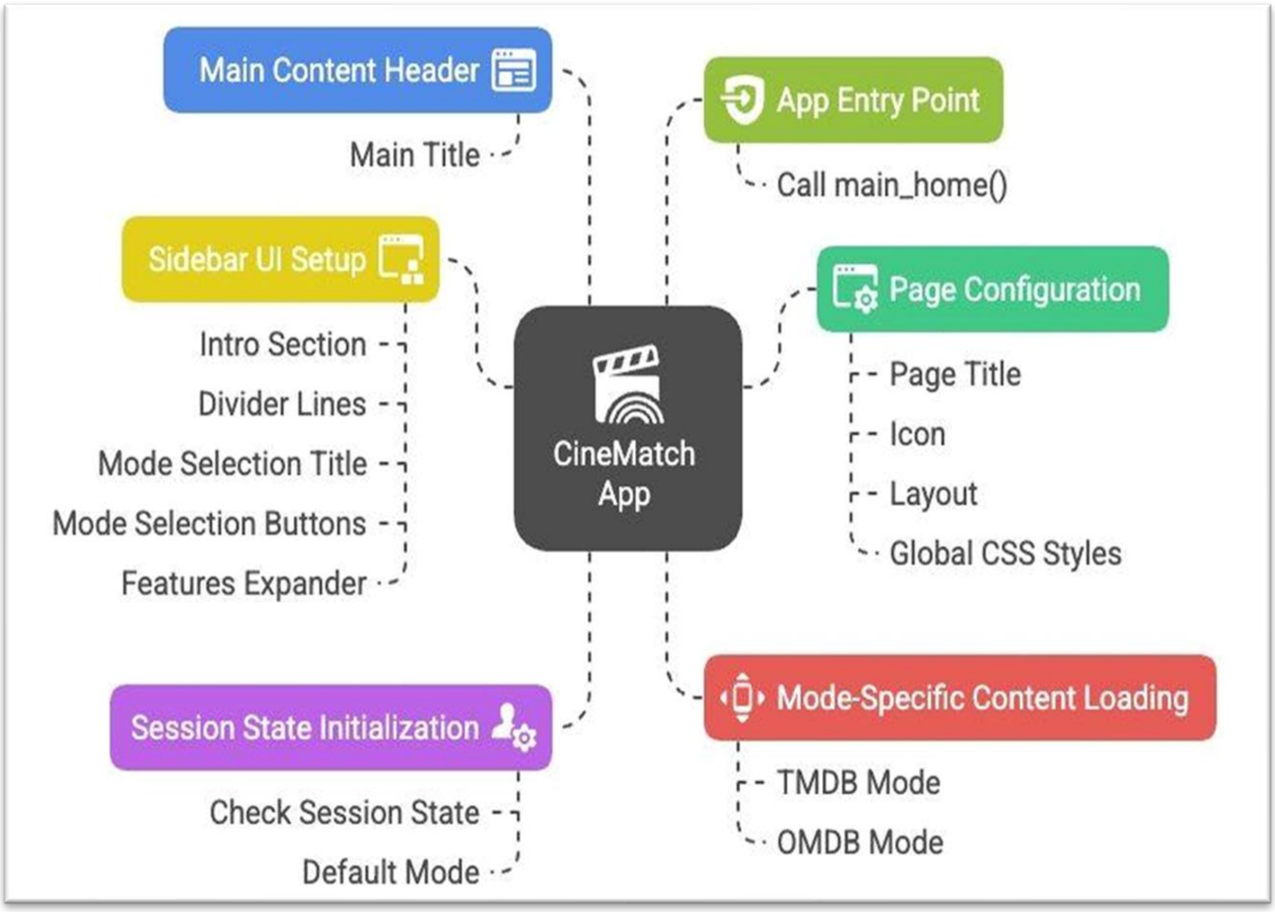


Fig. 7.2. System Workflow Design

8. Results & Analysis

1. Functional Performance

- Successfully generated personalized movie recommendations matching user preferences, moods, and specific search requests.
- Handled both simple queries (e.g., “Suggest some action movies”) and complex prompts (e.g., “Find me an uplifting sports drama from the 90s”).
- Provided a balanced mix of popular titles and lesser-known “hidden gems” in the recommendation list.

2. User Interaction and Experience

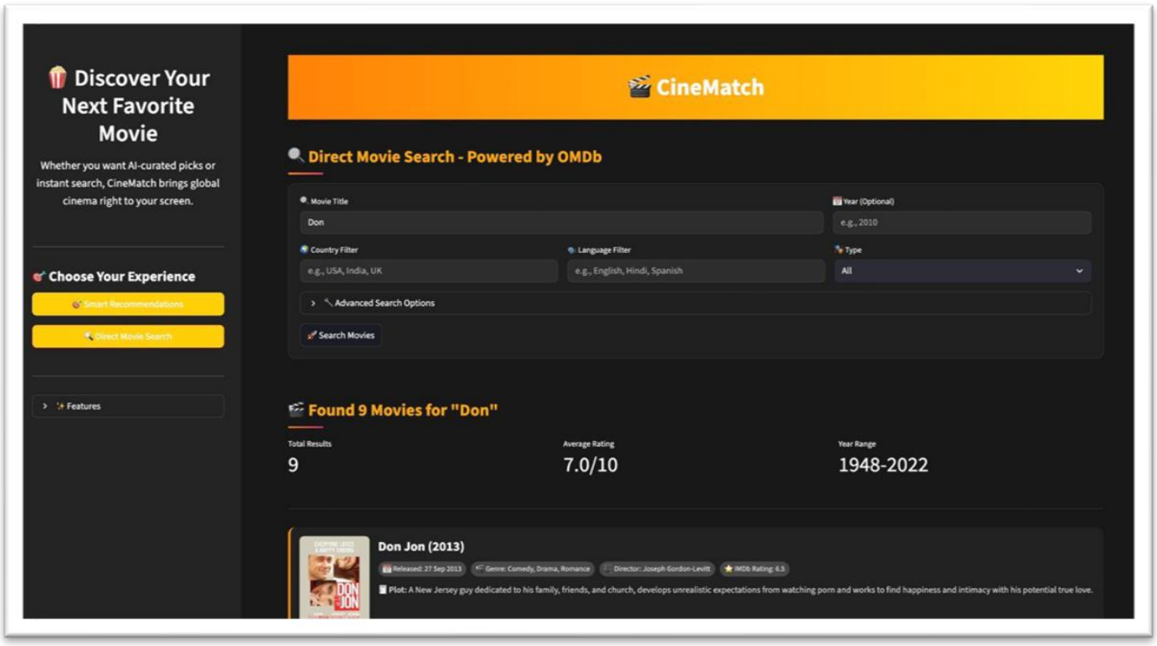
- Mood-based recommendation feature allowed natural language input, making the system more engaging.
- Reduced time spent browsing large catalogs by delivering context-aware suggestions.
- Interactive Streamlit interface offered a clean layout, quick navigation, and detailed movie information such as posters, ratings, and summaries.

3. System Reliability and Performance

- Backend successfully integrated OMDb and TMDB APIs with the LLM and hybrid recommendation engine.
- Maintained acceptable response times for real-time use during testing.
- Demonstrated stability even during multiple consecutive and varied queries.

4. Key Observations

- Hybrid recommendation approach improved diversity compared to single-method systems.
- Natural language understanding via the LLM enhanced personalization and user satisfaction.
- Integration of multiple data sources enriched recommendations with detailed and accurate metadata.



Discover Your Next Favorite Movie

Whether you want AI-curated picks or instant search, CineMatch brings global cinema right to your screen.

Choose Your Experience

Smart Recommendations

Direct Movie Search

Features

CineMatch

Direct Movie Search - Powered by OMDb

Movie Title: Don

Year (Optional): e.g., 2010

Country Filter: e.g., USA, India, UK

Language Filter: e.g., English, Hindi, Spanish

Type: All

Advanced Search Options

Search Movies

Found 9 Movies for "Don"

