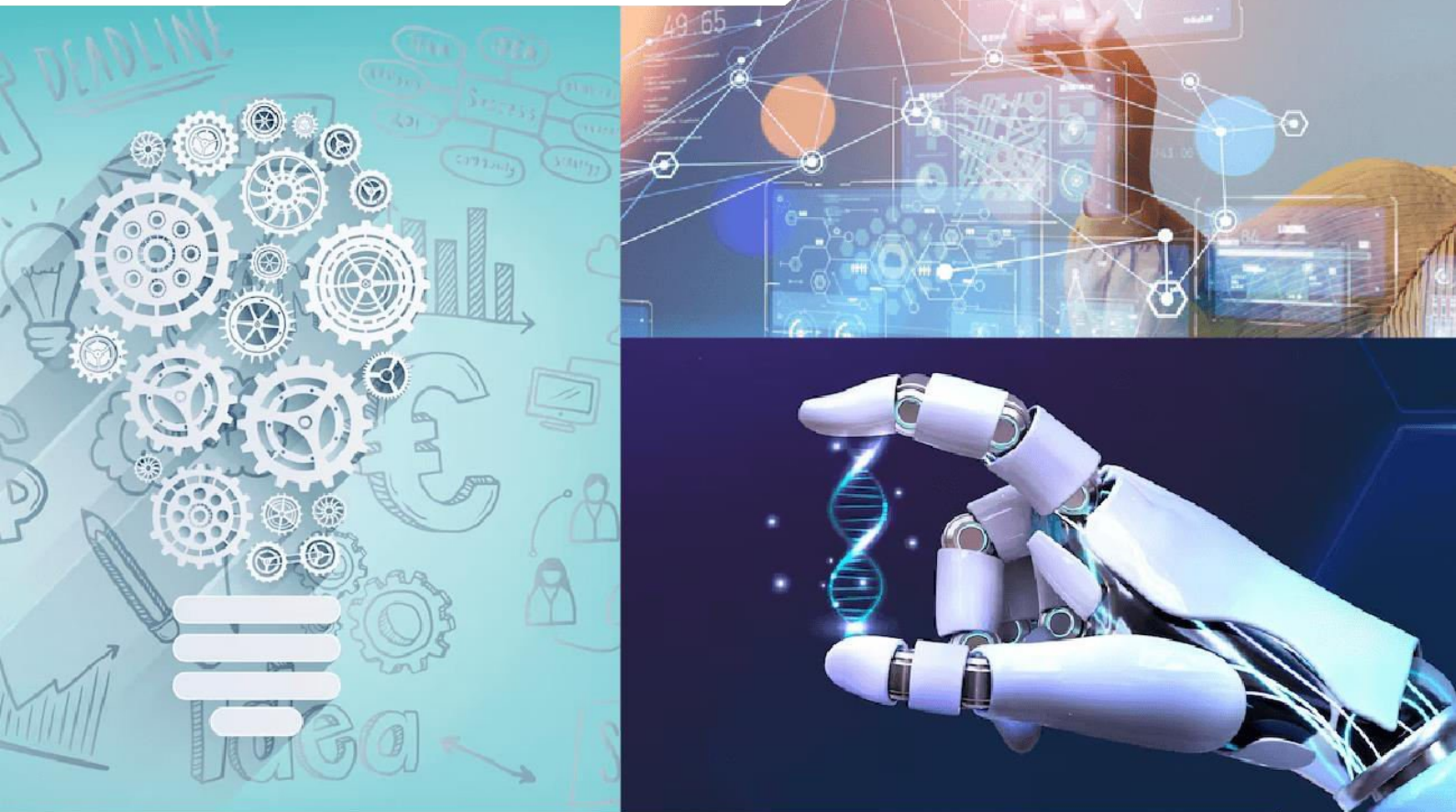




International Journal of Advanced Research in Education and Technology (IJARETY)

