Project Part 3:
Building a Mood-Based Movie Recommender Chatbot

Topics in DBI: Business Analytics and AI Project

Chuu

22229518

Problem Statement

Industry: Online Streaming & Digital Entertainment (e.g., Netflix, Amazon Prime, Hulu)

1. Paradox of choice leads to decision fatigue.
2. Static recommendations (User-User/Item-Item) fail to account for a user's immediate context (Intent/mood).
3. Availability gaps (Region, Platform) fail to offer the seamless experience.

Solution

Mood-Based Movie Recommender Chatbot

Why it’s important

New data collected (Natural language interaction logs)	Innovative Competitive edge & Seamless experience	Recommendation precision (User/Item/Mood)

Dataset Summary

```
--- Loading and Cleaning Data ---
Original data shape (with fewer columns): (45466, 9)
Cleaned data shape: (42277, 10)
<class 'pandas.core.frame.DataFrame'>
Index: 42277 entries, 0 to 45465
Data columns (total 10 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             42277 non-null  int64
 1   title          42277 non-null  object
 2   overview       42277 non-null  object
 3   popularity     42277 non-null  float64
 4   vote_average   42277 non-null  float64
 5   vote_count     42277 non-null  float64
 6   runtime        40545 non-null  float64
 7   revenue        6944 non-null   float64
 8   primary_genre  42277 non-null  object
 9   genre_list     42277 non-null  object
dtypes: float64(5), int64(1), object(4)
```

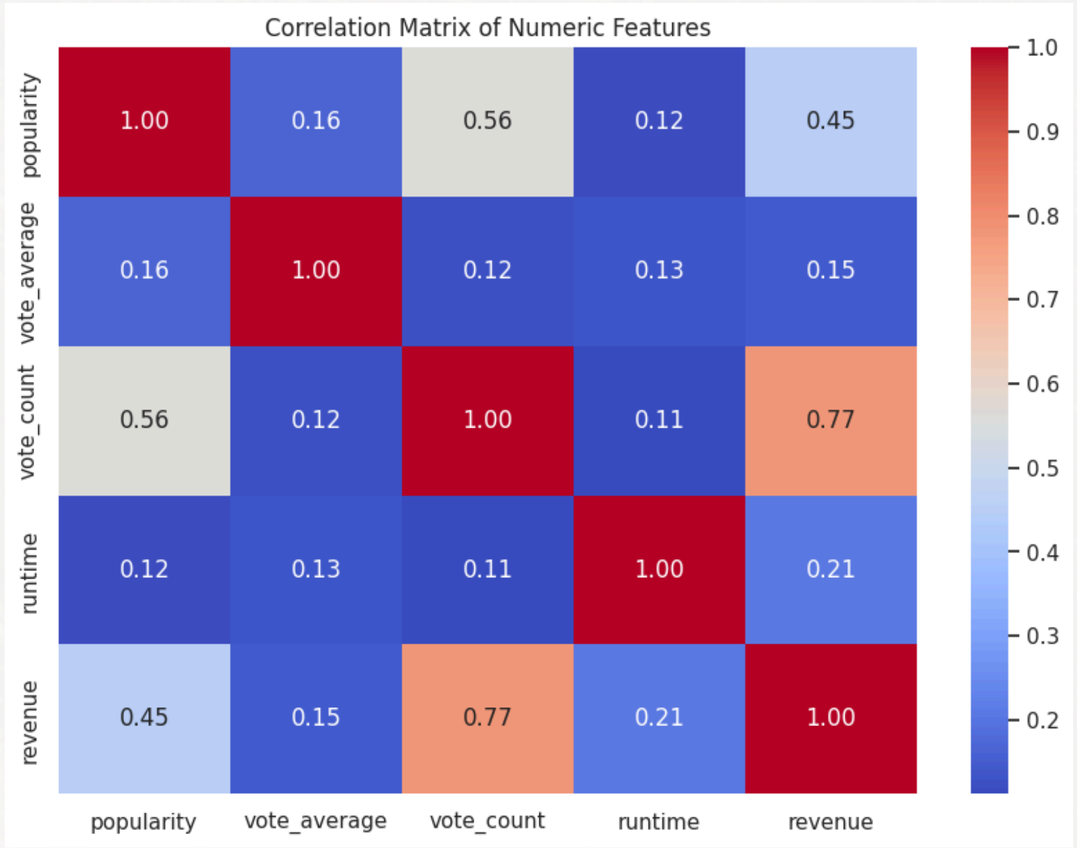
Initial Data Structure:

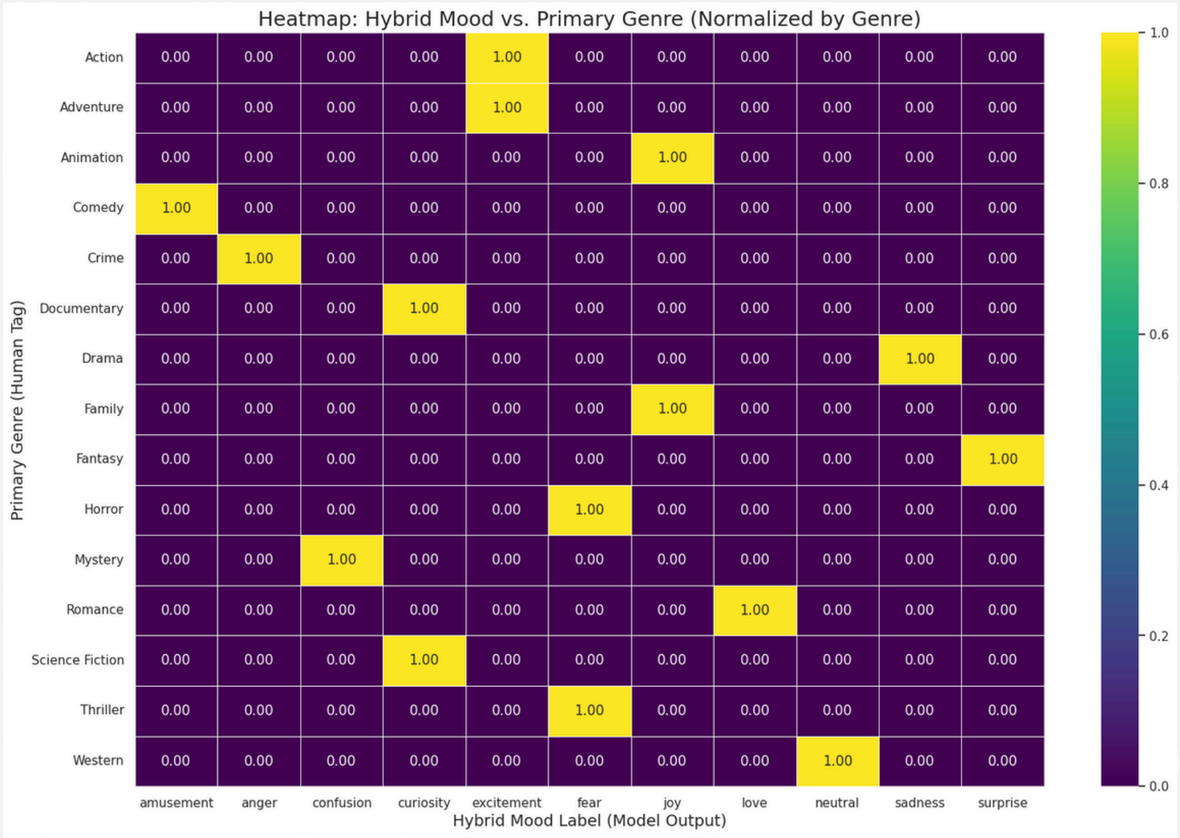
- Size: 45,000+ movies.
- Key Columns: title, genres (a JSON string), overview (raw text), vote_average (number), vote_count(number).

```
--- Summary Statistics (Numeric Columns) ---
         popularity  vote_average    vote_count       runtime       revenue
count  42277.000000  42277.000000  42277.000000  40545.000000  6.944000e+03
mean       2.908844      5.624940    109.183670     97.226119  6.785574e+07
std        5.740560      1.930599    488.895804     33.826171  1.444588e+08
min        0.000000      0.000000      0.000000      1.000000  1.000000e+00
25%        0.384436      5.000000      3.000000     86.000000  2.400000e+06
50%        1.133439      6.000000     10.000000     95.000000  1.657789e+07
75%        3.705199      6.800000     34.000000    107.000000  6.600050e+07
max      547.488298     10.000000  14075.000000   1256.000000  2.787965e+09
```

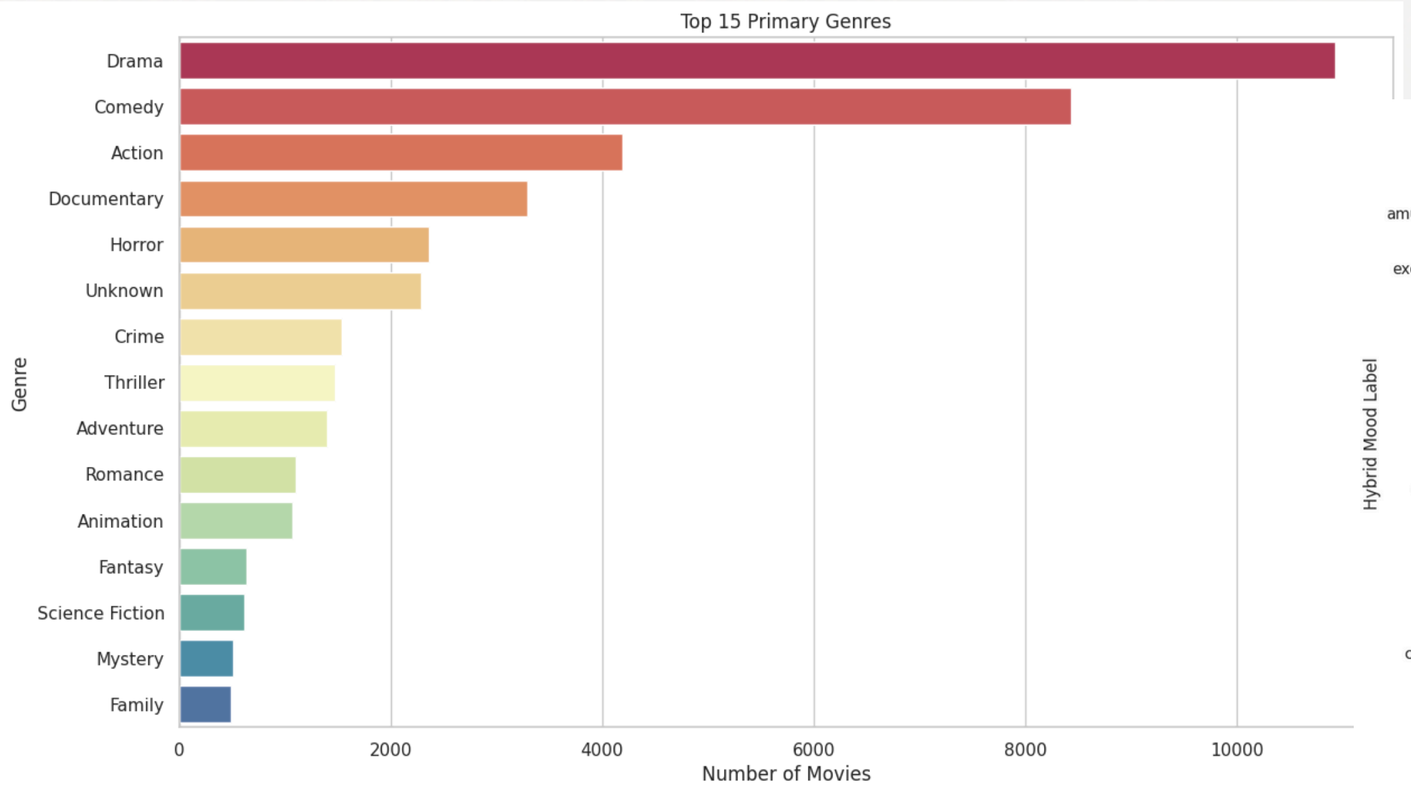
GOOGLE COLAB

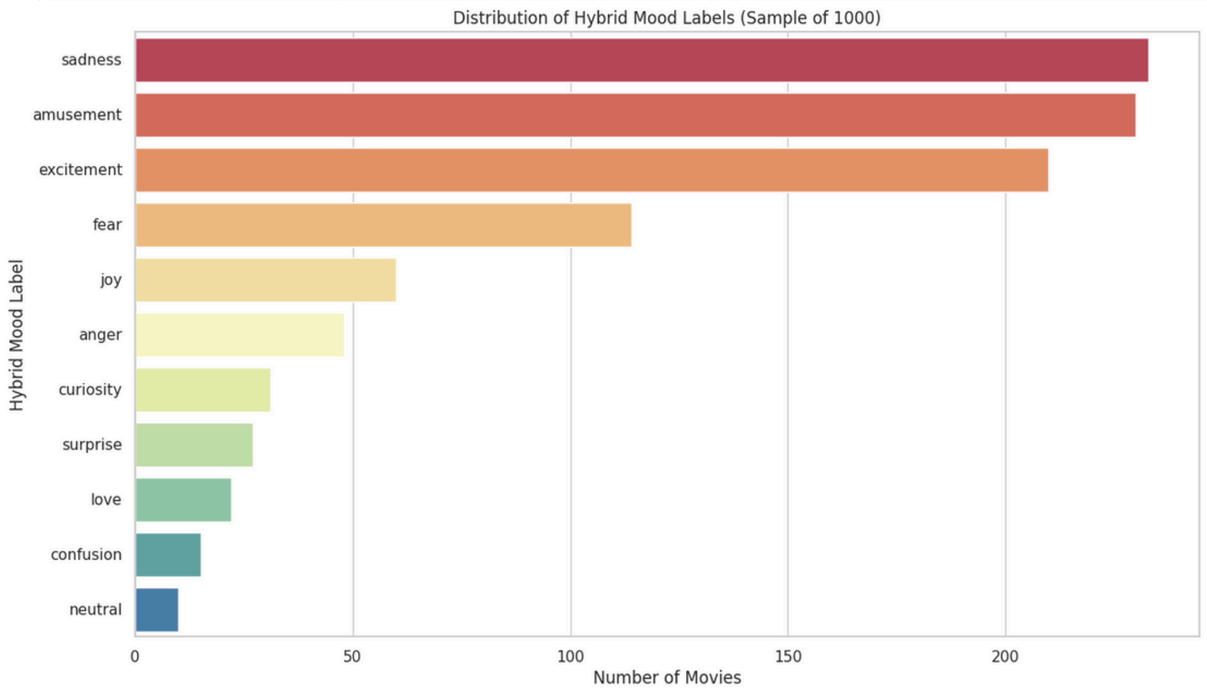
Explanatory Data Analysis





Explanatory Data Analysis





Attribute choices, Model choices, KPIs

Attributes used	Model 1	Output
vote_count vote_average	Popularity filter with weighted score	high-quality, universally acclaimed movies

$$\text{Weighted Score} = \frac{v}{v+m}R + \frac{m}{v+m}C$$

where:

- R = average rating of the movie
- v = number of votes
- C = mean rating across all movies
- m = minimum votes required to be considered

```
--- Building Recommender 1: Popularity Filter ---
Popularity filter 'm' (80th percentile) = 50 votes

--- Top 10 'Popularity' Recommendations ---
                          title  quality_score primary_genre
10309  Dilwale Dulhania Le Jayenge       8.855622        Comedy
314       The Shawshank Redemption       8.482903         Drama
834                  The Godfather       8.476333         Drama
40251                  Your Name.       8.366895       Romance
12481             The Dark Knight       8.289143         Drama
2843                   Fight Club       8.286251         Drama
292                   Pulp Fiction       8.284661      Thriller
522               Schindler's List       8.270184         Drama
23673                    Whiplash       8.269780         Drama
5481                Spirited Away       8.266712       Fantasy
```

Attributes used	Model 2	Output
overview **User input** Movie title	Content-Based (TF-IDF + cosine similarity)	Movies with similar plot descriptions

$$\text{Cosine Similarity} = \frac{A \cdot B}{||A||\,||B||}$$

```
--- Building Recommender 2: Content-Based Filter ---
Original movie count: 42277. New high-quality count for R2/R3: 8548
Preprocessing overviews for high-quality subset...
Fitting TF-IDF matrix (on smaller dataset)...
Calculating cosine similarity matrix...

--- Top 10 'Content-Based' Recommendations for 'The Dark Knight' ---
                                      title primary_genre  \
5750                   The Dark Knight Rises        Action
710                          Batman Returns        Action
5306               Batman: Under the Red Hood        Action
6160  Batman: The Dark Knight Returns, Part 2        Action
80                           Batman Forever        Action
3830                          Batman Begins        Action
8146                Batman: The Killing Joke        Action
6094  Batman: The Dark Knight Returns, Part 1        Action
7057                        Batman vs Dracula       Fantasy
5697                         Batman: Year One        Action

                                        overview
5750  Following the death of District Attorney Harve...
710   Having defeated the Joker, Batman now faces th...
5306  Batman faces his ultimate challenge as the mys...
6160  Batman has stopped the reign of terror that Th...
80    The Dark Knight of Gotham City confronts a das...
3830  Driven by tragedy, billionaire Bruce Wayne ded...
8146  As Batman hunts for the escaped Joker, the Clo...
6094  Batman has not been seen for ten years. A new ...
7057  Gotham City is terrorized not only by recent e...
5697  Two men come to Gotham City: Bruce Wayne after...
```

Attribute choices, Model choices, KPIs

Attributes used
overview, genre
User input
Mood

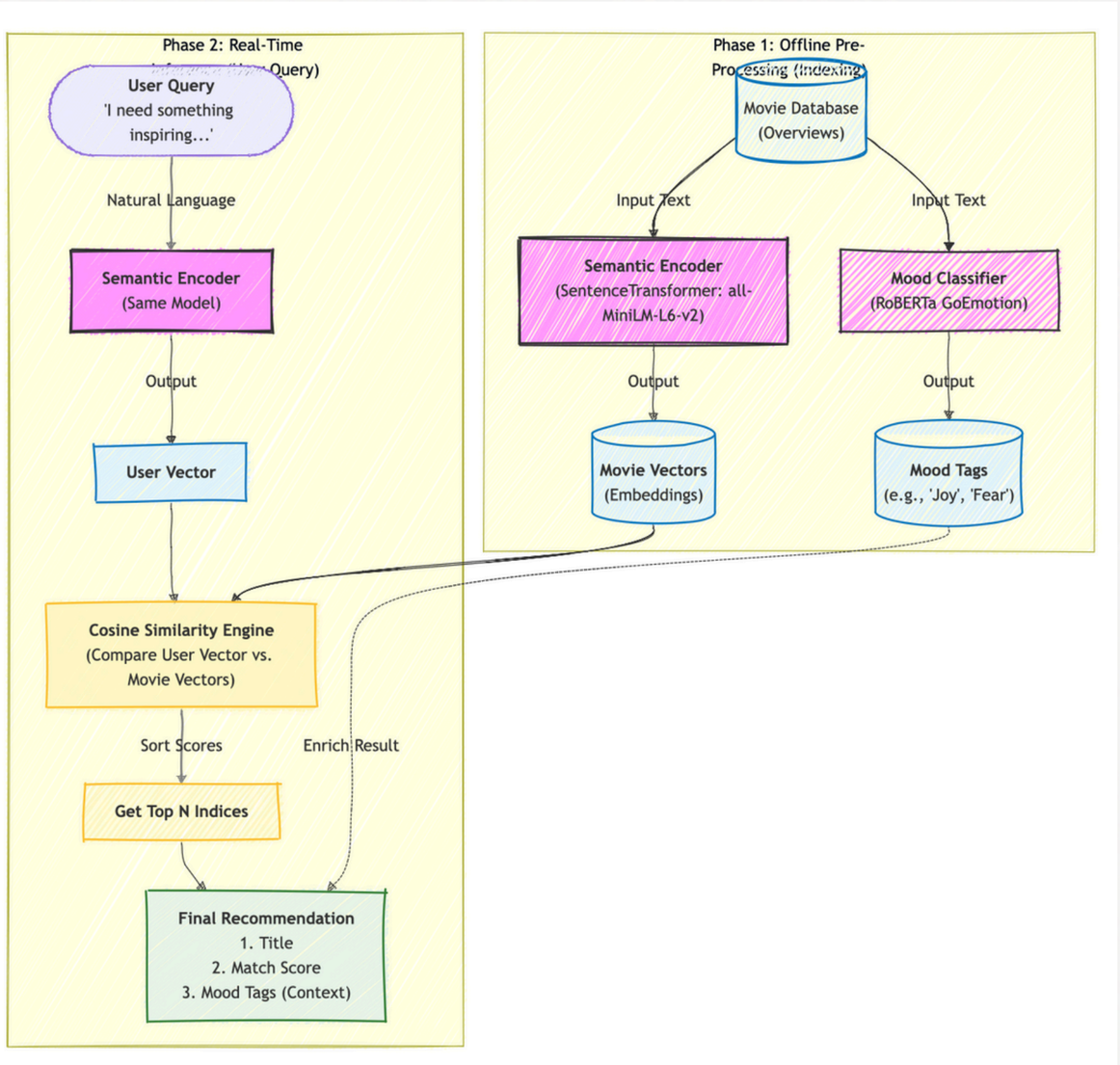
Model 3
mood labeling with go_emotion + weighted score

Output
Mood-based recommendations

```
--- R3: Mood-Based (Input: 'excitement') ---
                              title hybrid_mood primary_genre  quality_score
6646                    Interstellar  excitement     Adventure       8.088987
318       Terminator 2: Judgment Day  excitement        Action       7.676005
7175  Guardians of the Galaxy Vol. 2  excitement        Action       7.579879
1383                 The Iron Giant  excitement     Adventure       7.535031
319               Dances with Wolves  excitement     Adventure       7.512916
```

```python
print("\n--- Building Recommender 3: Hybrid Mood-Based Filter ---")

genre_emotion_map = {
    'Comedy': 'amusement', 'Horror': 'fear', 'Thriller': 'fear',
    'Mystery': 'confusion', 'Action': 'excitement', 'Adventure': 'excitement',
    'Fantasy': 'surprise', 'Science Fiction': 'curiosity', 'Romance': 'love',
    'Drama': 'sadness', 'Family': 'joy', 'Animation': 'joy',
    'Crime': 'anger', 'Documentary': 'curiosity', 'History': 'neutral',
    'War': 'sadness', 'Music': 'joy', 'Western': 'neutral'
}

# Load the GoEmotion pipeline
emotion_pipeline = pipeline("text-classification",
                            model="bsingh/roberta_goEmotion",
                            top_k= None,
                            truncation=True,
                            max_length=512,
                            device=device)

SAMPLE_SIZE = 1000
if SAMPLE_SIZE > len(df_rec):
    SAMPLE_SIZE = len(df_rec)

print(f"Running GoEmotion on a {SAMPLE_SIZE} sample from the high-quality dataset...")
df_sample = df_rec.sample(n=SAMPLE_SIZE, random_state=42).copy()

def get_goemotion_label(text):
    if not text: return 'neutral'
    try:
        truncated_text = " ".join(text.split()[:250])
        results = emotion_pipeline(truncated_text)
        return results[0]['label']
    except Exception: return 'unknown'

df_sample['go_emotion'] = df_sample['overview'].apply(get_goemotion_label)
print("GoEmotion analysis complete.")


def get_hybrid_mood(row):
    genre_mood = genre_emotion_map.get(row['primary_genre'], 'neutral')
    overview_mood = row['go_emotion'],

    if overview_mood in ['neutral', 'unknown']: return genre_mood
    if row['primary_genre'] == 'Drama' and overview_mood != 'sadness': return overview_mood
    if overview_mood == genre_mood: return overview_mood
    return genre_mood

df_sample['hybrid_mood'] = df_sample.apply(get_hybrid_mood, axis=1)
print("Hybrid mood labels created.")

def get_mood_recommendations(mood, data=df_sample, top_n=10):
    if mood not in data['hybrid_mood'].unique():
        return f"Error: Mood '{mood}' not found. Try one of: {data['hybrid_mood'].unique()}"

    mood_movies = data[data['hybrid_mood'] == mood]

    return mood_movies.sort_values('quality_score', ascending=False)[['title', 'hybrid_mood',

print("\n--- Top 10 'amusement' (Mood) Recommendations ---")
print(get_mood_recommendations('amusement'))
```

Attribute choices, Model choices, KPIs



Attributes used
overview, genre
User input
conversation logs

Model4
semantic encoding, mood labeling, cosine similarity

Output
Context-based recommendations

```
User Query: 'I want a complex story about love that ends tragically and makes me cry'
--- Top 5 Semantic Matches ---
Movie: Big Bully (Score: 0.4283)
   Generated Tags: [sadness, neutral, disappointment]
   Genre: Comedy
------------------------------
Movie: Dead Man Walking (Score: 0.3588)
   Generated Tags: [neutral, caring, admiration]
   Genre: Drama
------------------------------
Movie: Bed of Roses (Score: 0.3504)
   Generated Tags: [love, admiration, neutral]
   Genre: Drama
------------------------------
Movie: Restoration (Score: 0.3310)
   Generated Tags: [neutral, admiration, love]
   Genre: Drama
------------------------------
Movie: Eye for an Eye (Score: 0.3235)
   Generated Tags: [neutral, approval, annoyance]
   Genre: Drama
------------------------------
```

```
User Query: 'Space adventure with aliens and futuristic battles
--- Top 5 Semantic Matches ---
Movie: Mortal Kombat (Score: 0.3406)
   Generated Tags: [neutral, approval, optimism]
   Genre: Action
------------------------------
Movie: Screamers (Score: 0.2276)
   Generated Tags: [neutral, approval, caring]
   Genre: Horror
------------------------------
Movie: Shopping (Score: 0.2134)
   Generated Tags: [admiration, neutral, love]
   Genre: Action
------------------------------
Movie: Bio-Dome (Score: 0.2057)
   Generated Tags: [neutral, approval, annoyance]
   Genre: Comedy
------------------------------
Movie: Beautiful Girls (Score: 0.2027)
   Generated Tags: [neutral, amusement, approval]
   Genre: Comedy
------------------------------
```

```
User Query: 'I need something inspiring to lift my mood'
--- Top 5 Semantic Matches ---
Movie: Now and Then (Score: 0.2409)
   Generated Tags: [neutral, anger, annoyance]
   Genre: Comedy
------------------------------
Movie: Mr. Holland's Opus (Score: 0.2193)
   Generated Tags: [neutral, approval, caring]
   Genre: Music
------------------------------
Movie: Powder (Score: 0.2186)
   Generated Tags: [neutral, annoyance, anger]
   Genre: Drama
------------------------------
Movie: Beautiful Girls (Score: 0.2080)
   Generated Tags: [neutral, amusement, approval]
   Genre: Comedy
------------------------------
Movie: Bed of Roses (Score: 0.2069)
   Generated Tags: [love, admiration, neutral]
   Genre: Drama
------------------------------
```

Comparative Analysis

KPI: Relevance, Diversity, and User Satisfaction were chosen to measure if the emotional intent was met.

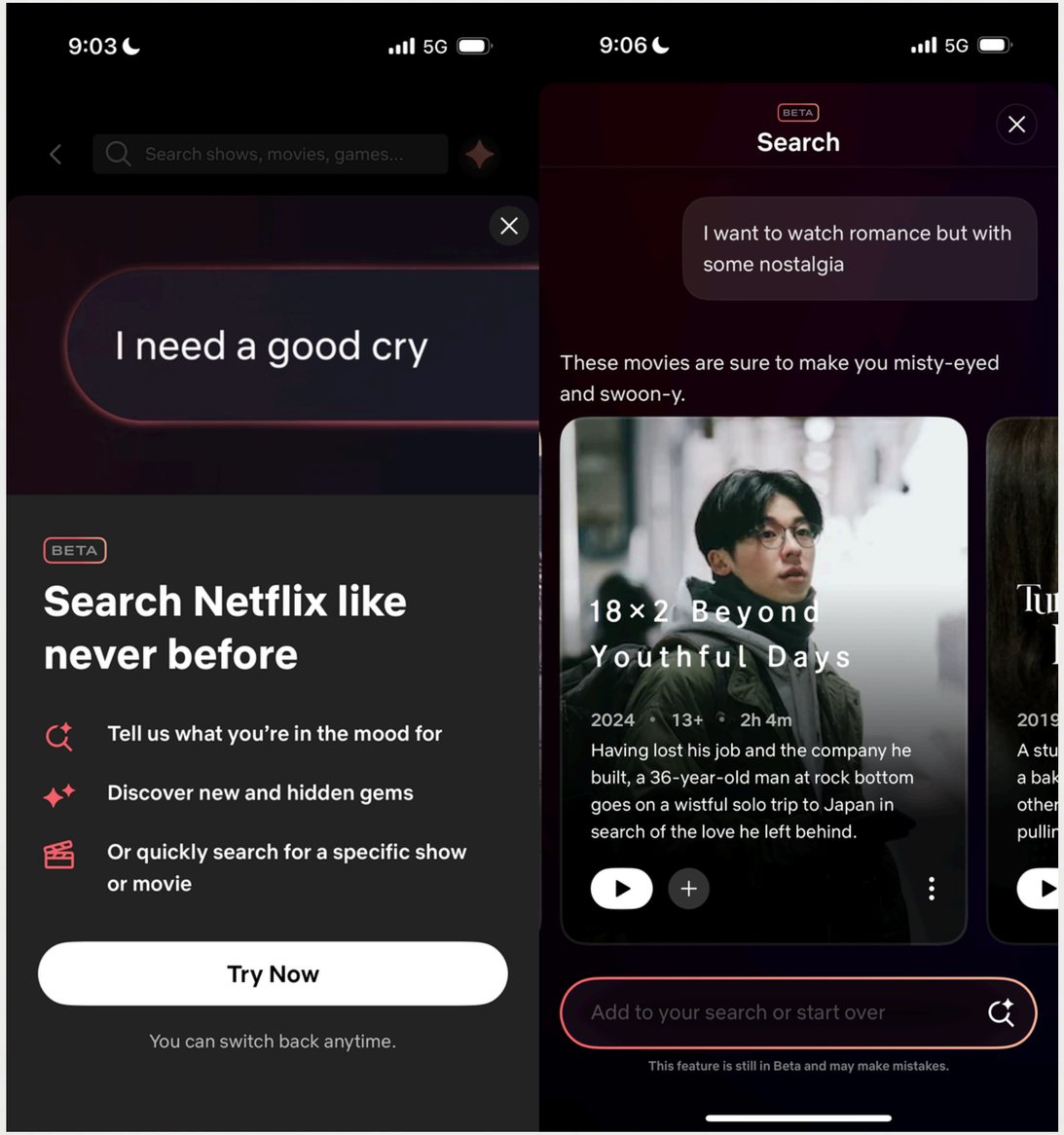
Feature	Model 1: Popularity	Model 2: Content (TF-IDF)	Model 3: Hybrid Mood Filter	Model 4: Context_based
Logic	Weighted Rating (Stats)	Keyword Matching	Genre Filter + Emotion Label	Deep Semantic Understanding
Best For	"What's good?" (Generic)	"I want more of *this* movie"	"I feel sad, I want joy"	"I want a specific complex vibe"
Pros	Safe, high quality	Finds similar plots	Direct emotional fix	Understands complex sentences
Cons	Impersonal	Fails if words don't match	Rigid (Must pick 1 mood)	Computationally heavier

Conclusion

If I have to choose....

Future work

- UI/UX
- Additional parameters(e.g run_time)
- User History Integration
- Real time feedback loop for reinforcement learning
- Integrate a Large Language Model



The End

THANK YOU FOR LISTENING