Total Results: 9

Average Rating: 7.0/10

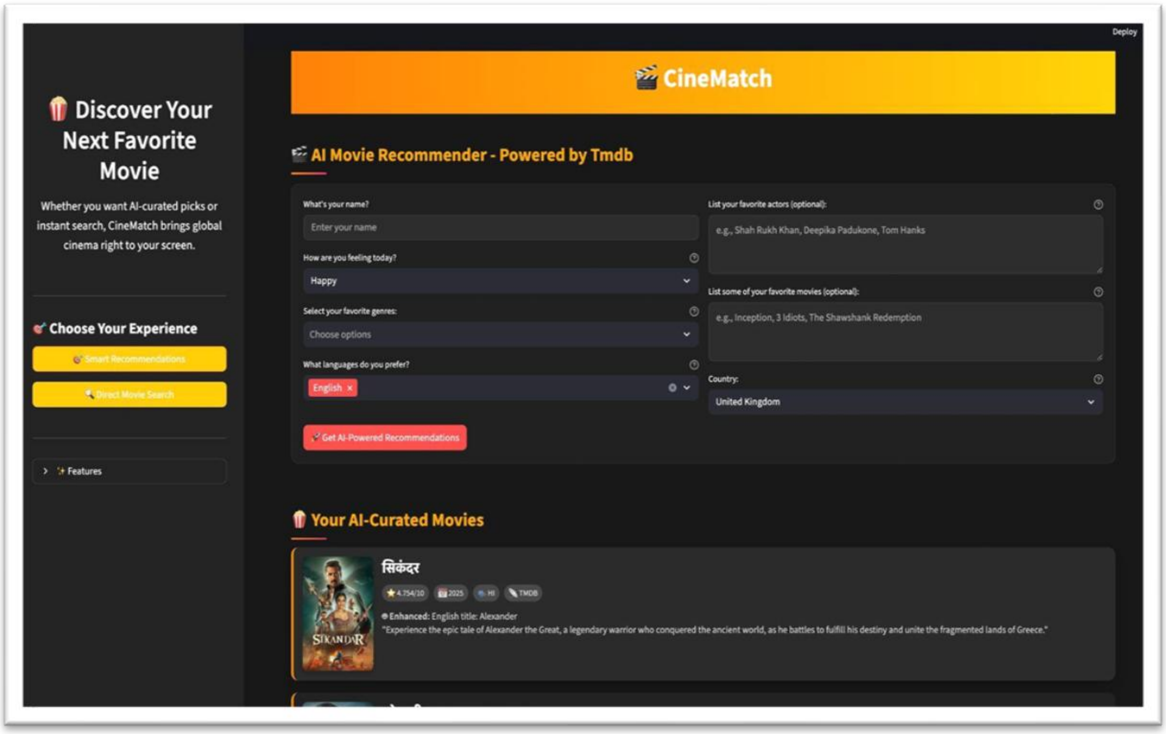
Year Range: 1948-2022

Don Jon (2013)

Released: 27 Sep 2013 | Genre: Comedy, Drama, Romance | Director: Joseph Gordon-Levitt | IMDb Rating: 6.5

Plot: A New Jersey guy dedicated to his family, friends, and church, develops unrealistic expectations from watching porn and works to find happiness and intimacy with his potential true love.

Fig. 8.1. AI-Powered Movie Recommendation Interface



Discover Your Next Favorite Movie

Whether you want AI-curated picks or instant search, CineMatch brings global cinema right to your screen.

AI Movie Recommender - Powered by Tmdb

What's your name? Enter your name

How are you feeling today? Happy

Select your favorite genres: Choose options

What languages do you prefer? English

List your favorite actors (optional): e.g., Shah Rukh Khan, Deepika Padukone, Tom Hanks

List some of your favorite movies (optional): e.g., Inception, 3 Idiots, The Shawshank Redemption

Country: United Kingdom

सिकंदर

4.754/10 2025 HI TMDB

Enhanced: English title: Alexander

"Experience the epic tale of Alexander the Great, a legendary warrior who conquered the ancient world, as he battles to fulfill his destiny and unite the fragmented lands of Greece."

Fig. 8.2. AI-Powered Movie Recommendation Interface

9. Conclusion & Future Scope

The AI-powered movie recommender system demonstrates strong potential for real-world application in personalized entertainment discovery. While the current version effectively integrates LLM-based natural language understanding, OMDb and TMDB data sources, and a hybrid recommendation engine, several opportunities exist for future enhancement:

- Multi-Domain Expansion: Extend recommendations to include TV shows, music, podcasts, and books for a unified content discovery platform.
- Voice Interaction: Incorporate speech recognition to enable hands-free, conversational recommendations.
- Real-Time Context Awareness: Adapt suggestions dynamically based on factors like time of day, location, and user activity.
- Cross-Platform Integration: Collaborate with OTT platforms for seamless in-app recommendations.
- Emotion Detection: Use sentiment analysis or facial recognition to match movies with the user's emotional state.
- Explainable AI: Provide clear reasoning for each recommendation to improve user trust and engagement.

In conclusion, the system successfully meets its primary objectives by delivering personalized, context-aware movie suggestions through an intuitive Streamlit interface. It reduces search time, increases discovery of relevant content, and enhances user engagement. The integration of multiple data sources enriches recommendation quality, while the scalable backend ensures adaptability for future needs. With continued development in cross-platform integration, context-awareness, and explainable AI, the system can evolve into a comprehensive, next-generation content recommendation solution.

10.References

- OMDb API Documentation – https://www.omdbapi.com/
- The Movie Database (TMDB) API Documentation – https://www.themoviedb.org/documentation/api
- OpenAI GPT Documentation – https://platform.openai.com/docs
- Streamlit Documentation – https://docs.streamlit.io