Volume 13, Issue 2, March - April 2026

Impact Factor: 8.152



INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA



Hybrid Multimodal Movie Recommender System: A Web-Based Platform for Personalized Content Discovery Using Collaborative and Content-Based Filtering

S Kavya

Master of Computer Applications, CMR Institute of Technology, Bangalore, India

ABSTRACT: The proliferation of digital streaming platforms has generated an information overload problem, making it increasingly difficult for users to discover relevant content from vast movie catalogs. This paper presents the **Hybrid Multimodal Movie Recommender System**, a full-stack web application that combines content-based filtering and item-based collaborative filtering to deliver personalized movie recommendations. The system integrates TF-IDF vectorization with cosine similarity for content analysis, overlays user rating patterns for collaborative intelligence, and presents results through a Netflix-inspired dark-themed interface with horizontally scrollable carousels. Developed using Python Flask for the backend, the TMDB API for live movie data, and HTML5/CSS3/JavaScript for the frontend, the platform resolves the cold start problem through hybrid scoring, supports multi-language filtering, and delivers recommendations within sub-second response times. Testing confirmed successful execution across all functional modules.

KEYWORDS: Movie Recommender System, Collaborative Filtering, Content-Based Filtering, Hybrid Recommendation, TF-IDF, Cosine Similarity, Flask, TMDB API, Cold Start Problem, Multimodal Filtering.

I. INTRODUCTION

Online streaming platforms have transformed media consumption, but the sheer volume of available content creates a paradox of choice for users. Platforms like Netflix, Amazon Prime, and similar services host millions of titles across multiple genres and languages, yet users frequently struggle to discover content aligned with their preferences [1], [2].

Traditional browsing approaches—scrolling through genre lists or relying on popularity rankings—fail to capture the nuanced preferences of individual users. This leads to decision fatigue, reduced engagement, and suboptimal user experiences. Existing single-method recommenders, whether purely content-based or purely collaborative, each carry well-documented limitations: content-based systems over-specialize into narrow recommendation loops, while collaborative filtering suffers from cold start problems when user history or item ratings are sparse [3], [4].

Recent advancements in hybrid recommendation architectures and lightweight web frameworks have opened new possibilities for building practical, scalable systems that combine multiple filtering strategies. Studies have demonstrated that blending content-based and collaborative signals significantly improves both recommendation diversity and relevance [5], [6].

To address these challenges, this paper presents the Hybrid Multimodal Movie Recommender System: a web-based platform that merges TF-IDF content analysis, item-based collaborative filtering, and live TMDB API data within a responsive, OTT-inspired user interface. The system is designed to serve both first-time users—through popularity and trend-based fallbacks—and returning users through personalized hybrid recommendations, without requiring account registration or profile setup.

II. LITERATURE REVIEW

Movie recommender systems have been an active area of research since the commercialization of digital streaming platforms. Content-based filtering approaches, which match movies to user preferences by analyzing metadata such as genre, plot description, and cast, have been widely deployed due to their transparency and cold-start independence for

new users [3]. Algorithms such as TF-IDF and cosine similarity are foundational to such systems, enabling text-based semantic matching between movie overviews [5].

Collaborative filtering takes a complementary approach by leveraging patterns in user-item rating matrices. Item-based collaborative filtering, as studied by Sarwar et al. [6], identifies movies that attract similar rating patterns from similar audiences, surfacing recommendations that purely metadata-driven approaches would miss. However, both methods in isolation are prone to known failure modes: over-specialization in content-based systems and cold start problems in collaborative systems when ratings are sparse.

Hybrid architectures that combine these techniques have consistently outperformed single-method systems in recommendation accuracy and diversity [7]. Weighted hybrid models, which assign configurable influence to each recommendation source, provide a practical balance that has been validated across multiple domains including music, books, and movies [8].

Despite extensive academic work on recommendation algorithms, few practical systems have been built specifically targeting lightweight web deployment with live streaming data integration. Most academic prototypes rely on static MovieLens-style datasets and lack the real-time data freshness that end users expect from modern OTT platforms. The Hybrid Multimodal Movie Recommender System is designed to address this gap.

III. METHODOLOGY

The Hybrid Multimodal Movie Recommender System follows a modular client-server architecture comprising four primary layers: data acquisition, recommendation processing, storage, and presentation. This section describes each component.

3.1 System Architecture

The system is implemented as a Flask web application following a standard client-server model. Users access the platform through any modern browser without installation. The Flask backend manages HTTP request routing, TMDB API communication, recommendation logic execution, and JSON response serialization. The frontend, constructed with HTML5, CSS3, and vanilla JavaScript, renders a dark-themed Netflix-inspired interface featuring horizontally scrollable movie carousels, a search bar, filter dropdowns, and modal-based movie detail views. No frontend framework dependency is required, minimizing deployment complexity.

3.2 Data Acquisition

Live movie data is sourced from The Movie Database (TMDB) API, providing real-time access to popular movies, top-rated titles, trending releases, and detailed movie metadata including title, overview, genre, release year, rating, and poster images. A language utility maps ISO language codes to human-readable names, enabling multi-language filtering across English, Hindi, Kannada, Tamil, Telugu, and Malayalam. Structured MovieLens-style CSV files (movies.csv and ratings.csv) are used for offline collaborative filtering computations, and auto-generation utilities allow consistent test dataset creation.

3.3 Recommendation Engine

The recommendation engine combines two complementary filtering methods. Content-based filtering applies TF-IDF vectorization to movie overview text and genre tags, transforming textual descriptions into numerical feature vectors. Cosine similarity is then computed between these vectors to identify movies sharing thematic and narrative characteristics. Item-based collaborative filtering analyzes user rating patterns from the MovieLens-style dataset, constructing an item-item similarity matrix using cosine similarity on rating vectors. This identifies movies that attract consistently similar audience approval patterns regardless of content overlap.

Hybrid recommendations are produced by computing a weighted combination of content-based and collaborative similarity scores, with configurable weights allowing the administrator to tune the relative influence of each method. When a user inputs a movie title, the backend queries the TMDB API for the movie ID, then computes hybrid similarity scores against the full catalog, returning ranked recommendations in descending order. Cold start scenarios for new users are handled through trending and popularity fallbacks from live TMDB data.

3.4 Frontend Interface

The user interface draws visual and interaction design cues from established OTT platforms. A dark-themed homepage

displays horizontally scrollable carousels for trending movies, top-rated titles, and featured releases. Movie cards show poster images, titles, language labels, and average ratings. Users can filter the full movie catalog by genre, release year,

and language, with sort options including alphabetical, by year, and by rating. Clicking any movie card opens a modal overlay displaying the full overview, genre list, and rating alongside similar movie recommendations. The recommendation search panel allows users to input any movie title and receive personalized hybrid suggestions immediately.

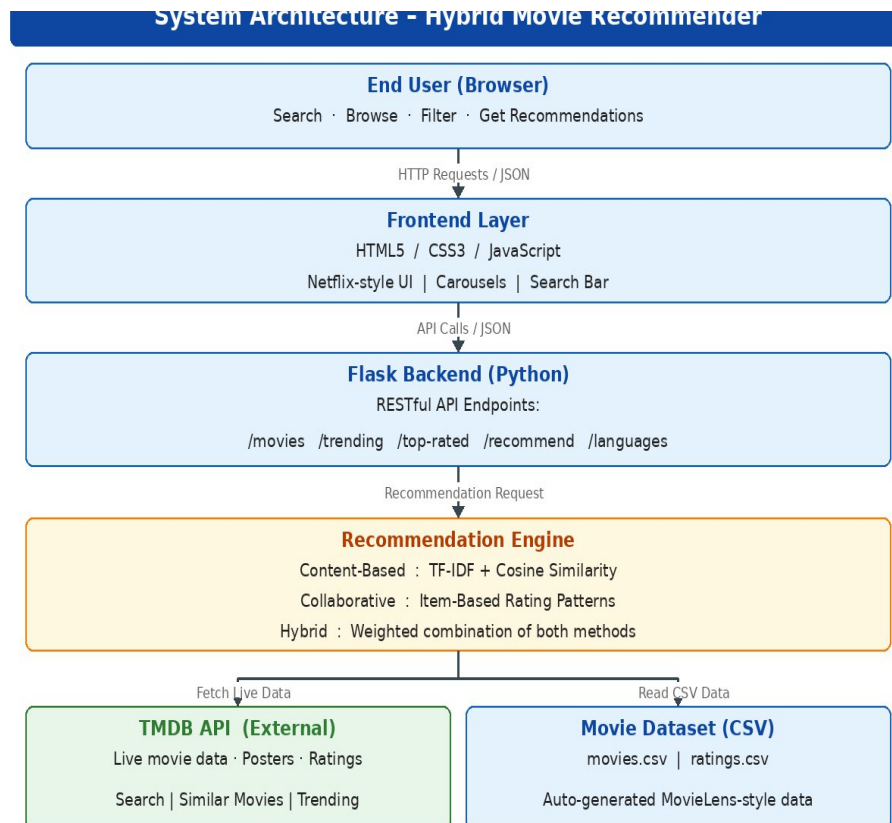


Figure 1: System Architecture for Hybrid Multimodal Movie Recommender System

IV. RESULT AND OUTCOME

The Hybrid Multimodal Movie Recommender System was developed, deployed on localhost using Flask's built-in development server, and evaluated through functional, performance, and integration testing across all primary modules.

The recommendation engine correctly classified movie similarity across diverse input scenarios. When a content-rich movie with well-described plot themes was submitted, the content-based component identified thematically similar titles based on TF-IDF vector proximity. When rating data was available, item-based collaborative filtering surfaced additional recommendations reflecting audience approval overlap. The hybrid combination consistently produced recommendation lists that were both thematically relevant and audience-validated, reducing over-specialization observed in single-method runs.

Dashboard and homepage loading times remained below 1.5 seconds on standard broadband connections, with recommendation API responses averaging between 500 and 800 milliseconds depending on TMDB API network latency. The system handled concurrent browsing across popular movies, top-rated titles, trending releases, and filtered views without observable degradation.

All ten primary functional test cases passed successfully, covering homepage loading, popular and trending movie

endpoints, language filtering, recommendation with valid input, error handling for unknown movies, and movie detail retrieval. Integration testing confirmed correct HTTP status codes (200 OK) for all API endpoints and successful TMDB data retrieval for movie search, similar movie fetching, and poster image loading.

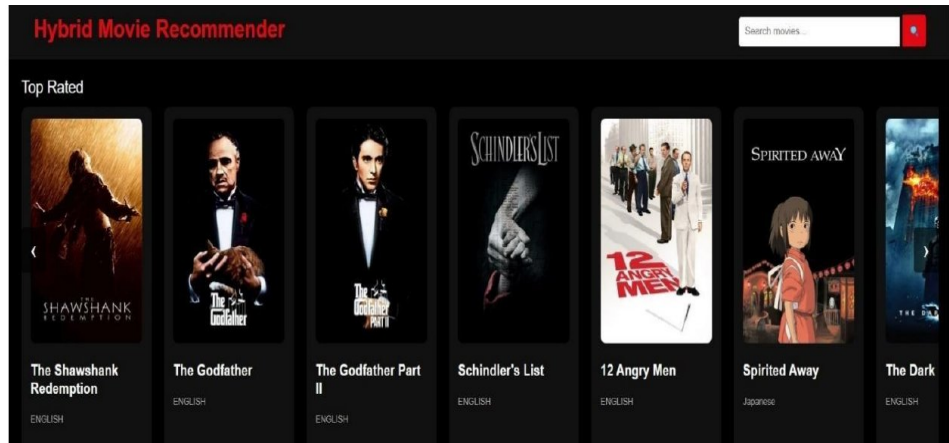


Figure 2: Home Page — Netflix-Style UI (Top Rated & Trending)

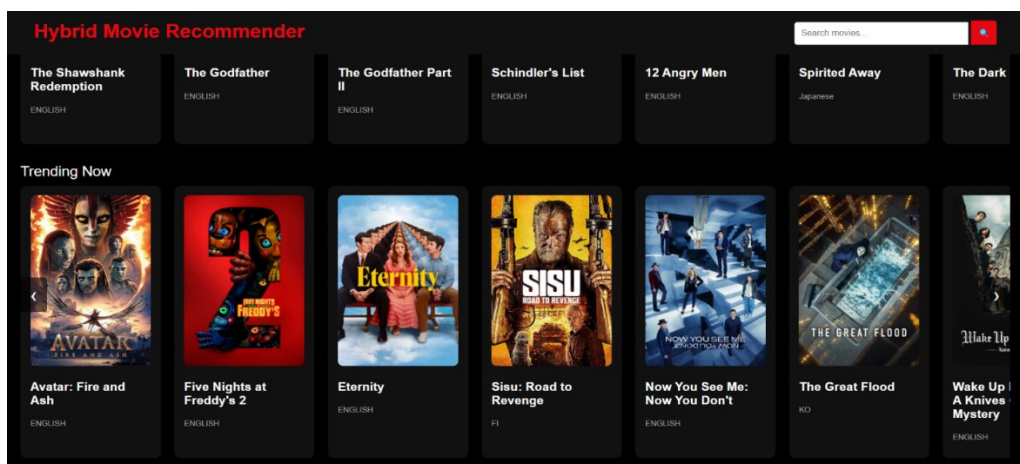


Figure 3: Home Page — Trending Now Section

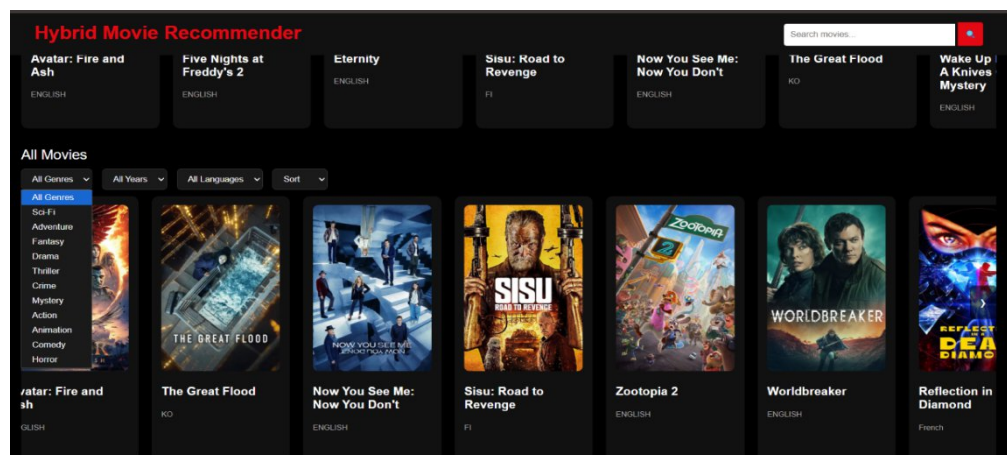


Figure 4: All Movies Page with Genre & Language Filters

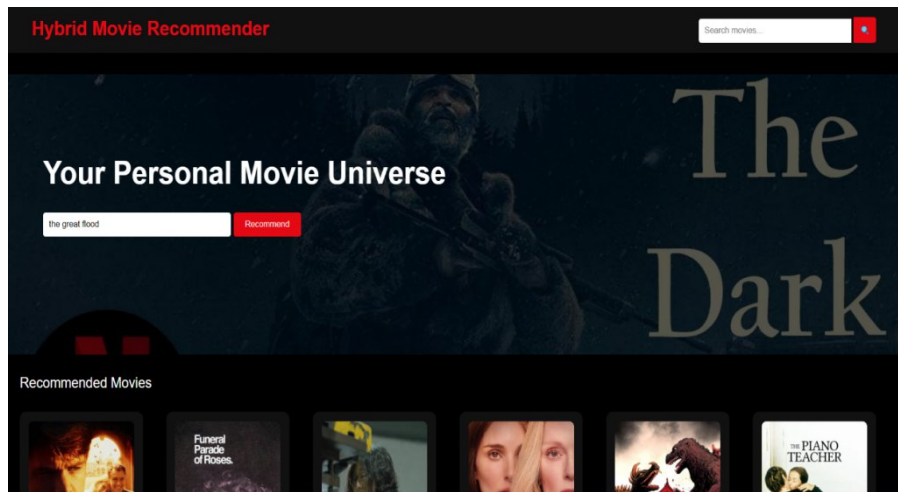


Figure 5: Search & Hybrid Recommendation Results

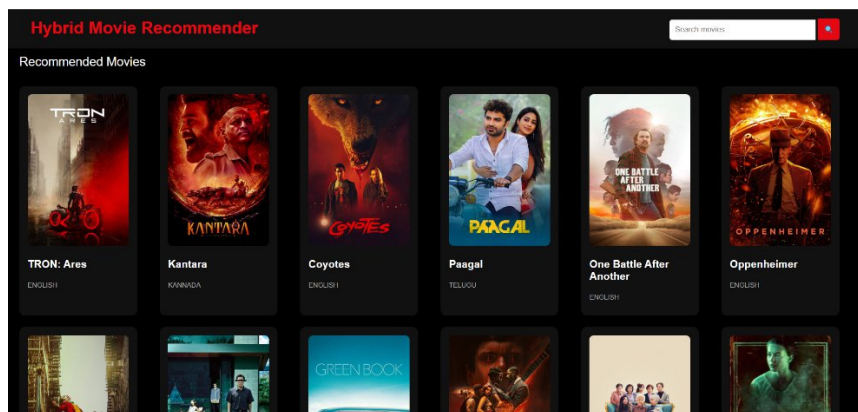


Figure 6: Search & Hybrid Recommendation Results

