Journal of Science, Computing and Engineering Research (JSCER)
Volume-9, Issue- 4, June 2026.
DOI: *https://doi.org/10.46379/jscer.2023.060503*

MOVIE RECOMMENDATION SYSTEM

[1]J.R.Arun Kumar,[2]Ms Tanu Saini,[3]Tanisha gupta,[4]Unnati,[5]Tamanna

[1]Professor, Department of CSE, Modern Institute of Technology and Research Centre, Rajasthan, India.

[2,3,4,5]UG Student, Department of CSE, Modern Institute of Technology and Research Centre, Rajasthan India

Article Information

Received : Mar 28 2026
Revised : Mar 29 2026
Accepted : Mar 31 2026
Published : Apr 1 2026

Corresponding Author:
Ms Tanu Saini

***Abstract*—** This project presents a personalized movie recommendation system built using the MERN (MongoDB, Express.js, React, Node.js) stack integrated with a machine learning model. The system leverages content based filtering to suggest movies tailored to individual user preferences, such as previously liked genres, cast, and storylines. It utilizes movie metadata—including title, genres, description, actors, and directors—to generate feature vectors that drive similarity-based recommendations.

Keywords: Keywords: Movie Recommendation System, Machine Learning, MERN Stack, Content-Based Filtering, Cosine Similarity

Citation: : J.R.Arun Kumar, Ms Tanu Saini, Tanisha Gupta, Unnati, Tamanna, "MOVIE RECOMMENDATION SYSTEM", Journal of Science, Computing and Engineering Research, 9(3), March2026.

I. INTRODUCTION

In the era of digital media, users are constantly exposed to a vast library of movies and entertainment content. As streaming platforms continue to expand, users often face the challenge of choosing content that aligns with their personal preferences. To address this, intelligent recommendation systems have become an essential component of modern web applications, enabling personalized content delivery and enhanced user engagement.

This project aims to develop a Personalized Movie Recommendation System using the MERN stack (MongoDB, Express.js, React.js, and Node.js) combined with a Machine Learning model based on content-based filtering techniques. The system recommends movies to users based on their previous preferences, favourite genres, and metadata associated with each film such as genre, description, and cast**.**

A. The Core Mechanism

The core of the recommendation logic is implemented using Python's machine learning libraries. It analyses user behaviour and metadata to generate similarity scores using TF-IDF (Term Frequency-Inverse Document Frequency) and cosine similarity techniques. These scores help the system identify and suggest movies that are most like the user's interests.

II. PROBLEM STATEMENT

With the exponential growth of digital content and online streaming platforms, users are overwhelmed by the vast number of available movies. This abundance of options often leads to decision fatigue, where users struggle to find content that matches their interests. Traditional search and filter methods are time-consuming and ineffective for delivering truly personalized experiences. There is a need for an intelligent system that can understand individual user preferences and recommend movies accordingly. Moreover, many existing recommendation platforms either rely on expensive APIs or offer limited access to full-length movies without subscriptions or licensing.

A. The Role of the MRS

- Delivers tailored movie suggestions based on content similarity and user preferences.
- Uses a content-based filtering approach with a machine learning model to analyze movie metadata (such as genre, cast, and description).
- Integrates legally accessible, full-length public domain movies from YouTube, allowing users to watch movies within the platform.
- Is built using the MERN stack for seamless integration of frontend, backend, and database

components, with a machine learning model connected to enable dynamic recommendation generation.

B. The Debilitating Disadvantage

The core problem addressed by this industrial training project is the lack of integrated, practical exposure to building complete end-to-end applications. Many learners can create static front-end pages or write simple server-side scripts independently, but they struggle to understand how real applications flow from the interface to the server and then to the database.

III. PROPOSED MODEL

A. Objectives of proposed model

- To develop a web-based platform that allows users to receive personalized movie recommendations.
- To implement a content-based filtering model using TF-IDF and cosine similarity to generate suggestions based on user input.
- To embed full-length movies from YouTube that are legally available (public domain or Creative Commons).
- To integrate machine learning with a modern web stack (MERN) for seamless user experience and scalability.

B. Key Features:

User Preference Input

Users can manually select favourite genres, liked movies, or keywords representing their interests. This structured input helps the system construct a preference profile and initiates the recommendation process even for first-time users.

Content-Based Recommendation Engine

A Python-based machine learning model processes metadata such as titles, genres, cast members, keywords, and plot descriptions. Using TF-IDF vectorization and cosine similarity, the engine ranks movies based on semantic similarity, delivering recommendations tailored to user preferences.

MERN Stack Integration

- React.js manages interactive UI components.
- Node.js and Express.js handle API routing, request processing, and communication with the ML model.
- MongoDB stores movie metadata, user inputs, and system configurations. This integration ensures consistent performance, maintainability, and ease of deployment.

Video Embedding

Using React Player, the application streams full-length YouTube videos directly within the interface. This eliminates the need for hosting media files, reduces system load, and ensures compliance with legal streaming guidelines.

Extensibility

The system's modular design allows future enhancements such as:

• User account creation

• Ratings and reviews

• Hybrid recommendation models

• Multi-language movie descriptions

• Mobile app extension via React Native This flexibility makes the proposed system adaptable for long-term expansion.

C. Workflow Overview:

1) User accesses the React frontend and selects movie preferences (liked titles or genres).
2) Frontend sends the preferences to the Node.js backend via a RESTful API.
3) Backend communicates with the Python ML model, which processes metadata and returns a list of similar movies.
4) Backend recieves recommendations and sends them back to the frontend.
5) Frontend displays the recommended movies and allows users to watch fulllength films directly embedded from YouTube.

D. Advantages Of Proposed Model

- Lightweight and Fast Recommendation Engine
- Legally Compliant Streaming
- Modular and Scalable Architecture
- High Personalization Even With Limited User Data

PROPOSED DIAGRAM

The Proposed Diagram illustrates the structural relationship among the software components.

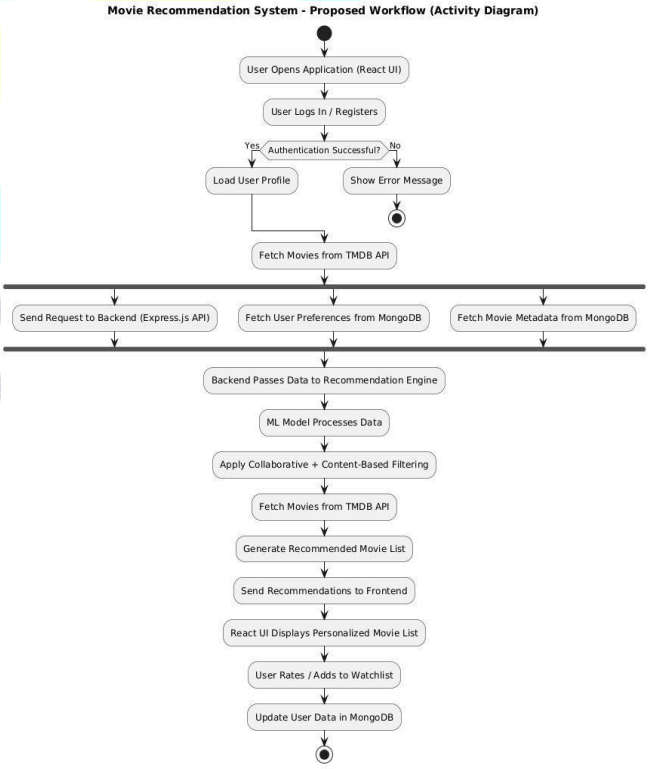

Fig. 2. Proposed Diagram

IV. TECH STACK

The proposed Movie Recommendation System is built using the MERN Stack, combined with additional tools and frameworks to ensure high performance, scalability, user friendliness, and efficient recommendation processing. The complete technology stack is divided into Frontend, Backend, Database, Software Tools. Each technology has been carefully selected based on its compatibility with system requirements and its ability to support the dynamic functionalities described in the system architecture.

A. Frontend Technologies

The frontend acts as the user-facing layer of the movie recommendation system. It is responsible for presenting movie lists, authentication screens, personalized suggestions, and user interactions in a clean and interactive interface.

B. Backend Technologies

The backend manages user authentication, movie data retrieval, recommendation generation, rating storage, and all communication between frontend and database.

C. Database Technology

The database is responsible for storing movie data, user information, ratings, genres, watchlists, and recommendation metadata. Smooth and optimized database performance ensures real-time interaction inside the application.

C. Technology Used

MongoDB Atlas:

- A cloud-based NoSQL database service.
- Stores data in flexible JSON-like documents, ideal for movie metadata and user profiles.

• Benefits:

o Scalable storage for large movie datasets.

o Schema-less structure allows dynamic addition of new movies or features.

o Optimized for quick read/write operations required for personalized recommendations.

o Easily integrates with Node.js and Express.js through Mongoose ORM.

E. Software Tools & Additional Technologies

❖ **Git & GitHub:**

Used for source code management and version control.

Supports collaborative development and deployment tracking.

❖ **Postman:**

Used for testing backend APIs such as login, register, rating submission

❖ **Mongoose (ODM):**

Object Data Modeling library for MongoDB.

Simplifies schema creation, validations, and database queries.

❖ **JWT (JSON Web Tokens):**

Provides secure user authentication and session management.

V. RESULT SCREENSHOTS

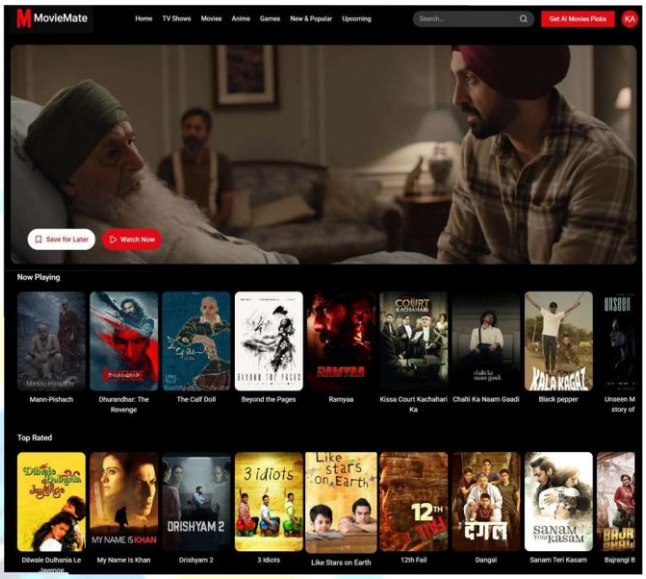

Fig. 2. Home Page

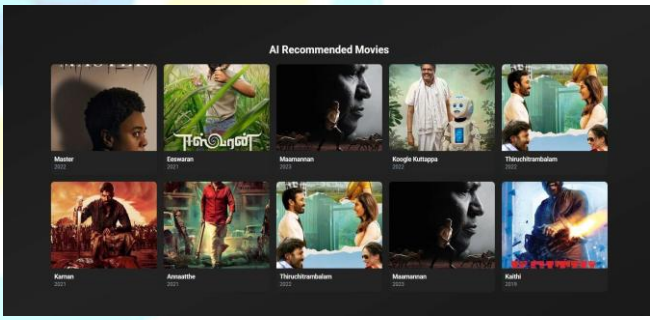

Fig. 2. Output Page

VI. CONCLUSION

A. Project Summary

The development of the Movie Recommendation System using the MERN stack and Machine Learning demonstrates an effective integration of modern web technologies with intelligent data-driven algorithms. The system successfully utilizes content-based filtering techniques to analyze movie attributes such as genre, cast, and keywords, transforming them into vector representations for similarity computation. By employing cosine similarity, the system generates personalized and relevant movie suggestions for users, thereby improving their overall browsing experience.

B. Key Achievements

The proposed system achieves several important objectives. It provides accurate and efficient recommendations based on user preferences without requiring extensive user history. The use of the MERN stack ensures a scalable and responsive architecture, where the frontend offers an interactive interface and the backend efficiently handles data processing and API requests. Additionally, the integration of machine learning enhances the intelligence of the system, making recommendations more meaningful and user-centric.

C. **Final Remarks**

In conclusion, the Movie Recommendation System addresses the problem of information overload in modern streaming platforms by offering a simple yet effective solution for personalized content discovery. The system can be further enhanced by incorporating collaborative filtering, hybrid approaches, and real-time user behavior analysis. Overall, this project highlights the potential of combining web development frameworks with machine learning techniques to build practical, scalable, and user-friendly applications.

REFERENCES

1) React official guides :- "React – A javascript library for building user interfaces," Available : https://react.dev/
2) TMDB API Documentation :- TMDB open source API is is used to provide basic information related to movies like -region language,cast , ratings etc. https://developer.themoviedb.org/
3) Research Papers on Recommender Systems :- GroupLens Research , "MovieLens Dataset"Available : https://grouplens.org/datasets/movielens/
4) MongoDB and Express.js official guides :- MongoDB is NoSQL database used to store data in Json -like format .Express.js is a light weight framework for Node,js used to built API and web applications.
5) Node.js documentation :- "Node.js – Javascript runtime built on Chrome's V8 engine",Available : https://nodejs.org/