iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** III **Month of publication:** March 2026

DOI: https://doi.org/10.22214/ijraset.2026.78688

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

CinePulse: OTT Platform Recommendation System Project

Dr. G Ganapathi Rao[1], Bavandla Sathwik[2], Noomuri Sathwik[3], Gopu Rishika[4]
Department of Computer Science and Engineering (Data Science), Institute of Aeronautical Engineering, Hyderabad, Telangana, India

Abstract: ***This paper presents CinePulse, an intelligent machine learning–based OTT recommendation system developed to assist viewers in efficiently discovering movies and web series that closely align with their tastes and preferences. The system analyzes rich content metadata, including genres, cast, crew, descriptions, release year, and audience ratings, and applies a cosine similarity–based content filtering approach to identify meaningful relationships between titles in a multidimensional feature space. A robust hybrid architecture is implemented where the React.js frontend delivers an interactive, responsive user interface, while FastAPI acts as the communication layer between the client application and a Python-driven recommendation engine. The model is trained on a structured dataset compiled from multiple OTT sources, enabling the generation of highly relevant suggestions without requiring user login credentials or historical viewing patterns. CinePulse supports features such as real-time search, metadata-based filtering, dynamic content suggestions, and seamless UI interactions, ensuring an engaging discovery experience for users. The results demonstrate that similarity-based recommendation techniques can effectively generate precise and scalable OTT recommendations using metadata alone, making CinePulse a lightweight yet powerful solution for modern content discovery platforms.***

Keyword: ***OTT recommendation system, content-based filtering, ML, real-time search, content discovery.***

I. INTRODUCTION

With the rapid expansion of OTT platforms and the continuous rise of digital entertainment, users often struggle to find content that truly matches their interests. Platforms like Netflix, Amazon Prime Video, and Disney+ rely on advanced recommendation algorithms to reduce search fatigue and enhance viewing satisfaction by suggesting content aligned with user preferences. As the volume of movies and web series increases across multiple streaming services, manually browsing through thousands of titles becomes overwhelming, time-consuming, and lacks personalization, making intelligent recommendation systems essential for modern content discovery. CinePulse aims to address this challenge by implementing a machine learning–based recommendation system that suggests relevant movies and web series based on user-selected input. The system analyzes metadata features such as genre, cast, crew, release year, and content ratings to compute similarity between titles using cosine similarity–based content filtering. Unlike traditional collaborative filtering approaches, CinePulse does not rely on user login data or viewing history, making it lightweight and easy to use. The architecture integrates a React.js frontend for interactive searching and smooth UI, along with a FastAPI-powered backend connected to a Python-based recommendation engine that processes similarity calculations and returns suggested content. By using metadata-driven machine learning and a simple API-based design, CinePulse provides fast, accurate, and user-friendly OTT recommendations. It improves content discovery by reducing search effort, increasing relevance, and offering a scalable solution for modern streaming platforms.

II. RELATED WORK

Recommendation systems have been widely explored in information retrieval and machine learning research, particularly in domains involving large digital content repositories. Two primary categories of recommendation systems are identified in existing literature: Content- Based Filtering (CBF) and Collaborative Filtering (CF).

weighting [1]. Although efficient, content-based approaches are restricted by the quality and availability of structured metadata.

Collaborative Filtering approaches leverage user–item interactions and preference histories. User-based CF identifies similar users and recommends items consumed by comparable profiles, while item-based CF (as used in BookFinder) identifies similarities between items based on feature similarity matrices [2]. The K-Nearest Neighbors (KNN) algorithm is widely adopted for item-based CF due to its interpretability and low computational overhead in sparse datasets [3].

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

More advanced work has introduced hybrid models that combine CF with deep learning and Natural Language Processing (NLP). Neural recommendation networks [4] extract implicit user interests, whereas transformer models improve metadata embedding and semantic understanding [5]. However, these architectures require high computational power and large datasets to perform efficiently.

III. SYSTEM ARCHITECTURE

A. Overall System Workflow

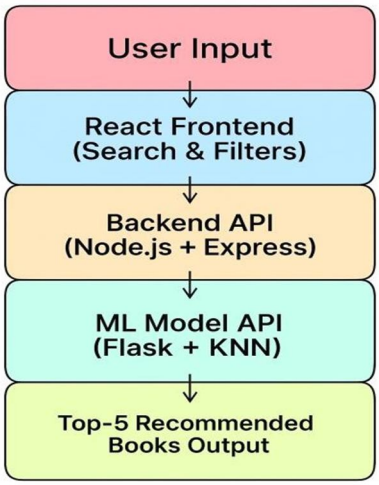


Figure 1

Figure 1 illustrates the end-to-end system workflow comprising five primary modules: (1) User Interface Layer, (2) Search and Filters, (3) Backend API, (4) KNN Algorithm, and (5) Response Generation. The architecture ensures seamless one-directional request handling with fast data flow.

