Hackathon Submission – level 1

## Use Case Details

**Use Case Title :** AI-Powered Movie Recommendation System

**Student Name : SWETHA B**

**Register Number :** 511323244010

**Institution :** Kingston Engineering College

**Department :** Computer Science And Business System

**Date of Submission :** 21-05-2025

# Problem Statement:

In the age of digital streaming, entertainment platforms host an enormous volume of movies across all genres, languages, and time periods. Users face a critical issue—**decision fatigue**—when choosing what to watch. Most users spend more time browsing than actually watching content. Existing recommendation engines on platforms like Netflix, Prime Video, and others rely on basic collaborative filtering, often leading to **repetitive, non-contextual suggestions**.

Moreover, these platforms lack **emotional intelligence**; they do not take into account a user's current mood, situational context, or evolving taste. For example, a user who enjoys action movies might want a calming drama one evening, but the system keeps pushing more action titles. There's also minimal attention to niche preferences or temporary viewing

desires.

Hence, there's a compelling need for an **intelligent, mood-aware, and context-**

**sensitive recommendation engine** that can adapt in real-time and offer a personalized movie discovery experience.

# Proposed Solution:

We propose **CineMatch**, an **AI-powered movie recommendation engine** that

dynamically adapts to user behavior, context, and preferences using machine learning and natural language processing.

CineMatch supports three core modes:

* **By Movie**: Users enter a movie they love. The system finds similar films using metadata analysis (genre, theme, cast, plot) and embeddings.
* **By Genre**: Users select one or more genres (e.g., Sci-Fi, Thriller, Romance), and CineMatch filters top-rated and trending content accordingly.
* **By Mood**: Users select a mood (e.g., Happy, Sad, Adventurous), and te system uses NLP to align emotions with movie themes and deliver emotionally relevant suggestions.

## Key Features:

* Personalized dashboard with adaptive filtering
* Emotion-to-movie mapping using sentiment classifiers
* Integration with TMDB API for real-time metadata
* Continuous learning based on user feedback and ratings

This creates a **human-like recommendation system** that evolves with the user and dramatically improves the movie discovery process.

# Technologies & Tools Considered:

To bring CineMatch to life, a robust technology stack is used:

## Front-End:

* **HTML, CSS, JavaScript**: Build the UI with interactive forms, dropdowns, and grids.
* **React.js (optional enhancement)**: For dynamic rendering and real-time search results.

## Back-End:

* **Python + Flask/Django**: Framework to handle APIs, user sessions, recommendation logic.

## Machine Learning:

* **Scikit-learn**: For collaborative and content-based filtering algorithms.
* **spaCy/NLTK**: NLP libraries used for mood classification and genre analysis.
* **Pandas/NumPy**: Data preprocessing and matrix operations.

## Database:

* **SQLite**: Lightweight, embedded database for prototyping.
* **MongoDB**: NoSQL option for user preferences and logs in production.

## APIs & External Resources:

* **TMDB API**: To fetch movie metadata (ratings, genres, posters).
* **Sentiment Lexicons**: For mood-to-movie mapping.

## Deployment:

* **Heroku/AWS/GCP**: Cloud platforms for live deployment, scalability, and load balancing.

# Solution Architecture & Workflow:

## System Components:

1. **User Interface (UI)**:
   * Allows user interaction via search bars, filters, and mood selectors.

## Recommendation Engine:

* + Implements collaborative filtering (based on user similarity).
  + Uses content-based filtering (movie metadata and keywords).
  + Maps moods to genres using NLP and sentiment analysis.

## Database:

* + Stores user interaction logs, watch history, feedback.
  + Maintains session-level mood state for adaptive recommendations.

## External APIs:

* + Connects to TMDB to fetch up-to-date movie data, including posters, cast, and trailers.

## Server & APIs:

* + Built in Flask/Django. Serves endpoints for fetching movie suggestions.

## Cloud Layer:

* + Hosted with CI/CD pipelines, ensuring seamless updates and performance monitoring.

## Workflow:

* User inputs a movie, genre, or mood.
* Backend processes this using trained models and mood mappings.
* Movies are fetched from local DB or TMDB based on match confidence.
* Final results are rendered as dynamic cards on the front-end.

# Feasibility & Challenges:

## Feasibility:

* Open APIs and libraries make it accessible for student-level development.
* Modular architecture allows easy testing and future integration into OTT platforms.
* Scalable deployment via cloud services ensures readiness for real-world usage.

## Challenges:

* **Data Sparsity**: New users may not have enough data for personalized recommendations.
* **Model Accuracy**: Matching moods with abstract movie themes is inherently subjective.
* **Privacy**: Storing and analyzing user behavior must comply with data protection norms.
* **Real-time Performance**: NLP and ML processing should not delay results.

## Mitigations:

* Use hybrid recommendation (combine collaborative and content-based).
* Preprocess mood-to-genre maps to reduce computation.
* Encrypt user data and apply anonymization methods.

# Expected Outcome & Impact:

The implementation of CineMatch offers several high-impact outcomes:

* **Reduced Decision Time**: Users quickly find movies they’re likely to enjoy.
* **Higher Engagement**: Contextual suggestions increase watching duration and satisfaction.
* **Scalable Use**: Can be adapted for web apps, mobile apps, smart TVs.
* **Commercial Integration**: OTT platforms can use CineMatch to improve retention and satisfaction metrics.
* **Inclusive Experience**: Mood-based and genre-based inputs serve a wider audience demographic.

This system represents a **shift from generic to empathetic recommendations**, aligning technology with user psychology.

# Future Enhancements:

To evolve CineMatch into a commercial-grade engine:

* **Voice Interface**: Integrate voice commands like “I want a feel-good movie” using speech-to-text + sentiment parsing.
* **Social Graph Integration**: Recommend based on friends’ watch history and ratings.
* **Cross-Language Support**: Enable recommendations in local languages using translation APIs.
* **Watch History Visualization**: Display timelines, genre heatmaps, and emotion trends.
* **Adaption to TV Series**: Expand the engine to suggest binge-worthy series based on time availability.

# Appendix: Visual Overview of CineMatch:

## Sample Recommended Movies:

1. **The Shawshank Redemption**
   * Rating: 8.4 | Genre: Drama | Duration: 142 min
   * Overview: Redemption and friendship in prison life.

## The Dark Knight

* + Rating: 9.0 | Genre: Action | Duration: 152 min
  + Overview: Batman faces his nemesis Joker in a psychological battle.

## Pulp Fiction

* + Rating: 8.9 | Genre: Crime | Duration: 154 min
  + Overview: Interconnected stories with gritty dialogues and moral conflicts.

## Interstellar

* + Rating: 8.6 | Genre: Sci-Fi | Duration: 169 min
  + Overview: Humanity’s survival mission across galaxies.