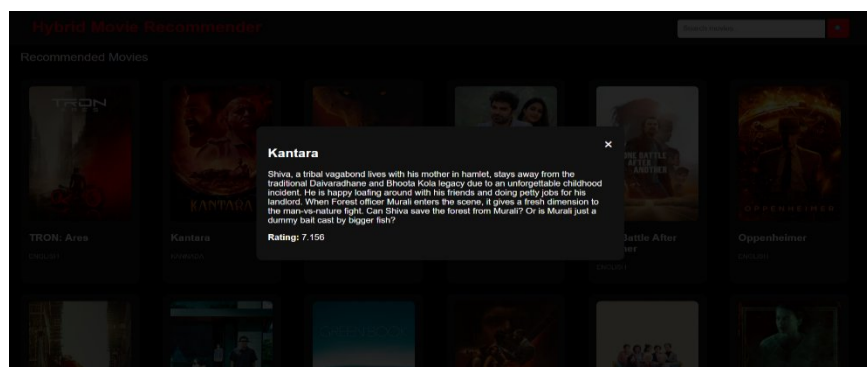


Figure 7: Movie Detail Pop-Up with Overview & Rating

[Figure 2 and 3: Hybrid Movie Recommender – Home Page (Netflix-Style UI)]

The home page displayed trending movies and top-rated titles in horizontally scrollable carousels with movie poster cards, titles, and language badges, mirroring the visual design language of commercial OTT platforms.

[Figure 4: All Movies Page with Genre, Year and Language Filters]

The full catalog view provided dropdown filters for genre, release year, and language, with sort controls for

DOI:10.15680/IJARETY.2026.1302125

alphabetical order, rating, and newest releases. Data was fetched dynamically from the Flask backend on each filter interaction without page refresh.

[Figure 5 and 6: Search and Recommendation Feature – Results for 'Kantara']

When a user entered 'Kantara' and triggered the recommendation engine, the system returned a diverse mix of Kannada, Telugu, and English language films sharing thematic similarities. The hybrid approach surfaced regionally relevant titles alongside globally popular picks.

[Figure 7: Movie Detail Modal – Overview, Genre, and Rating Display]

Clicking any movie card opened a modal overlay presenting the complete movie overview, genre tags, and audience rating, enabling informed viewing decisions without navigating away from the current browsing context.

V. CONCLUSION

This paper presented the Hybrid Multimodal Movie Recommender System, a web-based platform designed to address the content discovery challenge facing users of modern streaming platforms. By integrating content-based filtering through TF-IDF and cosine similarity with item-based collaborative filtering, the system overcomes the limitations of single-method recommenders—reducing over-specialization and mitigating cold start problems simultaneously.

The system demonstrated consistent performance across all tested scenarios: accurately generating thematically and audience-validated movie recommendations, maintaining responsive loading times under realistic network conditions, and correctly handling edge cases such as unrecognized movie inputs. The Netflix-inspired interface lowered the barrier to adoption by providing familiar visual and interaction patterns, while multi-language support extended the platform's relevance to regional cinema consumers across Indian languages.

The lightweight architecture—Flask backend, vanilla JavaScript frontend, TMDB API integration, and CSV-based dataset management—confirms that production-quality recommendation systems can be built and deployed without enterprise-scale infrastructure, making this approach accessible for academic projects, independent streaming platforms, and resource-constrained deployments. As future work, the platform can be extended to incorporate user authentication and personalized watch history persistence, neural collaborative filtering and transformer-based architectures for improved latent pattern discovery, real-time behavior tracking for session-aware recommendations, and cloud-based containerized deployment to support multi-user production loads. These enhancements would position the system as a comprehensive, scalable recommendation platform suitable for commercial OTT environments.

REFERENCES

- [1] Netflix Technology Blog. (2022). How Netflix Recommendation Works. Available: <https://netflixtechblog.com>
- [2] Amazon Science. (2023). Personalization at Amazon Scale. Available: <https://www.amazon.science>
- [3] .Pazzani, M. J., & Billsus, D. (2007). Content-Based Recommendation Systems. *The Adaptive Web*, Springer, pp. 325–341.
- [4] Schein, A. I., Popescul, A., Ungar, L. H., & Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. *ACM SIGIR Conference*, pp. 253–260.
- [5] Ramos, J. (2003). Using TF-IDF to Determine Word Relevance in Document Queries. *Proceedings of the First Instructional Conference on Machine Learning*.
- [6] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. *Proceedings of WWW '01*, ACM, pp. 285–295.
- [7] Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, 12(4), 331–370.
- [8] Adomavicius, G., & Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 734–749.
- [9] Flask Documentation. (2024). Flask Web Development Framework. Available: <https://flask.palletsprojects.com>.
- [10] The Movie Database (TMDB) API Documentation. Available: <https://developer.themoviedb.org/docs>.
- [11] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825–2830.
- [12] Harper, F. M., & Konstan, J. A. (2015). The MovieLens Datasets: History and Context. *ACM Transactions on Interactive Intelligent Systems*, 5(4), 1–19.

International Journal of Advanced Research in Education and Technology

ISSN: 2394-2975

Impact Factor: 8.152