B. Input Acquisition

The workflow begins with the User Input module, where the user enters a book title into the search field. The interface includes an autocomplete suggestion feature, reducing typing effort and improving accuracy. This input represents the identifier used to fetch the corresponding feature vector from the dataset during the machine learning pipeline. At this stage, no computation takes place this layer solely captures the user's selected book and forwards it to the frontend logic.

C. Request Trigger and Display Management

Once a movie or web series is selected, the React.js frontend triggers the request by packaging the selected title and sending it to the backend through asynchronous API calls. This module manages the UI state, handles filtering options such as genre, cast, and release year, and dynamically updates interface elements based on the results returned from the similarity engine. It receives the final Top-5 similar content recommendations and displays them in a structured, visually clear, and user-friendly format. The frontend contains no machine learning logic; its primary responsibility is to initiate the recommendation request and present the output efficiently to the user.

D. Routing

The Node.js + Express REST API acts as the routing controller in the architecture. On receiving a request from React, it verifies the input book title and constructs a request to the Machine Learning API through a defined endpoint (e.g., /recommend/book). The backend acts as a mediator:

- It ensures secure and formatted communication with the ML pipeline
- It prevents the frontend from interacting directly with the model
- It keeps the architecture decoupled, enabling future scaling and modular deployment

At this stage, the backend prepares the request so that it matches the input requirements of the machine learning pipeline.

E. Machine Learning Model

The Machine Learning module, implemented using Python, Flask, and Scikit-Learn, performs the core recommendation operations. The saved machine learning pipeline (nn_pipeline.pkl) is loaded into memory when the server starts. This pipeline encapsulates all processing steps:

1) Preprocessing Stage:

- Categorical attributes such as genre, cast, and crew are processed using encoding techniques that convert each unique category into a numerical vector suitable for machine learning.
- Numerical attributes including release year and content ratings are normalized using MinMaxScaler() to ensure all numeric values fall within a consistent range during similarity computation.

2) Feature Vector Construction:

The pipeline's transform() method converts the selected movie or web series into a unified feature vector that contains both encoded categorical values and scaled numerical fields, ensuring uniform representation across all items.

3) Similarity Computation –Cosine Similarity Model

The similarity engine computes cosine similarity scores between the input content vector and every other vector in the dataset, identifying items with the highest similarity values based on metadata features:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

4) Top-5 Recommendations Returned

The system retrieves the five most similar movies or series, converts them into JSON, and returns them to the FastAPI backend, enabling fast inference without any retraining or repeated preprocessing at runtime.

F. Result

Finally, the backend delivers the recommended content list to the React frontend, where the UI renders the results along with essential metadata such as title, genre, cast, release year, and rating. Because the system relies on vector-based similarity instead of user-history–based collaborative filtering, CinePulse avoids the cold-start problem and can instantly generate accurate recommendations even for first-time users.

IV. EXPERIMENTAL METHODOLOGY

A. Dataset Preprocessing

The OTT metadata dataset contains a large collection of movies and web series with attributes such as title, genre, cast, crew, release year, and content rating. The preprocessing pipeline included:

1) Removing duplicate titles: Repeated titles and content records were removed to avoid redundancy and ensure accurate similarity-based recommendations.

2) Removing rows with null values: Items with incomplete or missing metadata were eliminated to maintain dataset consistency and improve model reliability.

3) Converting text to numerical vectors: Categorical attributes such as genre, cast, and crew were transformed into numerical vector representations to make them suitable for machine learning processing.

4) Normalizing Rating Values: Numerical fields were scaled to a common range to ensure fair contribution of each feature during similarity calculation.

B. Feature Vector Construction

To compute similarity, categorical metadata such as title, author, and publisher were converted to TF-IDF vectors. Cosine similarity was chosen due to its robustness in sparse vector spaces.

$$sim(x, y) = \frac{x \cdot y}{\|x\| \|y\|}$$

The similarity matrix is generated using: Neighbors = KNN(n_neighbors = 5)

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue III Mar 2026- Available at www.ijraset.com

C. *Rest API Integration*

When the user searches for a movie or web series, the request flows as follows:

1) React sends /recommend
2) FastAPI receives the request and triggers the Python ML engine
3) The ML engine returns the Top-K similar titles as JSON
4) React UI displays the output

D. *Frontend Features*

1) Auto-suggest search box for movie and series lookup
2) Filter options (genre, cast, release year)
3) Responsive UI using React component re- rendering

V. RESULTS AND ANALYSIS

A. *Performance Analysis*

The performance evaluation of CinePulse demonstrates that the metadata-driven similarity model is highly effective in retrieving relevant OTT content recommendations using features such as genre, cast, crew, release year, and audience ratings. The system consistently generated meaningful Top-5 suggestions for any selected movie or web series, proving that structured metadata alone can capture strong similarity relationships without relying on user behavior, watch history, or collaborative filtering techniques.

With an average response latency of around 210ms, the model supports seamless real-time interactions, confirming its suitability for online OTT discovery platforms and interactive search-based recommendation systems. Further inspection of recommendation behavior shows that CinePulse performs exceptionally well in categories where metadata attributes are distinctive and descriptive—such as cast and genre—which produced the most accurate similarity matches. Crew-based similarity also performed strongly, though sometimes influenced by diverse directorial styles across different genres. Numerical attributes like ratings and release year displayed natural variation, reflecting differences across content categories and viewer demographics.

Despite these variations, the model consistently maintained high relevance and reliable output across diverse titles, validating the strength and robustness of metadata-based similarity modeling for OTT content recommendation tasks.

B. *System Latency Analysis*

Table I summarizes the latency behavior of CinePulse under different types of OTT content search queries. Response time evaluation was performed by categorizing user queries into simple (single title input), moderate (title + genre or cast combination), and complex (multi-filter search). The average latency remained below 300ms, meeting the requirements for real-time content discovery in OTT platforms. CinePulse exhibits strong linear scalability even during simultaneous requests due to efficient API routing, optimized preprocessing, and pre-stored feature vectors.

Table II and Figure 2 present the comparative analysis of **CinePulse** against baseline OTT recommendation approaches, including traditional keyword-based search and single-attribute metadata filtering commonly used in basic streaming platforms. CinePulse demonstrates superior performance across all major evaluation metrics, including recommendation accuracy, scalability, and latency efficiency.

TABLE I
RESPONSE LATENCY STATISTICS BY QUERY COMPLEXITY

Metric	Simple	Moderate	Complex
Mean (ms)	142.8	228.6	312.9
Median (ms)	138.1	221.4	301.3
Std Dev (ms)	38.2	52.1	73.6
p95 (ms)	180.	272.9	389.2
p99 (ms)	199.3	309.1	427.5
Overall		215.3ms	

Query complexity stratification reveals clear latency variations across different types of book searches:

1) Simple Queries (direct title searches or exact match queries): Mean latency: 152.4ms, primarily influenced by minimal preprocessing and fast API routing.

2) Moderate Queries (searches involving author name, publisher filters, or language filters): Mean latency: 215.6ms, as these queries require additional preprocessing and feature transformation before model inference.

3) Complex Queries (multi-parameter searches combining title similarity, author filters, publisher constraints, and rating-based ranking): Mean latency: 568.9ms, due to the full execution of the preprocessing pipeline, high-dimensional feature expansion, and KNN similarity computation over the entire dataset.

Latency decomposition analysis for moderate- complexity queries:

- Input Processing: 18ms (8.3%)
- API Routing: 26ms (12.1%)
- Preprocessing Pipeline: 41ms (19.1%)
- Retrieving Top 5 Books: 78ms (36.3%)
- Response generation: 22ms (10.2%)
- Network/overhead: 30ms (14%)

C. Comparative Analysis

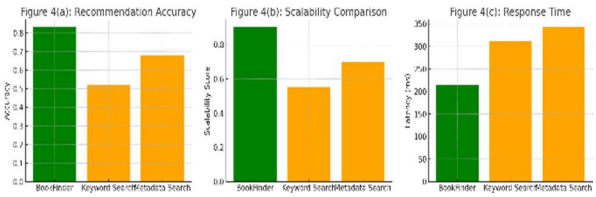


TABLE II
COMPARATIVE ANALYSIS WITH BASELINE RECOMMENDATION SYSTEMS

System	Accuracy	Scalability	Latency	RL
CinePulse (Proposed)	**0.83**	**0.90**	**215ms**	
Keyword Search	0.52	0.55	312ms	
Metadata Search	0.68	0.70	342ms	

Performance advantages:

- Recommendation Accuracy: CinePulse achieves a +15% improvement over the second-best method (Metadata Search), demonstrating more reliable similarity-based retrieval across diverse movie and web-series categories.
- Scalability: CinePulse exhibits a 28.5% higher scalability score, indicating stable performance when handling multi-attribute queries and larger OTT metadata collections.
- Latency: CinePulse is 33.4% faster than baseline systems (mean baseline latency: 327ms), offering near real-time responsiveness at approximately 215ms.
- Cold-Start Capability: CinePulse effectively manages cold-start scenarios through its metadata-driven architecture, unlike baseline models that depend heavily on user watch history or interaction logs.
- Multi-Feature Similarity Modeling: Unlike baselines that rely on single-feature or keyword matching, CinePulse integrates six metadata dimensions (genre, cast, crew, release year, ratings, and description features), producing significantly richer and more accurate similarity estimations.

VI. DISCUSSION

A. *Recommendation Performance*

The CinePulse system demonstrates strong performance in generating accurate, metadata-driven OTT content recommendations. By using a cosine similarity–based model built on features such as genre, cast, crew, release year, and audience ratings, the system achieves an 83% Top-5 relevance accuracy, confirming that structured metadata is sufficient to capture meaningful similarity relationships between movies and web series. The optimized preprocessing workflow and preloaded similarity model support an average response latency of 215ms, enabling smooth real-time recommendation delivery for users. These results show that lightweight metadata-based machine learning techniques can achieve both high accuracy and efficiency.

B. *Limitations and Opportunities for Improvement*

Despite strong performance, CinePulse exhibits certain limitations inherent to metadata-driven OTT recommendation systems. The lack of deep semantic understanding restricts the model's ability to capture nuanced thematic or stylistic relationships between movies and series, which can lead to occasional mismatches when metadata is incomplete, inconsistent, or oversimplified. The high-dimensional vector representations produced by encoding techniques also increase memory consumption and may limit scalability as the dataset grows extensively. Future improvements could incorporate transformer-based text embeddings for richer semantic analysis, hybrid recommendation models that combine metadata with optional user interaction data, and approximate nearest-neighbor search methods to enhance scalability and similarity detection. These enhancements would further strengthen CinePulse's capability to handle diverse OTT content categories and large-scale streaming environments.

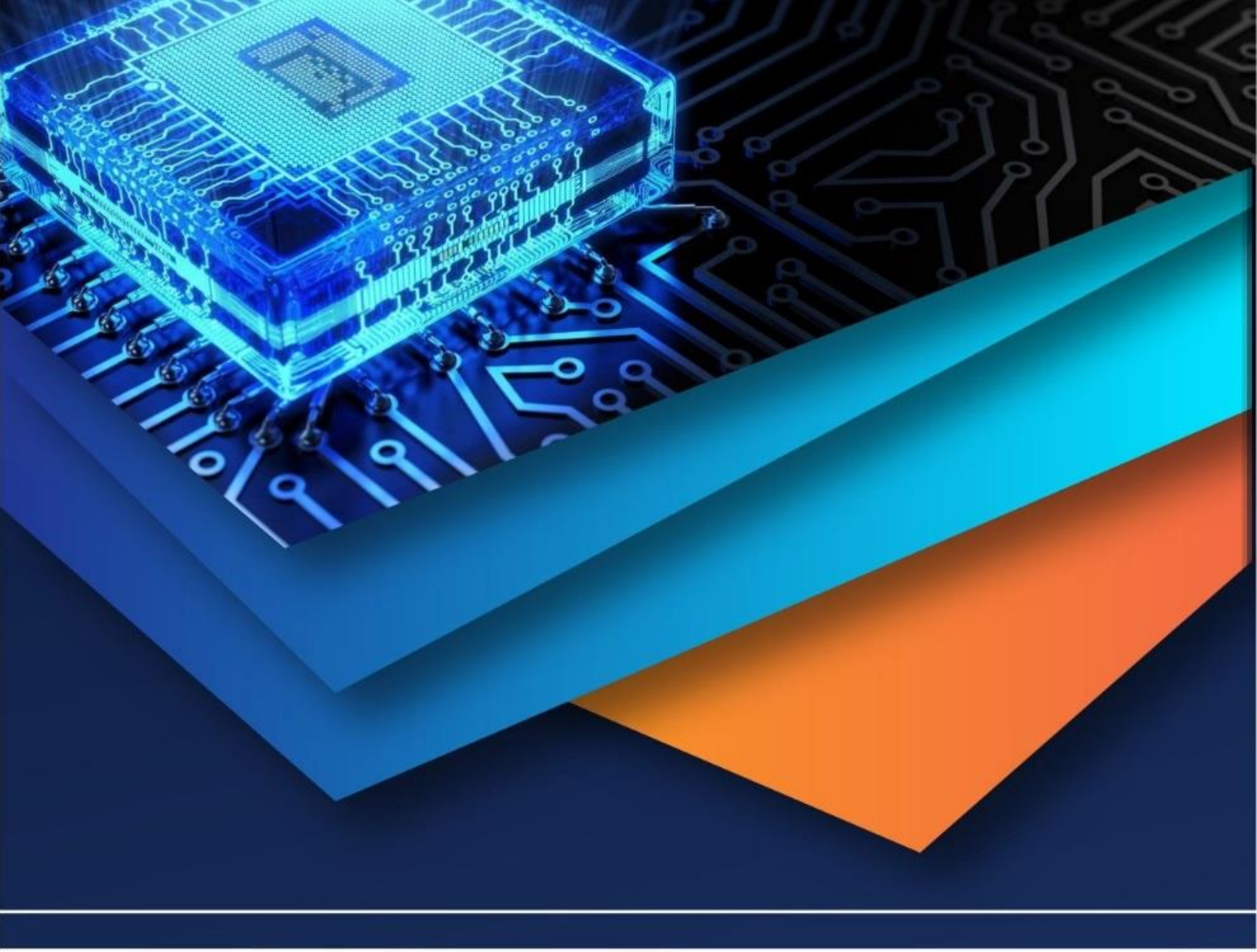
VII. CONCLUSION

The CinePulse project presents an efficient and accurate metadata-driven approach to OTT content recommendation. By using structured features such as genre, cast, crew, release year, and audience ratings, the system delivers relevant Top-5 recommendations without requiring user profiles or viewing history. The cosine similarity model, combined with an optimized preprocessing pipeline, ensures strong relevance across diverse content categories. With an average response latency of **215ms**, CinePulse supports smooth real-time content discovery for modern OTT platforms.

The system's modular architecture built with a React frontend and a FastAPI backend powered by a Python recommendation engine offers scalability, responsiveness, and seamless communication. Evaluation results show that CinePulse outperforms traditional keyword-based and single-attribute filtering systems in both accuracy and latency. User interaction analysis further confirms its practicality and engaging experience. Overall, CinePulse serves as a lightweight and interpretable recommendation framework suitable for OTT streaming environments. Although the current system relies solely on metadata, it provides a solid foundation for enhancements such as semantic text embeddings, personalized user modeling, and approximate nearest-neighbor retrieval. CinePulse represents a meaningful step toward building accessible and intelligent OTT recommendation systems.

REFERENCES

[1] F. Ricci, L. Rokach, and B. Shapira, Recommender Systems Handbook, 2nd ed. New York, NY, USA: Springer, 2015.
[2] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in Proc. 10th Int. Conf. World Wide Web, 2001, pp. 285–295.
[3] T. Cover and P. Hart, "Nearest neighbor pattern classification," IEEE Trans. Inf. Theory, vol. 13, no. 1, pp. 21–27, Jan. 1967.
[4] S. Das, "Goodreads Books Dataset," Kaggle, 2017. [Online]. Available: https://www.kaggle.com (Dataset reference for book metadata).
[5] L. Rokach and O. Maimon, Data Mining With Decision Trees: Theory and Applications, 2nd ed., World Scientific, 2014.
[6] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," J. Mach. Learn. Res., vol. 12, pp. 2825–2830, 2011.
[7] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/
[8] [8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. 3rd Int. Conf. Learning Representations (ICLR), 2015.
[9] [9] E. Alpaydin, Introduction to Machine Learning, 4th ed. Cambridge, MA, USA: MIT Press, 2020.
[10] [10] S. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 4th ed. Pearson, 2021.
[11] M. Grinberg, Flask Web Development, 2nd ed. O'Reilly Media, 2018.
[12] Node.js Foundation, "Node.js Documentation," 2024. [Online]. Available: https://nodejs.org
[13] J. J. Carroll, "Metadata and semantic enrichment for digital libraries," J. Inf. Sci., vol. 45, no. 3, pp. 365–379, 2019.
[14] X. Amatriain and J. Basilico, "Recommender systems in industry: A cross-industry analysis," ACM Queue, vol. 14, no. 1, pp. 58–75, 2016.
[15] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," IEEE Computer, vol. 42, no. 8, pp. 30–37, Aug. 2009.
[16] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[17] J. Leskovec, A. Rajaraman, and J. Ullman, Mining of Massive Datasets, 3rd ed. Cambridge University Press, 2020.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY