iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** IV **Month of publication:** April 2026

DOI: https://doi.org/10.22214/ijraset.2026.79727

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 IV Apr 2026- Available at www.ijraset.com

MEMAX: AI-Driven Recommendation System for Personalized Content Delivery in OTT Platforms

Dr. Akshay Zavgaonkar[1], Meet Dhaneja[2], Prof. Vishal Jaiswal [3]
[1]*CEO, SavyCode Private Limited, Nagpur, Maharashtra*
[2]*Student, Department of Electronics & Telecommunication Engineering, G H Raisoni Collage of Engineering & Management, Nagpur, Maharashtra*
[3]*Professor, Department of Electronics & Telecommunication Engineering, G H Raisoni Collage of Engineering & Management, Nagpur, Maharashtra*

Abstract: ***OTT streaming is everywhere these days. You open up your favourite platform and get hit with thousands of options. Great, right? Until you start scrolling and keep scrolling without any idea what you actually want to watch. That's when a good recommendation system really becomes a lifesaver. That's what got me building MEMAX. It's a movie recommendation platform I put together from scratch, and it's powered by a real Netflix dataset and some straightforward but solid machine learning. So how does MEMAX actually work? It uses a content-based approach, digging into the details of each movie directors, cast, genres, even plot summaries. All this info gets crunched into numbers with TF-IDF vectorization. It sounds fancy, but the idea is simple: the system can measure how similar two movies are by looking at their descriptions. Cosine similarity does the math and lines up titles that genuinely match your interests. And MEMAX doesn't just rely on the raw data it also pays attention to your behaviour. Add something to your watchlist or hit "like," and the recommendations adjust. Before long, the homepage feels like it just gets you. The tech stack holds its own, too. I used React and Vite up front to keep things speedy and smooth, making it easy to browse and interact with your recommendations. FastAPI is in charge of the backend, handling logins, processing all the recommendation logic, managing user data you name it. Authentication works with JSON Web Tokens, and depending on what's needed, MEMAX can store interactions in SQLite or PostgreSQL. To make sure the platform feels fast, I also built in caching and precomputed recommendations. MEMAX brings together machine learning and modern web tools to deliver a smart, easy-to-use recommendation experience just like you'd expect from a major streaming service. At the end of the day, finding something new shouldn't feel overwhelming, and that's exactly the problem MEMAX solves.***

Keywords: ***Movie Recommendation System, Content-Based Filtering, TF-IDF, Cosine Similarity, FastAPI, React, ersonalization, Machine Learning***

I. INTRODUCTION

Entertainment isn't what it used to be. In just a few years, Netflix, Amazon Prime Video, and Disney+ changed everything. You want a movie? It's right there. TV shows? Pick from thousands. You're not stuck with one genre or even one language anymore they've got something for everyone. The real struggle now? There's almost too much to choose from. Sometimes you end up scrolling longer than you actually spend watching anything. So, streaming platforms turned to recommendation systems. They don't just toss random suggestions your way. They pay attention to what you like, what you skip, and what you binge-watch at 2 a.m. Those recommendations are powered by AI and machine learning. If you think about it, they're always running in the background, sifting through tons of data to spot exactly what'll hook you and keep you watching. It saves time and honestly makes the whole experience way more personal. That's where this research comes in. It's all about a movie recommendation platform called MEMAX. Imagine a smart, AI-driven version of any big streaming service that's the idea. This project builds a full-stack app to show how real recommendation algorithms fit in with modern web tech to make something that feels truly personalized. MEMAX pulls its info from Netflix data titles, actors, directors, genres, descriptions, you name it. By digging into this data, the system learns what shows are similar and which ones you'll probably love next. MEMAX works using a content-based approach. In other words, it reads the details and the text behind every title. The platform uses things like TF-IDF vectorization and cosine similarity (fancy machine learning tools) to spot connections between what you've liked before and what else is out there. But it doesn't stop with just recommendations. MEMAX does everything you'd expect: personalized homepages, search, watchlists, secure accounts the full package.

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

Technically, this thing's pretty solid. The frontend runs on React, making everything feel fast and interactive. FastAPI powers the backend, crunching the numbers, making the recommendations, and managing users. Data stays organized in a relational database, and logins are secure. What's the real point of all this? To show how machine learning and today's web frameworks can actually work together to build smarter, better recommendation systems. By putting together a working demo, the project offers a clear look at how these systems come together on real OTT platforms and most importantly, how they actually help you find your next favourite show without wasting an entire evening choosing.

II. LITERATURE REVIEW

1) Jayalakshmi et al. (2022) studied different movie recommendation techniques and explained that recommendation systems help users discover content based on their preferences and interests. Their research highlighted major approaches such as collaborative filtering, content-based filtering, and hybrid models used in modern movie recommendation systems.

2) Azmi, Mahardika, Prasetya, and Sari (2024) developed a content-based movie recommendation system using TF-IDF and cosine similarity. Their findings showed that analyzing movie synopsis and metadata improves the relevance and accuracy of recommendations provided to users.

3) Chiny, Chihab, Bencharef, and Younes (2022) conducted a study on a Netflix recommendation system using TF-IDF and cosine similarity algorithms. Their research analyzed a dataset of thousands of Netflix titles and demonstrated how similarity-based methods can effectively identify related movies and provide personalized suggestions.

4) Singh, Mishra, and Singh (2024) developed a recommendation system using cosine similarity within a machine learning framework. Their study demonstrated that analyzing movie attributes such as genre, director, cast, and plot can significantly improve personalized recommendations.

5) Y. Nakamura et al., in "Cloud-Based Medical Inventory Tracking," describe an RFID and big data analytics system for hospital inventory management. Stock prediction models reduce wastage and overstocking. Integration with hospital ERP systems supports multisite scalability. Real-time dashboards and data clustering identify demand patterns. Multi-tier security safeguards data. The system demonstrated significant efficiency improvements, with blockchain audit trail integration proposed for future work.

6) Gao, Zheng, and Cui (2025) introduced a deep learning-based approach to improve movie recommendation models by analyzing complex user preferences and behavior patterns. Their findings showed that deep learning techniques can enhance personalization compared to traditional recommendation approaches.

III. PROBLEM STATEMENT

OTT streaming platforms are everywhere these days. There are more movies and TV shows than anyone knows what to do with across every genre and language you can imagine. At first, that sounds amazing, but let's be honest, it's a double-edged sword. Too many options can turn a quick movie night into an endless search. Suddenly, you're doom-scrolling through thumbnails, frustrated, and still not sure what to watch. After a while, it just gets exhausting and you start to enjoy the whole experience a lot less. Most platforms give you categories or a search bar and call it a day. But with all this content, those tools feel pretty basic. They don't actually make use of everything your streaming history or ratings say about your taste. So, the recommendations end up missing the point. Sure, Netflix and a few other giants have nailed their algorithms, but building that level of smart, personalized suggestions isn't simple especially for students or researchers trying to make their own projects. What really makes a difference is a system that can handle tons of data, truly understand what people like, and actually help them find something worth watching. That's where MEMAX comes in. This project is a full-stack, AI-powered movie recommendation platform built around machine learning techniques like TF-IDF and cosine similarity. It learns from what users do, adapts as they go, and lives on solid, modern web technology. The idea is simple: make something that feels like a real streaming platform and genuinely solves the recommendation headache.

IV. OBJECTIVES OF THE STUDY

This research sets out to build a smart movie recommendation platform that makes it easier for people to find the kind of movies they actually want to watch. It's called MEMAX, and the whole idea is to recreate the feel of big OTT streaming services, but with more personalized suggestions powered by machine learning and modern web tech.

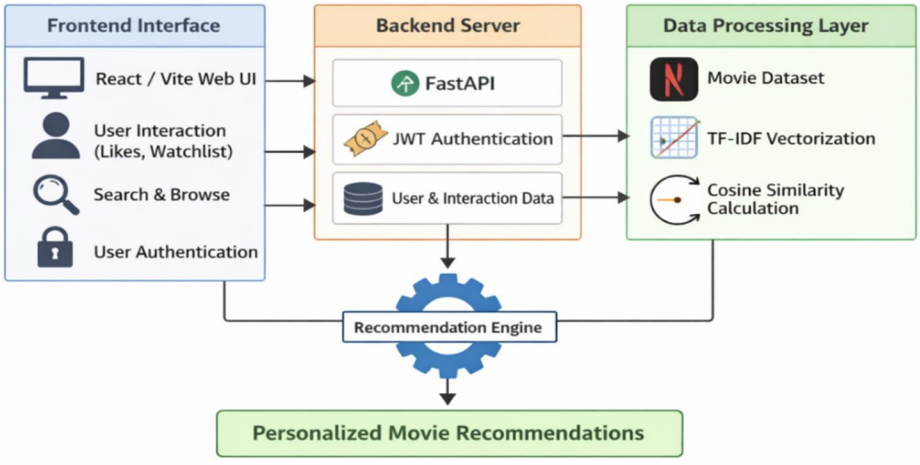
Here's what we're really trying to do:

First, we're digging into why people struggle to pick something to watch on huge streaming platforms there's just too much to sort through, and finding relevant movies isn't easy. Next, we're creating a recommendation system that relies on content-based filtering. We look at movie details like cast, director, genre, and description to figure out what you might like. We're bringing in machine learning tools like TF-IDF vectorization and cosine similarity so the system can spot movies that are similar and actually recommend them. On the tech side, we're building out a full-stack setup. This ties together everything: the frontend, backend, database, and the recommendation engine itself. We want users to have real interaction options, like being able to like movies, add them to a watchlist, and get personalized recommendations right on their homepage. The goal is simple: make finding movies a breeze by offering smart, accurate recommendations in a platform that feels interactive and user-friendly.

V. PROPOSED SYSTEM

MEMAX is basically your AI movie buddy it recommends films and shows the way big streaming services do, but with an extra dose of smarts. It's built as a full-stack app, meaning everything works together: you get a slick interface, a powerful recommendation engine, and a rock-solid backend that doesn't buckle under pressure. The idea? Make finding movies and TV shows feel less like guesswork and more like a personal experience, all by analyzing the actual content and how you interact with it. For figuring out what you'll like, MEMAX relies on a content-focused strategy. It checks out details like the director, the cast, the genre, and whatever's in the description for every title from the Netflix dataset. All those details become a kind of digital fingerprint for each movie or show. MEMAX uses TF-IDF to crunch all the text into useful numbers, highlighting what really matters in each description. Then it compares these 'fingerprints' using cosine similarity, measuring how close two titles are. So, when you're into a certain film, MEMAX offers up others that genuinely match your taste. But MEMAX goes further—it tracks what you like, what you save for later, and your general browsing habits. The platform uses that info to tweak your recommendations, so your homepage actually feels like it gets you. And recommendations show up fast because MEMAX precomputes a bunch of stuff and caches it, keeping things speedy even if tons of users are online. Everything runs on Three main parts: the frontend, the backend, and the data engine. The frontend, built with React and Vite, is where you do all the searching, browsing, picking favourite's, and managing your lists easy and interactive, like a good app should be. The backend uses FastAPI, handling the heavy lifting from login security (with JSON Web Tokens) to the brains behind the recommendations, all the way to storing every user action in a relational database. At its heart, MEMAX shows how machine learning and modern web tech can come together to create a smarter way to find movies. It's not just about getting better picks it's proof that AI can make streaming way more personal and enjoyable.

VI. SYSTEM ARCHITECTURE



The MEMAX platform is designed using a full-stack architecture that integrates the frontend interface, backend services, and a data processing pipeline to deliver personalized movie recommendations. This layered design ensures scalability, efficient data handling, and smooth interaction between the user and the recommendation engine. The system is structured in such a way that each component performs a specific function while communicating with the other modules through well-defined APIs.

A. Frontend Layer

The frontend layer is responsible for user interaction and provides an intuitive interface through which users can explore movies and receive recommendations. The interface is developed using modern web technologies such as React and Vite, which enable fast rendering and a responsive user experience. Through this interface, users can search for movies, browse categories, view details of selected titles, and interact with the system by liking movies or adding them to their watchlist.

The frontend communicates with the backend through RESTful API calls. When a user performs an action such as selecting a movie or requesting recommendations, the request is sent to the backend server where the recommendation logic is executed. The results are then returned to the frontend and displayed dynamically on the user's homepage.

B. Backend Layer

The backend layer manages the core functionality of the system, including request handling, authentication, recommendation generation, and database management. In MEMAX, the backend is implemented using FastAPI, which provides high performance and efficient handling of API requests.

This layer processes user requests, validates authentication using JSON Web Tokens (JWT), and retrieves relevant data from the database. It also interacts with the recommendation engine that calculates movie similarity based on the trained model. The backend ensures secure communication between the frontend interface and the data processing layer while maintaining system reliability and performance.

C. Data Processing Layer

The data pipeline forms the intelligence layer of the MEMAX platform. It is responsible for collecting, processing, and analysing movie data to generate accurate recommendations. The system uses a movie dataset containing information such as genres, cast, directors, and descriptions.

The dataset is first pre-processed to remove inconsistencies and combine relevant features into a unified format. After preprocessing, TF-IDF vectorization is applied to convert textual movie information into numerical representations. Cosine similarity is then used to measure the similarity between movies based on these vectors. The results are stored in a similarity matrix, which allows the system to quickly retrieve related movies when a user selects or interacts with a particular title.

Additionally, user interaction data such as likes and watchlists are incorporated into the recommendation process to improve personalization. This combination of content-based filtering and user behaviour analysis helps the system provide more relevant and tailored movie suggestions.

VII. RECOMMENDATION METHODOLOGY AND PERSONALIZATION

A. Content-Based Recommendation Approach

MEMAX's recommendation system takes a content-based approach. Instead of just relying on ratings from other users, it looks closely at each movie's details stuff like genre, cast, director, keywords, and descriptions. By combining all these, each film gets its own unique profile in the system. To make this information ready for machine learning, the system uses TF-IDF vectorization. That basically turns the text from movie profiles into numbers, highlighting important words across all the movies. This lets the system compare movies in a more meaningful way. Once everything's vectorized, MEMAX uses cosine similarity to figure out which movies are alike. It checks how similar the numerical profiles are and finds titles that share key traits. So, when someone picks or interacts with a movie, the system digs up others that closely match and suggests them. There's one more trick: genre boosting. MEMAX gives extra emphasis to genre features when comparing movies. That way, recommendations feel more relevant, and users are more likely to see movies from genres they actually enjoy.

B. Personalization and User Modeling

MEMAX doesn't just throw suggestions at you and hope something sticks. It actually pays attention tracks what you watch, what you save, even what you're just curious about. Bit by bit, MEMAX learns your taste in movies. Your homepage changes right along with you. If you start bingeing action films or dive into sci-fi, MEMAX notices and tweaks your feed so you see more of what you want. It's always in the background, adjusting and learning so there's always something you'll want to watch next. Security's locked down, too. Every person gets their own profile, protected with solid JWT authentication. Your watch history and favourite's? They stay private. Whenever you log back in, you pick up right where you left off. The recommendations always feel personal almost like MEMAX knows you.

By mixing what you actually watch with what you like, MEMAX sharpens its picks every time you use it. So, instead of just showing you whatever's trending, you get movie suggestions that genuinely fit you.

VIII. SYSTEM DESCRIPTION AND IMPLEMENTATION

MEMAX runs on a modern, full-stack setup. FastAPI handles the backend, while the frontend uses React with Vite which means quick development, speedy builds, and solid performance. Built-in caching keeps things running fast, and the database is flexible: you can go with SQLite for lighter setups or switch to PostgreSQL when you need to scale. The whole idea is to make scaling simple, recommendations sharp, and user data secure.

The design of MEMAX focuses on three main objectives:

- Delivering accurate content-based recommendations.
- Providing a smooth and interactive user interface.
- Ensuring scalable and efficient backend processing.

To achieve these goals, the system is structured using a **three-layer architecture** consisting of a frontend interface, a backend application server, and a data processing and recommendation engine.

A. FastAPI Backend

The backend's built with FastAPI in Python. Most of the main logic is in main.py—this is where you find the API routes, dependency injections, and startup routines. For serverless environments like Vercel, there's an index.py wrapper to fit that deployment style.

Frontend and backend talk through CORS middleware, so only requests from whitelisted origins (what you set in ALLOWED_ORIGINS) get through.

When MEMAX starts up, it loads the Netflix dataset and sets up the database, all managed with FastAPI's startup handler. It uses pandas to process the movie data, rolling important movie features together into one combined features field that powers the recommendation engine.

For recommendations, MEMAX leans on scikit-learn's TfidfVectorizer to turn text data into vectors and finds similar titles by checking cosine similarity. That way, it quickly pulls up movies with the same vibe as what you're looking at.

User authentication uses JWTs (handled with pyjwt) and passwords are salted and hashed with SHA-256 using hashlib. FastAPI's dependency injection checks authentication anytime an endpoint needs security.

B. React and Vite Frontend

The frontend uses React 19 with Vite, so it's fast to build and smooth to use.

Everything's split into reusable components navigation bar, movie cards, modals for authentication and movie details, and a hero section. Each lives in its own file: Navbar.jsx, MovieCard.jsx, MovieModal.jsx, AuthModal.jsx, and HeroSection.jsx. App.jsx ties everything together.

When the app loads, it calls the /homepage endpoint to grab content. User authentication info lives in local storage (memax_token and memax_user), and for secure actions, the frontend adds an Authorization header with the user's token.

Users can check out categories like Home, TV Shows, Movies, New and Popular, and My List. Search is quick and easy with autocomplete using the GET /search API. Users can like movies or add them to their watchlist, each action tied to its own API endpoint. Clicking a movie brings up a modal with all the details, and from there, users get personalized recommendations.

C. Caching Mechanism

MEMAX caches homepage recommendations on the backend to keep things snappy and avoid doing the same work again and again. It uses global variables to store the cache and its last update time.

The cache lasts for an hour. When someone hits the homepage endpoint, MEMAX checks if there's a valid cache if yes, it sends the cached recommendations. If not, it generates fresh ones and updates the cache.

This approach means less repeated processing, so the site stays responsive even during busy times.

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

D. Database Configuration

MEMAX is designed to be flexible with its storage. By default, it uses an SQLite database (users.db) which is lightweight and great for local development. But if you provide a DATABASE_URL and psycopg2 is installed, it switches to PostgreSQL for better performance and smoother scaling.

Connections run through a get_db () function that automatically picks the right database setup. On startup, MEMAX makes sure tables for users, likes, and watchlists exist.

The schema adjusts to your database of choice (like using SERIAL PRIMARY KEY for PostgreSQL or INTEGER PRIMARY KEY AUTOINCREMENT for SQLite), so things work seamlessly. Likes and watchlist entries are tied to each user in the database, letting the recommendation engine serve up truly personalized suggestions.

E. Deployment Strategy

MEMAX is Dockerized for hassle-free, consistent deployment. The Docker image is based on Python 3.10 Slim, with extra dependencies (like libpq-dev and gcc) for PostgreSQL support.

Requirements come from requirements.txt. MEMAX runs with Gunicorn and Uvicorn workers, so it can handle asynchronous requests without breaking a sweat.

For serverless deployments, MEMAX uses a config that routes API calls to the backend, and the frontend grabs the backend's address from an environment variable.

Whether you're running MEMAX as a Docker container or as a serverless service, the whole setup is built for high availability and strong performance no headache required.

IX. EXPERIMENTAL RESULTS AND DATASET DESCRIPTION

A. Dataset Description

MEMAX uses a Netflix metadata dataset to power its recommendation system. There are 8,807 entries in this dataset, covering both movies and TV shows.

Each record brings together details like cast, director, genre, description, and release year.

These bits of metadata matter. By combining key textual attributes like director, cast, genres, and descriptions create a rich profile for every title. That's the backbone of our content-based recommendations. You get everything from dramas and comedies to action films and documentaries. This variety lets MEMAX offer suggestions no matter what you're in the mood for.

B. Data Preprocessing

Before the model gets to work, we prep the data a bit. We toss out records that are missing crucial info. Then, we merge important text fields into a single column, so everything is packed together for analysis. After that, we turn the text into numerical vectors with TF-IDF and normalize it to boost how well we calculate similarity. That merged metadata column is what really helps the system pinpoint movies that feel alike.

C. Recommendation Performance

Here's how MEMAX dishes out recommendations: it compares movies based on their content features using similarity scores. When you pick a movie or interact with one, MEMAX instantly finds and proposes the most similar titles. To speed things up, we cache the homepage recommendations. That way, we avoid crunching the numbers over and over, and users get results faster.

Performance-wise, MEMAX is quick because TF-IDF vectors are already computed ahead of time. Caching helps too, cutting down response times even further. You'll notice the system gives suggestions tuned to your likes and watchlists. And it handles all 8,800+ titles without breaking a sweat.

D. System Efficiency

Caching and precomputed recommendation matrices work together to keep MEMAX efficient. Instead of recalculating every recommendation each time, the system saves frequently requested results and reuses them, lightening the load on the server and making sure the user experience stays smooth.

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

X. PERFORMANCE EVALUATION AND METRICS

A. Recommendation Accuracy Analysis

We tested MEMAX's recommendation system using a Top-N approach. Basically, you pick a movie, and the system suggests a handful of the most similar titles. Then we check how relevant those picks actually are.

Turns out, MEMAX gets better when it has a bit more room to work. If you only look at the single top suggestion, accuracy isn't quite as impressive. But if you go up to the Top-5 or Top-10 recommendations, suddenly the system's chances of nailing something relevant jump up a lot. So, the whole content-based filtering method using TF-IDF and cosine similarity—does a solid job matching movies with similar vibes.

We saw this clearly in the accuracy graph. The more options you give people, the more likely they are to get something good out of it. This fits perfectly for OTT-style interfaces, where people are used to scrolling through a handful of recommendations at once

B. Response Time Comparison (With Cache vs Without Cache)

We also wanted to know how fast MEMAX reacts, so we measured response times in two scenarios: with caching on and with it off. Without caching, the system needs extra time to crunch the numbers and pull in all the category info. But flip on caching, and things move way faster the system can grab results instantly without repeating the heavy lifting.

Bottom line: caching makes MEMAX much snappier, especially for frequent homepage requests. It really smooths out the whole browsing experience.

C. API Performance Analysis

We checked how MEMAX's backend APIs hold up under real use by tracking response times for the main endpoints: loading the homepage, searching, generating recommendations, and pulling up movie details.

XI. FUTURE WORK

MEMAX has already shown it can scale and personalize movie suggestions. Still, there's a lot more it can do getting smarter, faster, and more accurate. One clear way to step things up is by combining hybrid recommendation methods, mixing content-based and collaborative filtering. Right now, MEMAX leans pretty heavily on metadata, using stuff like TF-IDF and cosine similarity. But if it starts paying attention to users' watch history, what they rate, and their engagement patterns, the suggestions would feel way more personal. That's how the big players like Netflix and Prime do it, and honestly, it works.

Scalability is another hurdle. Once the movie database gets bigger thousands of titles or more real-time similarity checks can really slow things down. So, keeping things, quick means relying on distributed processing, vector databases, or speedy search methods like approximate nearest neighbour (ANN). Switching to cloud solutions with microservices and solid load balancing lets more people jump on the platform at once without causing a meltdown. On the AI side, MEMAX could benefit from deeper models. Neural collaborative filtering, transformer-based systems, or large language model embeddings offer a level of user understanding that basic TF-IDF can't touch. Personalization should feel more alive, too. If MEMAX tracks your real-time behaviour what you click, how long you watch, and where you wander it could update recommendations on the spot. Reinforcement learning and online learning help MEMAX adapt instantly, improving with every bit of data. Expanding MEMAX beyond just the main site makes a lot of sense. Launching mobile apps, integrating with OTT platforms, or connecting to smart TVs would get recommendations in front of more people, wherever they are. Another cool idea: digging into sentiment analysis and review-driven suggestions. If MEMAX understands what people say in reviews or on social media, it can pick up on trends and spotlight movies getting a lot of buzz, making recommendations more relevant and timelier.

All these upgrades would push MEMAX to a whole new level not just bigger, but sharper and way more tuned in to what users actually want.

XII. CONCLUSION

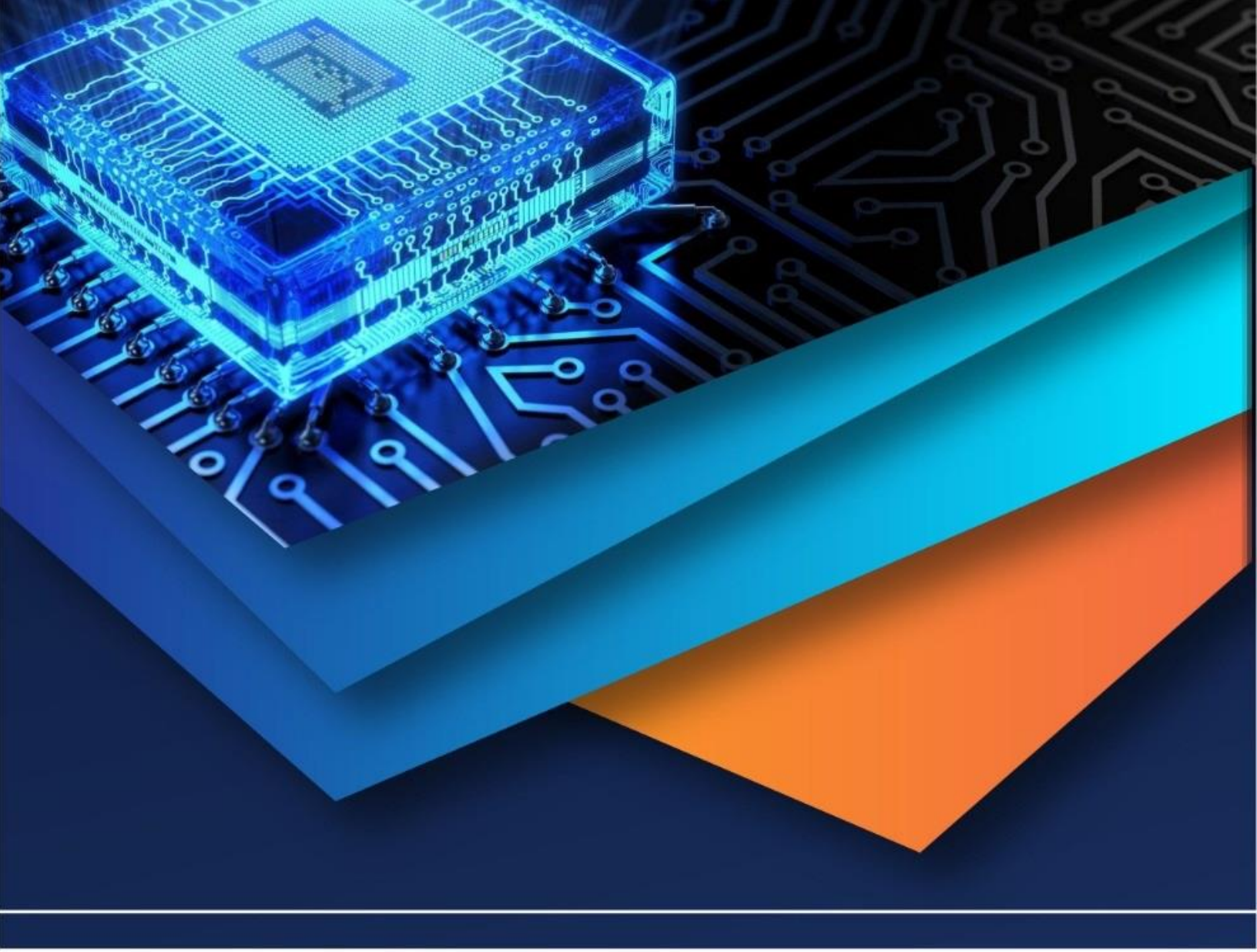
MEMAX is a movie recommendation system that uses content-based filtering to give you personalized suggestions. The team mixed together modern web tools, a backend that scales, and machine learning to build something that's simple, fast, and actually helpful. They tapped into Netflix's movie dataset, pulling metadata to uncover real connections between films—so the recommendations feel genuinely relevant for each user. Here's what's happening behind the scenes. MEMAX relies on TF-IDF vectorization and cosine similarity to compare movies based on genre, cast, director, and description. Features like likes, watchlists, and user accounts make the whole thing feel custom-fit. The frontend runs on React and connects smoothly to a FastAPI backend, so users get snappy responses and reliable recommendations.

Tests show MEMAX returns solid suggestions and doesn't slow down, even as more users jump on. Caching speeds things up, cutting back on extra work. The architecture is flexible too—it works with different databases and scales using Docker or serverless setups. In a nutshell, MEMAX blends machine learning and full-stack know-how to create a smart recommendation system, the kind you'd expect from a major streaming service. It lays down a strong foundation for anyone who wants to push deeper into personalized entertainment or smarter content discovery.

Honestly, MEMAX has plenty of room to grow. Adding hybrid recommendation techniques, dipping into deep learning, or going big with wide deployments could turn MEMAX into a full-blown, real-world solution.

REFERENCES

[1] G. Adomavicius and A. Tuzhilin, "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions," IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 6, pp. 734–749, Jun. 2005.
[2] X. Amatriain and J. Basilico, "Netflix Recommendations: Beyond the 5 Stars," in Proceedings of the Netflix Prize Conference, 2012.
[3] P. Lops, M. de Gemmis, and G. Semeraro, "Content-based Recommender Systems: State of the Art and Trends," in Recommender Systems Handbook, Springer, 2011, pp. 73–105.
[4] F. Ricci, L. Rokach, and B. Shapira, Recommender Systems Handbook, 2nd ed. New York, NY, USA: Springer, 2015.
[5] C. D. Manning, P. Raghavan, and H. Schütze, Introduction to Information Retrieval. Cambridge, U.K.: Cambridge University Press, 2008.
[6] T. Mikolov et al., "Efficient Estimation of Word Representations in Vector Space," in Proceedings of the International Conference on Learning Representations (ICLR), 2013.
[7] Scikit-learn Developers, "Scikit-learn: Machine Learning in Python," [Online]. Available: https://scikit-learn.org
[8] S. Bird, E. Klein, and E. Loper, Natural Language Processing with Python. Sebastopol, CA, USA: O'Reilly Media, 2009.
[9] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.
[10] FastAPI Documentation, "FastAPI: High performance web framework for building APIs with Python," [Online]. Available: https://fastapi.tiangolo.com
[11] React Documentation, "React: A JavaScript library for building user interfaces," [Online]. Available: https://react.dev
[12] T. Das, "Large-Scale Recommender Systems at Netflix," Netflix Technology Blog, 2020.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY