Project 2 CinemAI

Name of Team Members: Parker Nunley, Germano de Carvalho **Date:** 04/14/2025

Project Description

Choosing a movie to watch can be surprisingly difficult. With so many streaming services and thousands of options available, people often spend more time browsing than actually watching. This project aims to solve that problem by recommending movies that match a user's preferences, helping them make a decision quickly and confidently.

This kind of recommendation system is useful in many real-world scenarios for example, when a couple is trying to pick a movie for a night in. Instead of endlessly scrolling through lists or disagreeing on what to watch, they can get a few strong suggestions based on what they like.

From a technical perspective, this project connects directly to what we're learning in our Intro to AI class. It uses artificial intelligence to process a database of movies and make intelligent guesses about what a person might enjoy, based on their input. This shows how AI can help solve everyday problems using simple but powerful tools.

Program and Method Description

CinemAI is a hybrid AI movie recommendation system that combines a Windows Forms C# frontend, a Python-based AI backend, and a SQL Server database. The system helps users find movies they're likely to enjoy by analyzing either their past watch history or a natural language description of what they want to see.

Frontend & User Interaction

The user interacts with the system through a C# Windows Forms interface. This includes features like browsing available movies, logging in, booking tickets, and receiving personalized recommendations. The frontend gathers information from the user such as previously watched movies or a text input like "I want a romantic comedy" and sends that data to the backend through a Flask API.

Backend & AI Logic

The Python backend handles the recommendation logic using machine learning and natural language processing techniques. It has two main versions:

1. **Standard API (`app.py`)** This version uses a local database of movies connected to the

SQL Server system. It powers features like recommending movies based on a user's watch history or natural language input.

2. **Extended API (`app_large.py`)** This version connects to a much larger dataset of movies sourced from Kaggle's TMDB Movie Metadata dataset [Kaggle TMDB Dataset]. This allows for more diverse and accurate recommendations using thousands of movies, beyond what's in the user's immediate environment or watch history.

The AI backend uses the following methods:

- **TextBlob** for spell correction
- **Fuzzy NLP** for genre detection, even with typos or casual phrases
- **TF-IDF + Cosine Similarity** to measure how closely movie descriptions match the input
- **K-Nearest Neighbors (KNN)** to find similar movies
- **Hybrid Scoring**: A custom scoring system combining genre match (60%) and semantic similarity (40%)

