**Use Case Title**: AI- powered Movie Recommendation

System

**Student Name**: MOHAMMED SAAD V

**Register Number**:510623104064

**Institution**: C ABDUL HAKEEM COLLEGE OF ENGINEERING AND TECHNOLOGY

**Course &Department**: B.E CSE

**Date of Submission**: 19-05-2025

# 1.Problem Statement

With the exponential growth of online streaming platforms, users face a major challenge: content overload. With thousands of movies available, users often waste time deciding what to watch, leading to frustration and platform fatigue. The lack of personalized suggestions reduces user engagement and satisfaction. There is a clear need for a smart system that understands user preferences and delivers tailored recommendations.

**2. Proposed Solution**

We propose building an AI-powered recommendation system that uses machine learning algorithms to predict and suggest movies tailored to user tastes. The system will incorporate two core techniques:

Content-Based Filtering: Recommends movies similar to ones a user has liked, using features like genres, descriptions, and keywords.

Collaborative Filtering: Suggests movies based on similarities between user ratings and behaviors, even when content features differ.

**Features to include:**

Movie search bar.

Display of top 10 similar or recommended movies with posters.

Interactive, user-friendly UI built using Streamlit or Flask.

Option to select recommendations by mood, genre, or actor in future versions.

This solution addresses the problem by minimizing decision fatigue and maximizing entertainment value through intelligent automation.

**3. Technologies & Tools Considered**

Programming Language: Python Libraries & Frameworks:

Pandas, NumPy – for data processing

Scikit-learn – for ML models and similarity calculation

Streamlit or Flask – for UI development

Requests, TMDB API – for movie poster retrieval

**Data Sources:**

MovieLens Dataset – user ratings and metadata

TMDB API – movie posters and detailed info

ML Techniques:

TF-IDF Vectorization (text similarity)

Cosine Similarity (closeness of movies)

SVD / Matrix Factorization (latent features of user-movie interaction)

Deployment Platforms:

Streamlit Cloud

Render.com

GitHub Pages for documentation

**4. Solution Architecture & Workflow**

1. Data Ingestion: Load MovieLens dataset and preprocess it
2. Feature Extraction:

Generate TF-IDF vectors for descriptions, keywords, and genres.

Build a user-item interaction matrix.

1. **Similarity Computation:**

Apply Cosine Similarity for content-based filtering.

Use Matrix Factorization for collaborative filtering.

1. **Recommendation Logic:**

Given a user-selected movie or preferences, return similar titles.

1. **UI Interface:**

Streamlit app: search bar, results section, and poster display.

1. **Deployment:**

Hosted app with GitHub repository including README and setup

Optional: Add a flowchart or architecture diagram showing data flow from user input to recommendation output.

**5. Feasibility & Challenges**

**Feasibility:**

Uses open datasets and accessible APIs.

Readily implementable using Python and popular libraries.

Lightweight enough for fast deployment and use on standard systems.

**Challenges:**

Cold-start problem: New users or new movies may lack sufficient data.

Scalability: With very large datasets, performance tuning is needed.

Diversity vs Accuracy: Overfitting to specific tastes can reduce content diversity.

API limitations: TMDB API has request limits; may require caching or alternatives.

**Mitigation Strategies:**

Use fallback methods for cold-start (e.g., trending movies).

Introduce caching for API responses.

Add diversity metrics in future models.

**6. Expected Outcome & Impact**

User Experience: Quick, personalized movie suggestions reduce user frustration and enhance viewing satisfaction.

Platform Benefit: Increases engagement time and user retention for streaming platforms.

Educational Value: Excellent case study for applying ML to real-world recommendation problems.

Community Benefit: Open-source code can help other developers build or improve similar systems.