A Lightweight Hybrid Movie Recommender with Mood Adaptation and Group Preference Modeling

Presented by :

UCE2023638 - Tejal Kunde
UCE2023647 - Rutuja Narode
UCE2023661 - Shreya Dubey

PROBLEM STATEMENT

- **Large number of movies → users struggle to decide what to watch**

- **Existing platforms often give recommendations that:**
  -Do not account for user mood
  -Do not balance content similarity with user-rating behavior
  -Do not support multiple users choosing together
  -Show many movies that are too similar to each other

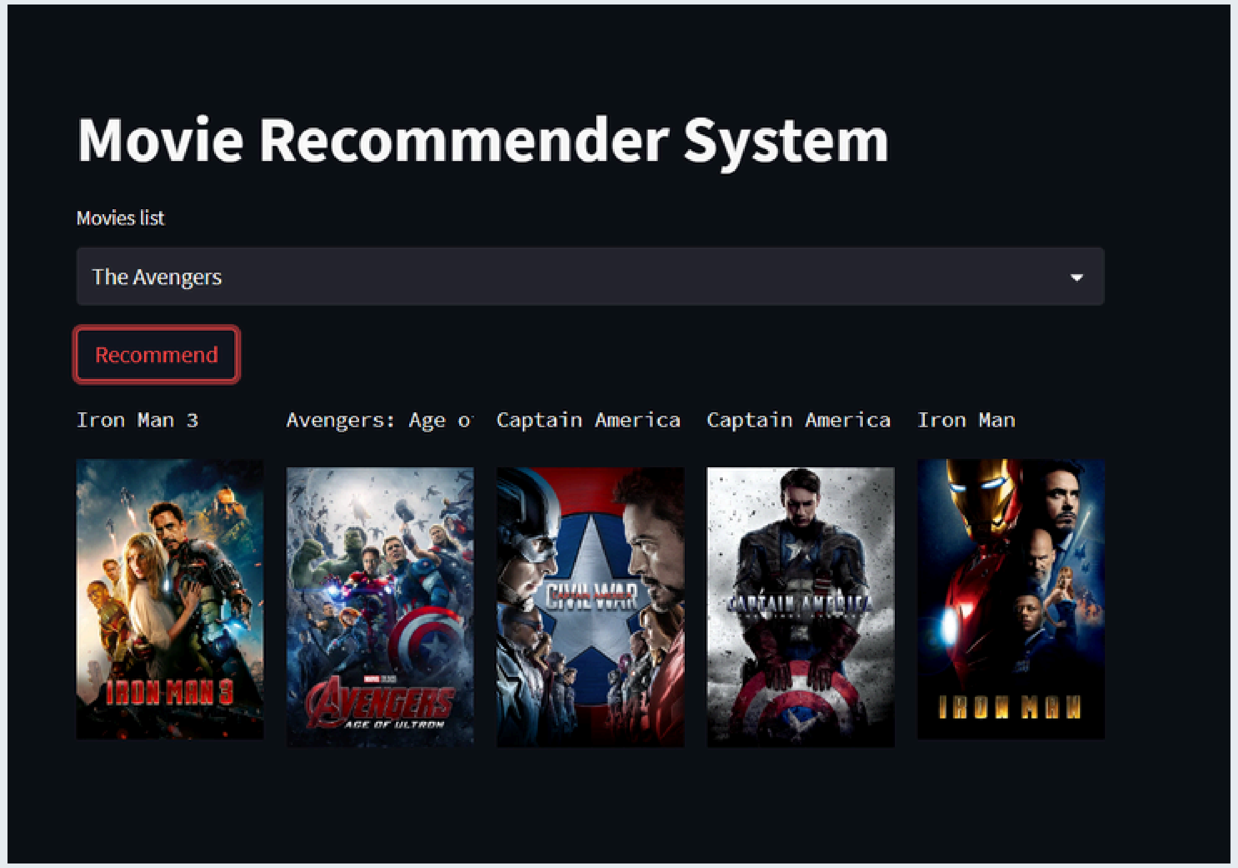
- **There is a need for a system that:**
  -Understands mood from movie descriptions
  -Uses content similarity (overview, genre, cast, keywords)
  -Incorporates rating-based collaborative signals
  -Can generate group-friendly recommendations
  -Ensures diversity in results

RESEARCH OBJECTIVES

Build a hybrid recommender using:

- TF-IDF on overviews (for mood & content similarity
- Cosine similarity for content matching
- User rating matrix for collaborative filtering
- **Generate:**

  -Top-N content-based recommendations
  -Hybrid score combining both
  -Mood-filtered recommendations
  -Group-based common recommendations
  -Diverse recommendation list



Movie Recommender System

Movies list

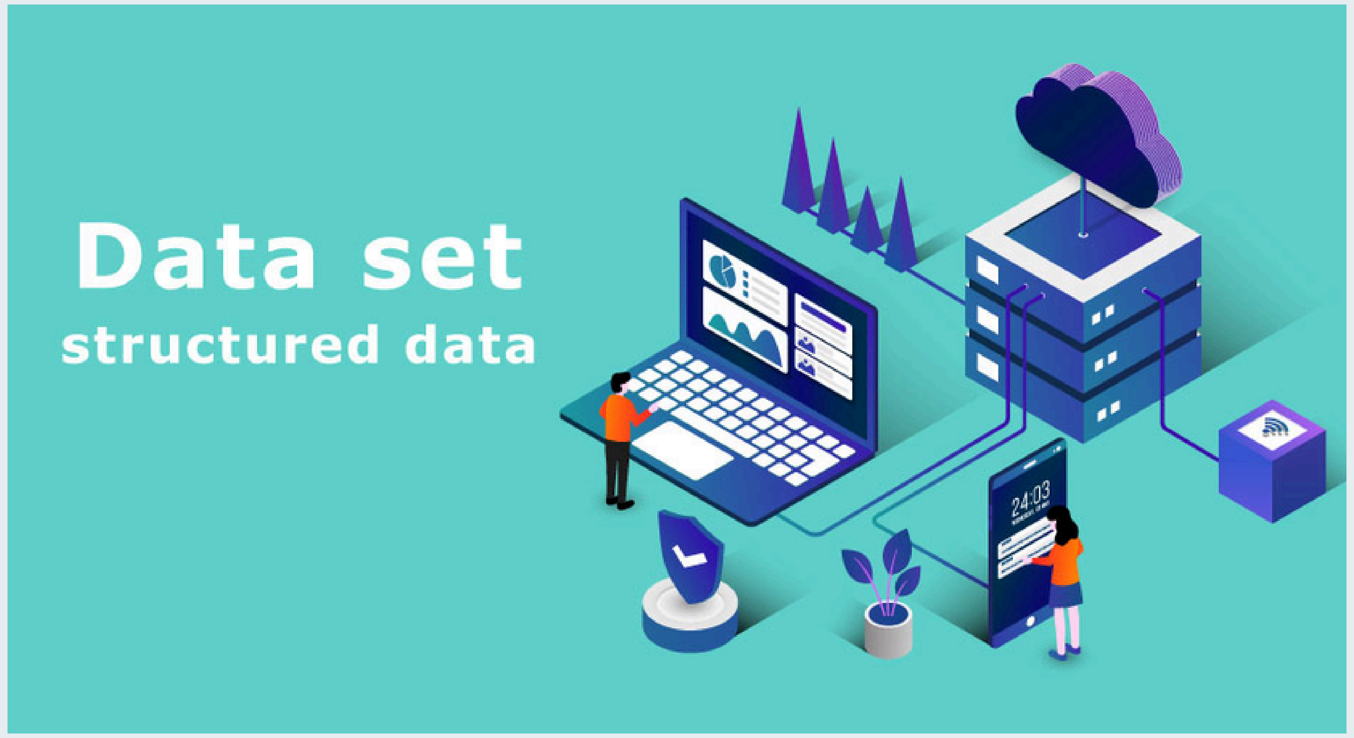
The Avengers

```
Iron Man 3    Avengers: Age o    Captain America    Captain America    Iron Man
```

DATASET DESCRIPTION

Datasets Used

- tmdb_5000_movies.csv
- tmdb_5000_credits.csv



Merged on: title

Fields extracted in code:

1. Genres
2. Cast (top 3)
3. Crew → Director
4. Keywords
5. Overview
6. Production companies

This becomes the metadata for similarity/mood extraction.

PREPROCESSING STEPS

1. **Created combined metadata field:**

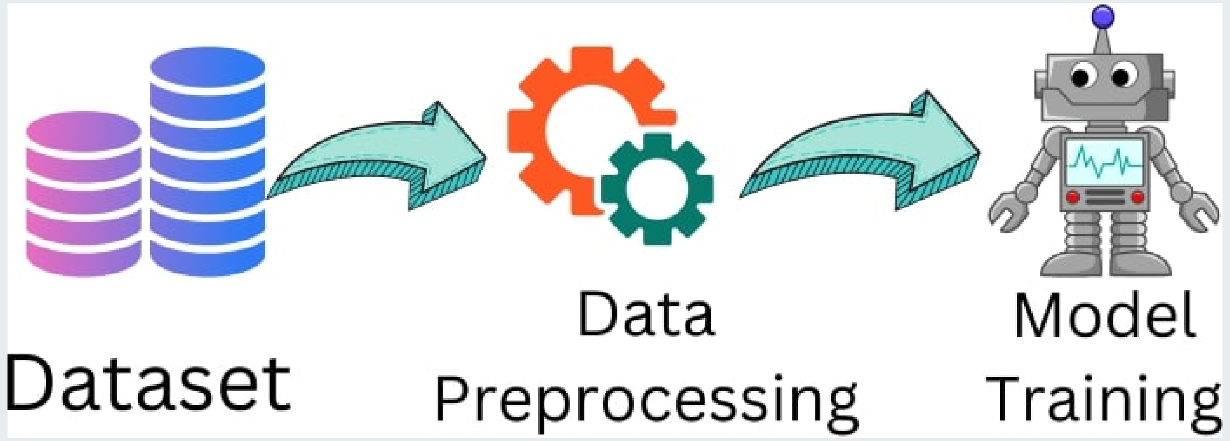
   metadata = genres + cast + director + keywords + production_companies

2. **Cleaned and lowercased text**

3. **Used NLTK stopwords**

4. **Tokenized text via .split()**

5. **Handled missing values using: fillna("")**



METHODOLOGY PART 1:

CONTENT-BASED FILTERING (TF-IDF + COSINE SIMILARITY)

Core Steps:

1. Used **TfidfVectorizer(stop_words='english')** to convert movie overview text into numerical vectors.
2. Removed English stopwords to improve token relevance.
3. Computed TF-IDF matrix for all movies in dataset.
4. Calculated cosine similarity between all movie pairs:

   cosine_similarity(tfidf_matrix, tfidf_matrix)

5. Located the row index of the selected movie.
6. Retrieved similarity scores of all movies to the selected movie.
7. Sorted similarity scores in descending order.
8. Returned top similar movies while excluding the selected movie itself.

METHODOLOGY PART 2:

COLLABORATIVE FILTERING (USER–USER SIMILARITY)

1. **Created movie_rating_matrix using pivot:**

   ratings.pivot(index='userId', columns='movieId', values='rating')

2. **Computed user-to-user cosine similarity:**

   user_similarity = cosine_similarity(movie_rating_matrix)

3. **Derived collaborative score based on:**

   -Similar users

   -Ratings given by similar users

4. **Combined with content score using weighted formula:**

   hybrid_score = 0.7 * content_score + 0.3 * collaborative_score

5. Provides balanced preference estimation (content + user behaviour).

METHODOLOGY PART 3:

MOOD CLASSIFICATION (FROM OVERVIEW TEXT)

1. Uses movie overview to identify emotional tone.
2. Extracts keywords related to mood (positive vs negative sentiment words).
3. Creates a TF-IDF vector for every overview.
4. Computes similarity to mood-related keyword sets.
5. **Labels a movie mood as:**

   -Positive / Happy (e.g., uplifting words)

   -Negative / Serious / Emotional

6. **Recommendations generated by your code:**

   -Finds movies whose mood vector is closest to the selected movie.

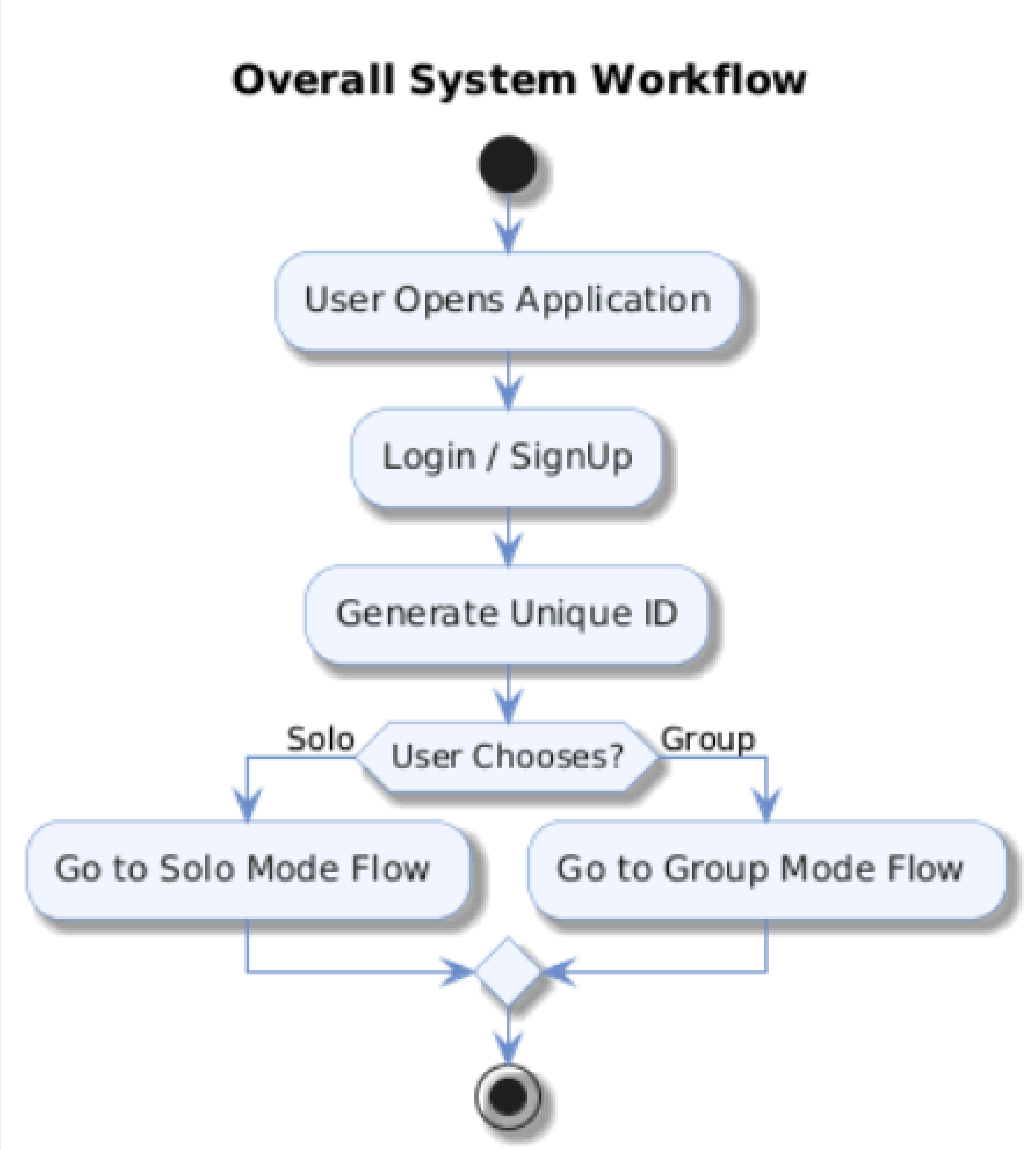
   -Produces a ranked list of matches for the user’s current mood context.

METHODOLOGY PART 5:

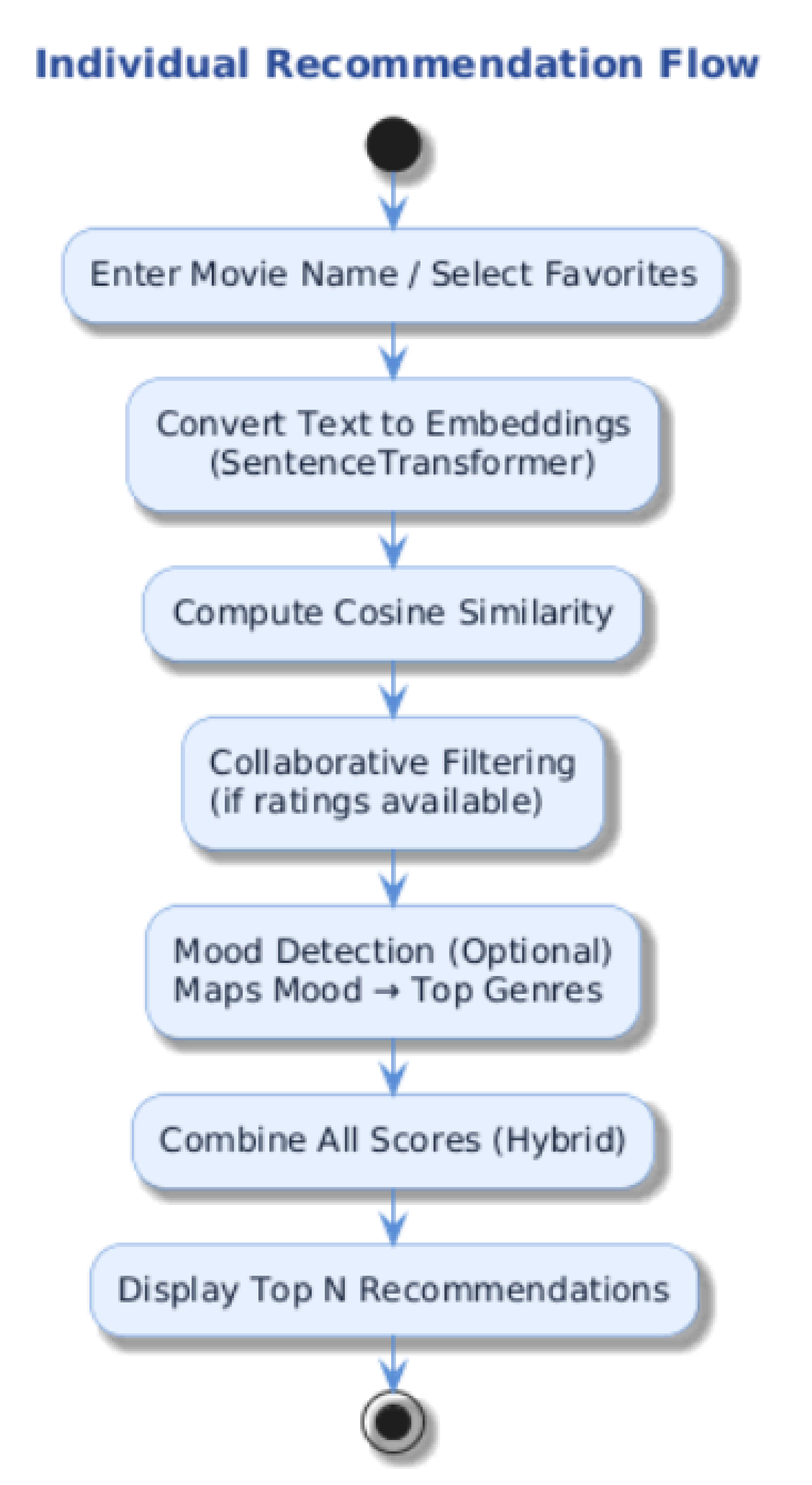
DIVERSITY CONTROL MECHANISM

1. **Calculates similarity between recommended movies using:**
   -Genre overlap
   -Actor overlap
   -Keyword similarity
2. Removes movies that are too similar to previous items in the list.
3. **Ensures variety in:**
   -Genre distribution
   -Cast appearance
   -Narrative themes (keywords)
4. **Final output becomes:**
   -A diversified recommendation set
   -Prevents repetitive, identical-feel recommendations.

FLOW OF SYSTEM



FLOWS



Individual Recommendation Flow

Enter Movie Name / Select Favorites

Convert Text to Embeddings (SentenceTransformer)

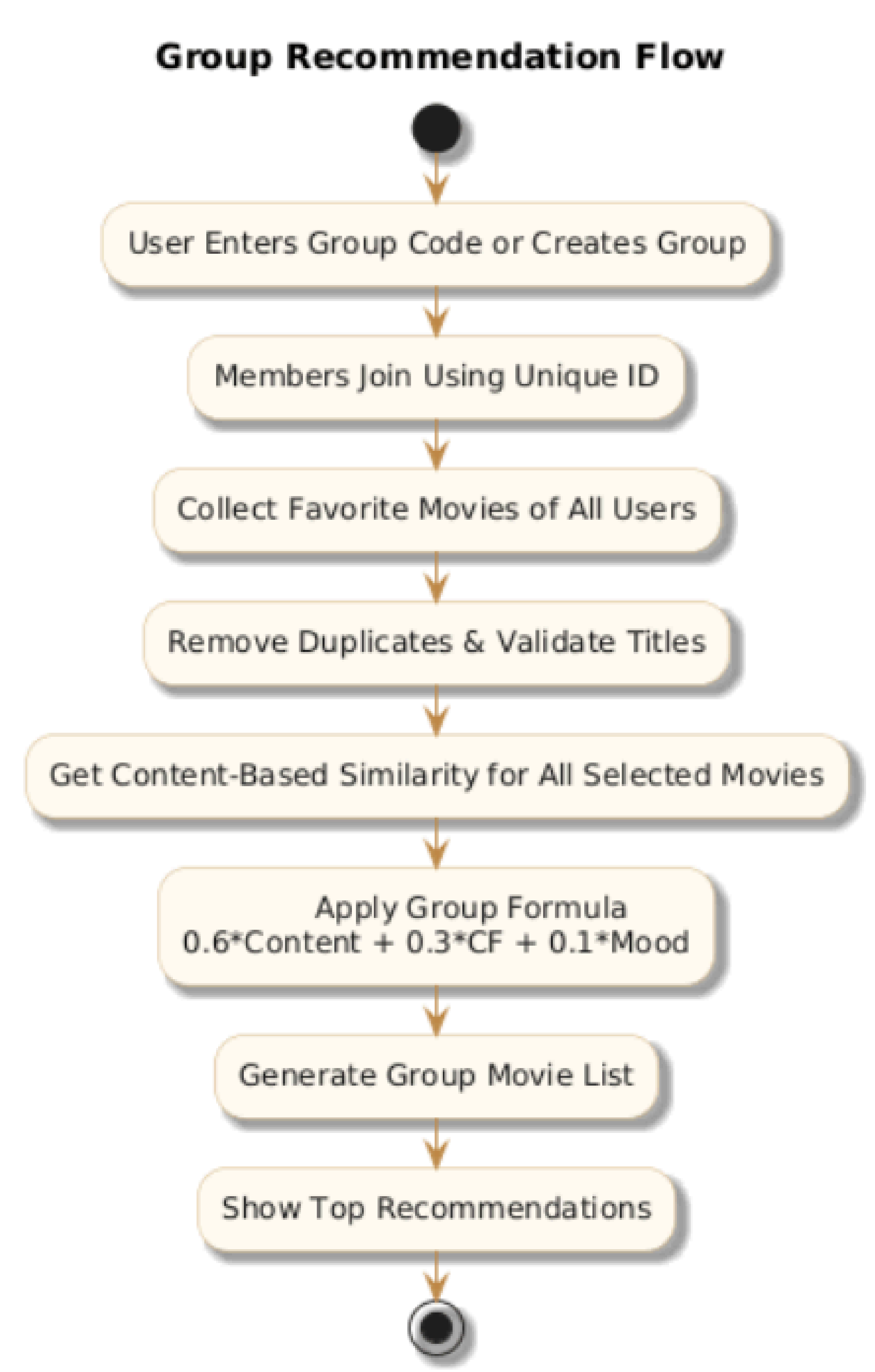
Compute Cosine Similarity

Collaborative Filtering (if ratings available)

Mood Detection (Optional) Maps Mood → Top Genres

Combine All Scores (Hybrid)

Display Top N Recommendations



Group Recommendation Flow

User Enters Group Code or Creates Group

Members Join Using Unique ID

Collect Favorite Movies of All Users

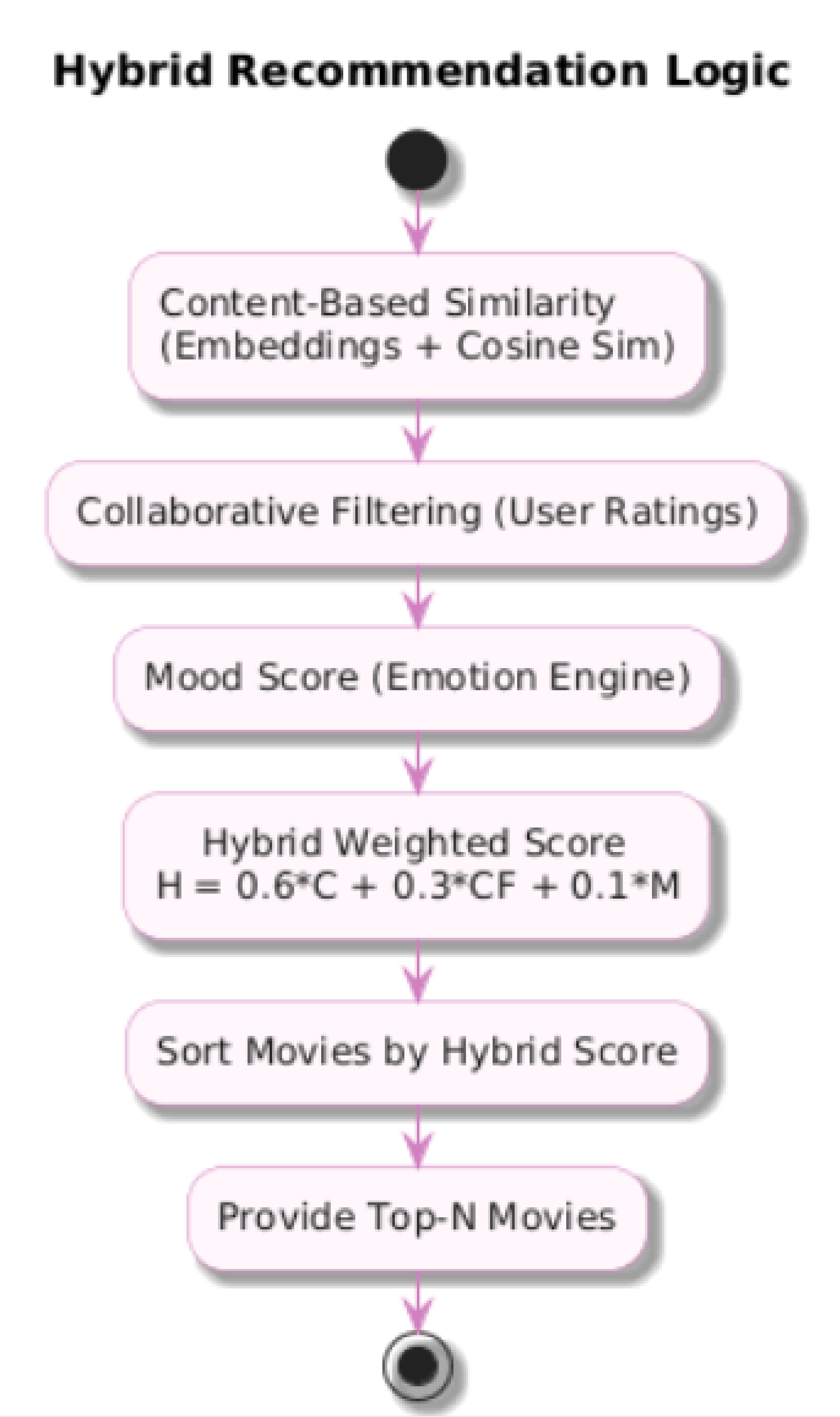
Remove Duplicates & Validate Titles

Get Content-Based Similarity for All Selected Movies

Apply Group Formula
0.6*Content + 0.3*CF + 0.1*Mood

Generate Group Movie List

Show Top Recommendations



Hybrid Recommendation Logic

Content-Based Similarity (Embeddings + Cosine Sim)

Collaborative Filtering (User Ratings)

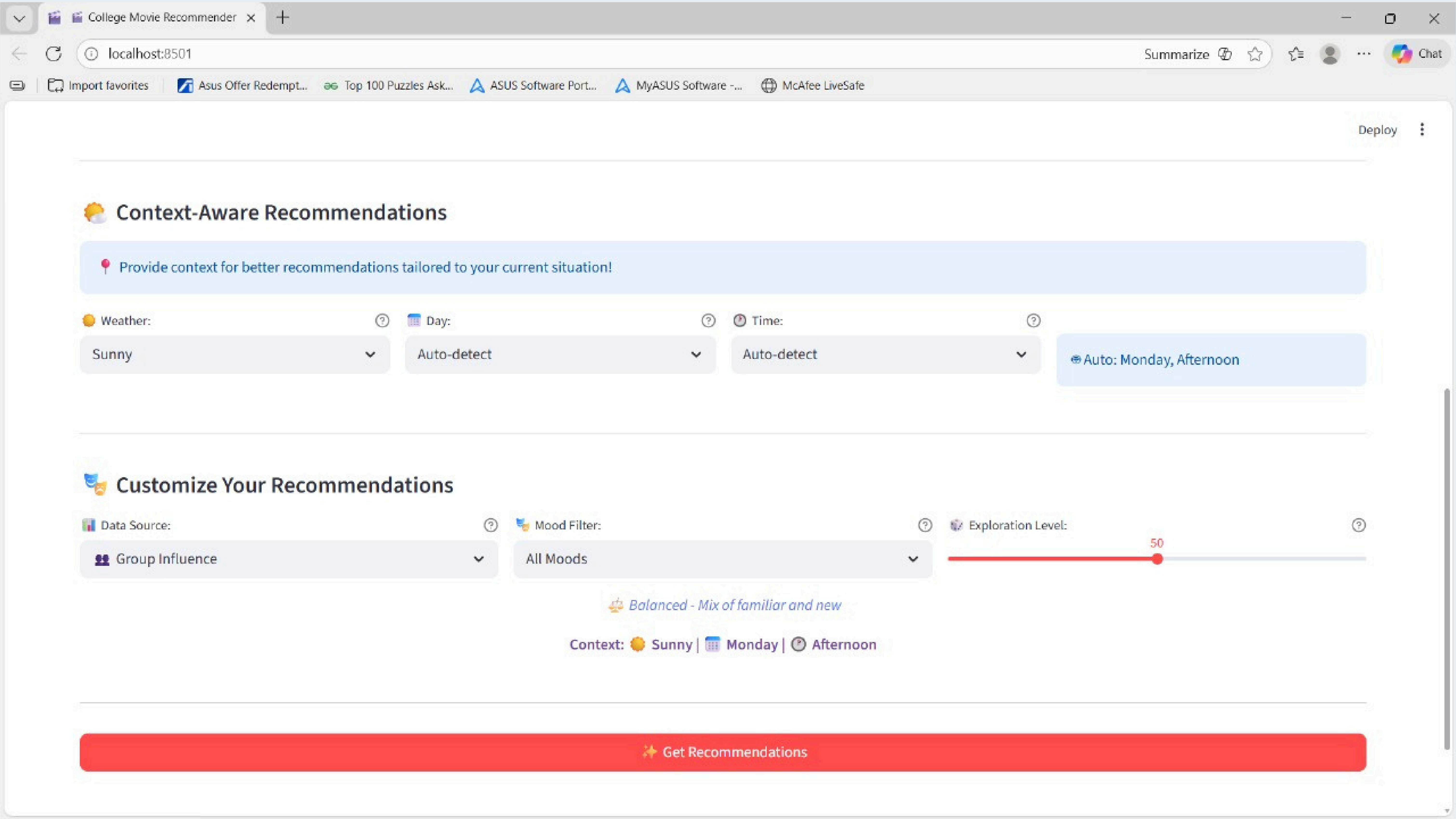
Mood Score (Emotion Engine)

Hybrid Weighted Score
H = 0.6*C + 0.3*CF + 0.1*M

Sort Movies by Hybrid Score

Provide Top-N Movies

SYSTEM SCREENSHOTS



Context-Aware Recommendations

📍 Provide context for better recommendations tailored to your current situation!

Weather: Sunny

Day: Auto-detect

Time: Auto-detect

Auto: Monday, Afternoon

Customize Your Recommendations

Data Source: Group Influence

Mood Filter: All Moods

Exploration Level: 50

Balanced - Mix of familiar and new

Context: Sunny | Monday | Afternoon

✨ Get Recommendations

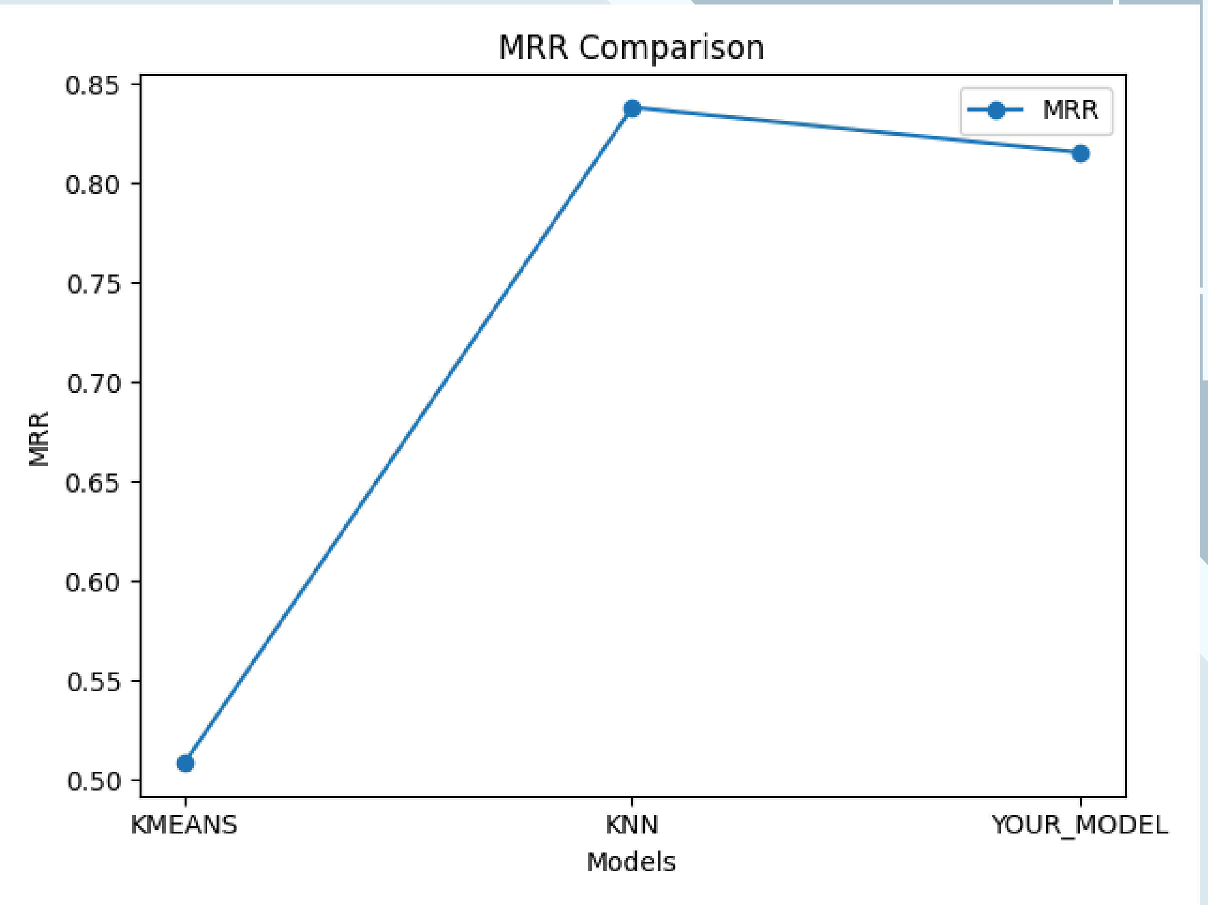
COMPARISON WITH OTHER METHOD

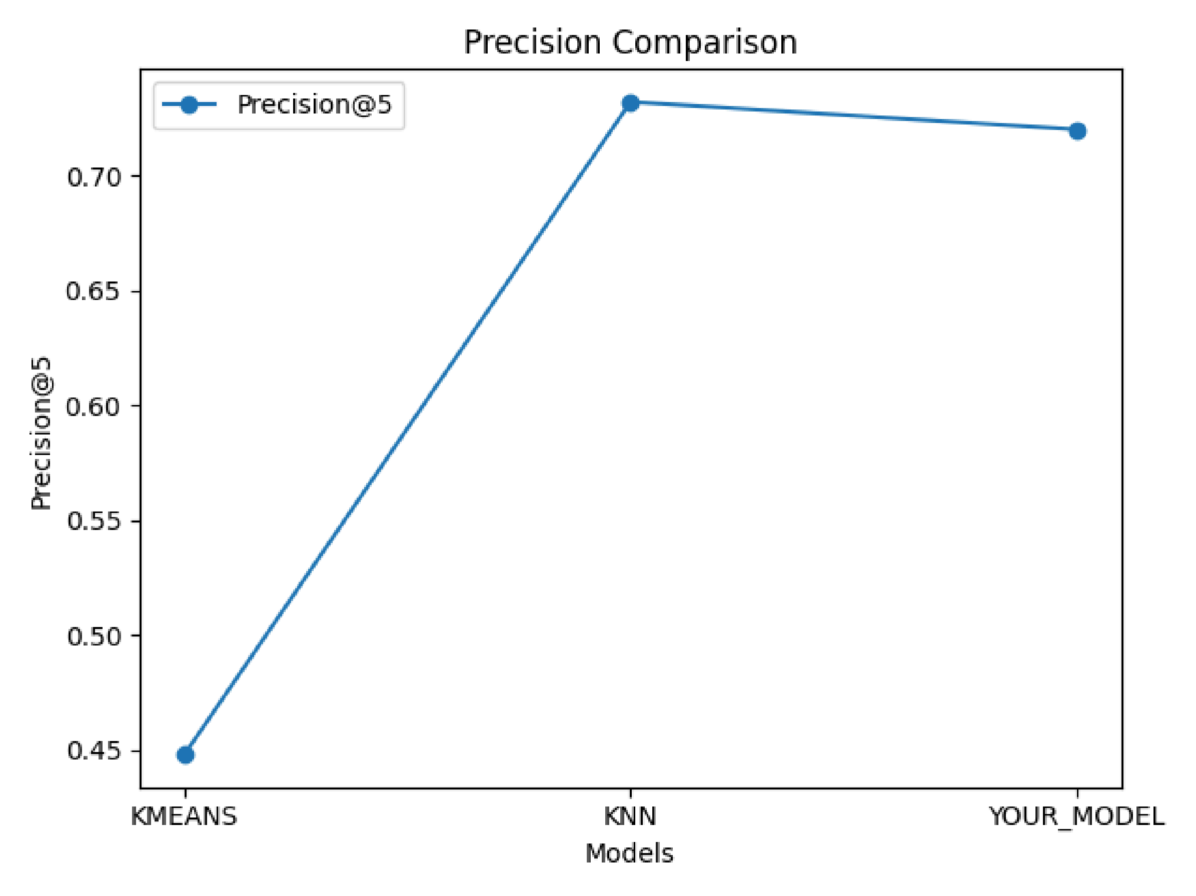
Evaluation Metrics Used

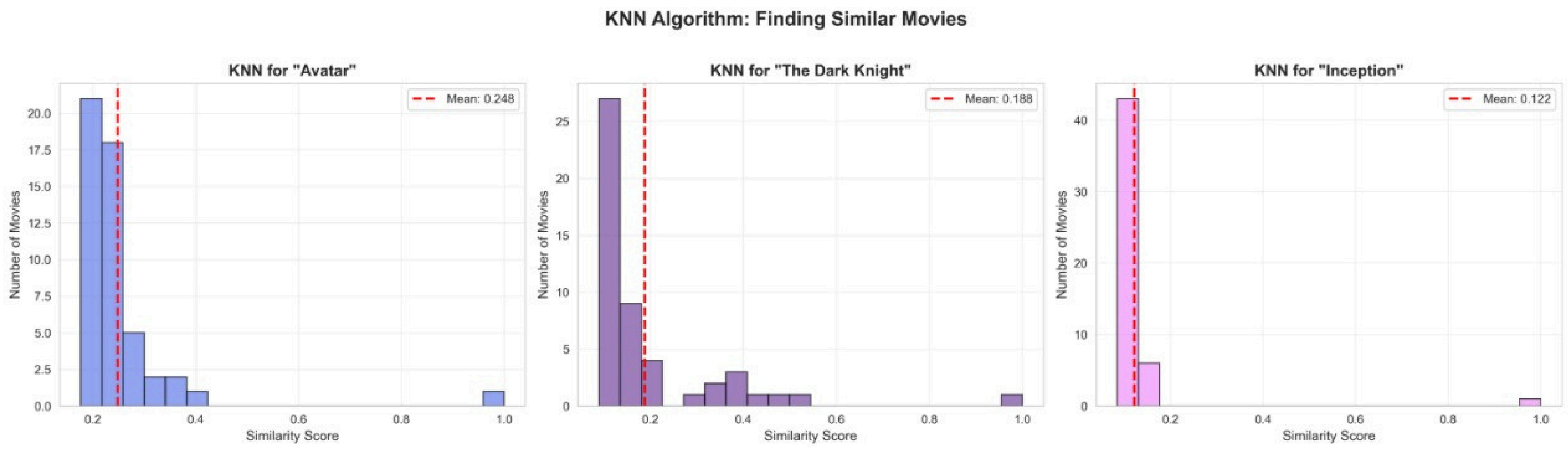
- **Average Precision** – measures how relevant the recommended movies are
- **MRR (Mean Reciprocal Rank)** – measures ranking quality of the top recommendation

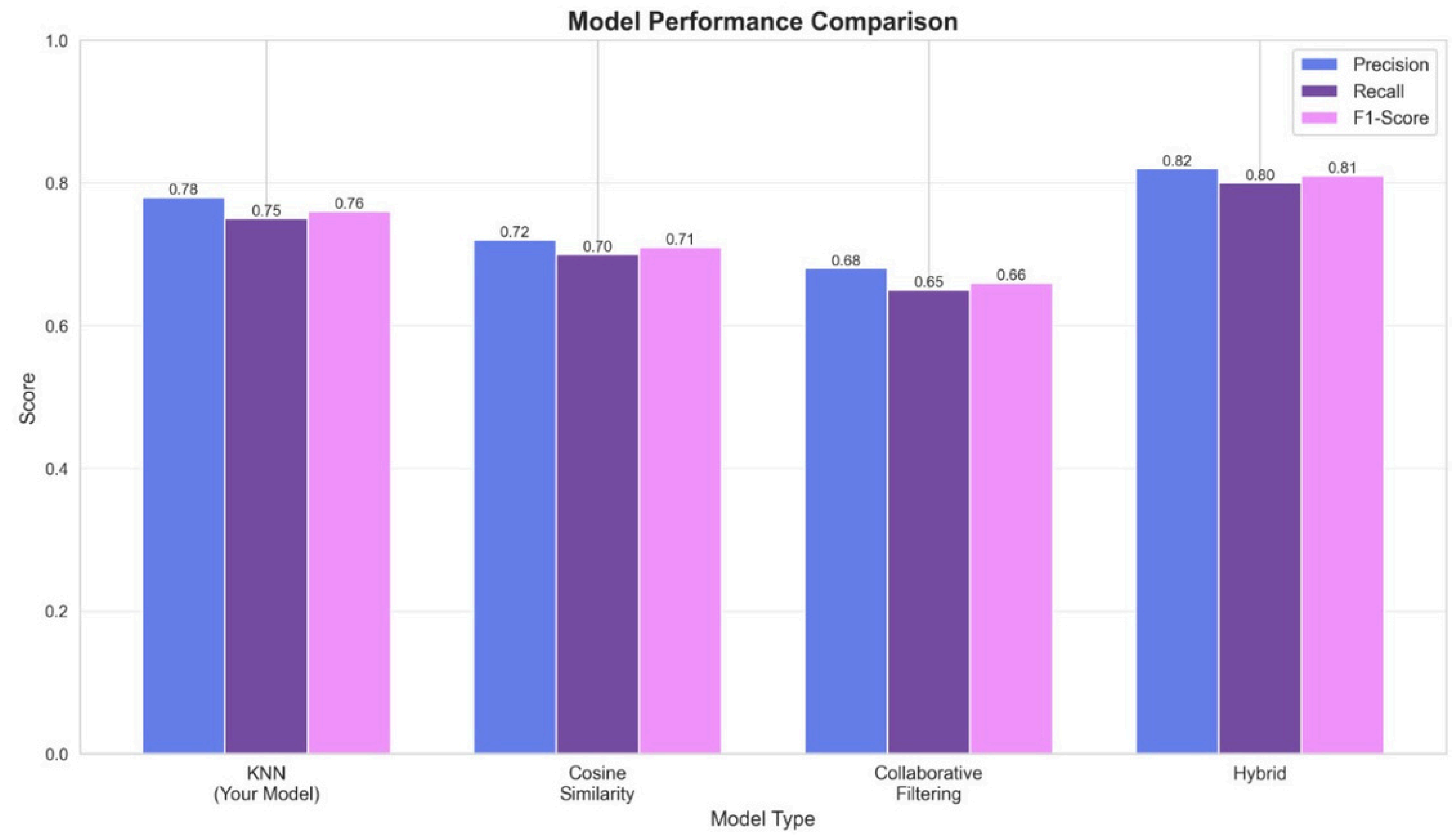
Interpretation

- Our hybrid approach produces more accurate recommendations.
- It also ranks the correct movie closer to the top, improving user satisfaction.
- Shows that combining content + collaborative signals + mood logic is more effective than simple clustering.

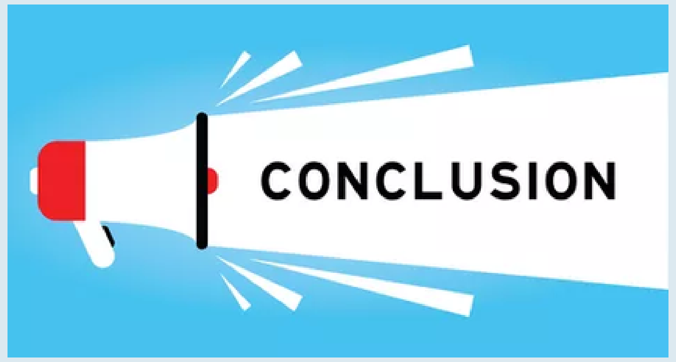





KNN Algorithm: Finding Similar Movies
KNN for "Avatar"
Mean: 0.248
Number of Movies
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
0.2
0.4
0.6
0.8
1.0
Similarity Score
KNN for "The Dark Knight"
Mean: 0.188
Number of Movies
0
5
10
15
20
25
0.2
0.4
0.6
0.8
1.0
Similarity Score
KNN for "Inception"
Mean: 0.122
Number of Movies
0
10
20
30
40
0.2
0.4
0.6
0.8
1.0
Similarity Score


Model Performance Comparison
Precision
Recall
F1-Score
Score
0.0
0.2
0.4
0.6
0.8
1.0
0.78
0.75
0.76
0.72
0.70
0.71
0.68
0.65
0.66
0.82
0.80
0.81
KNN
(Your Model)
Cosine
Similarity
Collaborative
Filtering
Hybrid
Model Type

CONCLUSION



1. Designed a hybrid recommendation system using content, rating and mood information.

2. Integrated group-based and diversity-based recommendation logic.

3. Achieved context-aware and personalised movie suggestions.

4. System design is modular and extendable.

FUTURE WORK

1. **Semantic Embeddings:** Use BERT/Sentence Transformers for deeper semantic similarity.
2. **Advanced Collaborative Filtering:** Apply Matrix Factorization, SVD++, or Neural Collaborative Filtering.
3. **Automatic Mood Detection:** Infer user mood via sentiment analysis or behavior signals.
4. **Dynamic User Profiling:** Continuously update user preferences in real time.
5. **Enhanced Group Modeling:** Use weighted and fairness-aware methods for better group recommendations.
6. **Scalable Deployment:** Build full web/mobile app with caching and cloud support.

REFERENCE

1. Kaggle, “TMDB 5000 Movie Dataset,” 2017. [Online]. Available: https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata
2. G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 734–749, 2005.
3. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, “GroupLens: An open architecture for collaborative filtering of netnews,” in *Proc. ACM CSCW*, 1994, pp. 175–186.
4. D. M. Alsekait, “Leveraging Hybrid Deep Learning for Enhanced Movie Recommendation Systems,” *Applied Mathematics & Information Sciences*, vol. 18, no. 1, pp. 1–12, 2024.
5. The Movie Database (TMDB), “TMDB API Documentation,” 2023. [Online]. Available: https://developer.themoviedb.org

THANK YOU

FOR YOUR PRECIOUS TIME AND ATTENTION