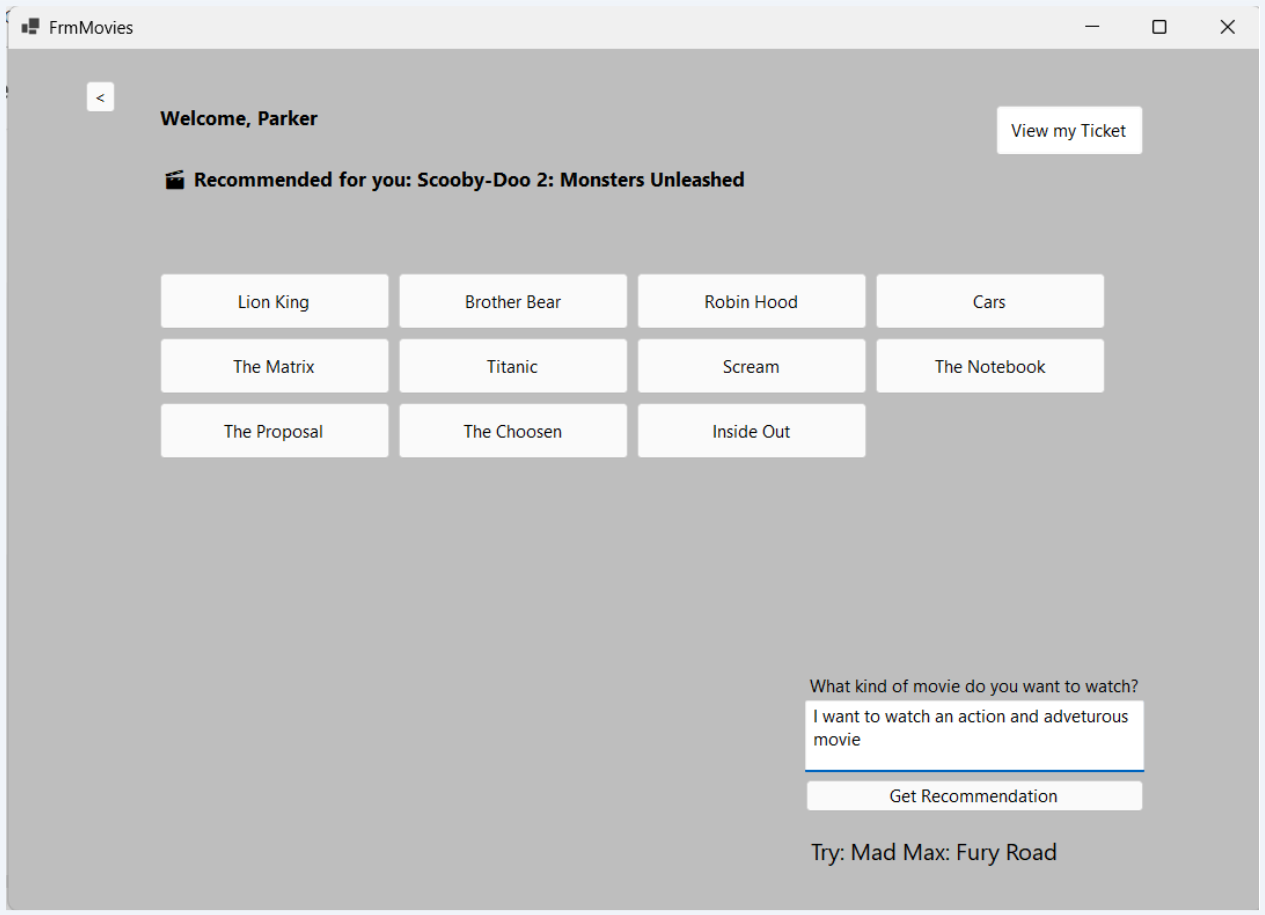
Database Integration

The SQL Server database stores users, movies, tickets, and showtimes. The backend uses this data to tailor suggestions. For example, when a user logs in, it automatically suggests a movie they haven't seen yet based on their viewing history.

Results

The CinemAI recommendation system performs very well in practical testing. It successfully interprets a wide range of natural language inputs and returns accurate, relevant movie recommendations. For example, entering something like *"I want to watch an animated movie about emotions"* returns *Inside Out*, while inputs like *"action adventure"* suggest films like *Mad Max*. Across a variety of test phrases, from genre specific queries to more conversational prompts, the system consistently produces results that make sense.

It also works smoothly with more general input like "horror" or "romantic comedy," often returning top-rated and well-known films that fit the request. This shows that the hybrid scoring model, which combines genre tagging with text similarity, is effectively tuned.



Example showing the C#+Windows Forms interacting with the large database

Some more runs from the text recommendation:

User Input	Recommended Movie
I want an animated movie with emotions	Inside Out
I feel like watching a romantic comedy	Midnight in Paris
Give me a scary horror movie	Diary of the Dead
Looking for a comedy with teenagers	16 to Life

While overall performance is strong, there may be occasional edge cases where very unusual or vague inputs return unexpected results. For example, a description that doesn't clearly map to a known genre or storyline might lead to a less accurate recommendation. However, in most

normal usage, the system delivers impressively reliable outputs. Also, if no reasonable results were found it just randomly recommends a movie, so if you text “I want to watch something”, it just gives you a random movie.

Conclusion

Overall, this project went really well. It was fun and satisfying to bring all the parts together, especially connecting the AI backend with a large database of over 5,000 movies. The implementation focused on being streamlined and readable, which made it faster to build and helped avoid unexpected bugs or issues.

One possible area for future improvement would be the user interface. Right now, it’s functional, but with some design upgrades, like adding animations, improved visuals, or more interaction options could feel more modern and engaging. Another potential next step could be integrating user ratings or feedback to help improve recommendations over time.

This project also helped reinforce how powerful AI techniques can be when combined. By applying natural language processing, fuzzy matching, TF-IDF vectorization, cosine similarity, and K-nearest neighbors, we were able to build a system that understands user intent and responds intelligently. These methods combined in a hybrid approach show how AI can solve real-world problems in creative and efficient ways.

References

Kaggle. (n.d.). *TMDB movie metadata*. Retrieved April 14, 2025, from https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata

Loria, S. (n.d.). *TextBlob: Simplified text processing in Python* (Version dev). Retrieved April 14, 2025, from https://textblob.readthedocs.io/en/dev/

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830. Retrieved from https://scikit-learn.org/stable/

Pallets Projects. (n.d.). *Flask (2.3) documentation*. Retrieved April 14, 2025, from https://flask.palletsprojects.com/

scikit-learn developers. (n.d.). *Text feature extraction using TF-IDF and cosine similarity*. Retrieved April 14, 2025, from https://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction