Final Project Report: AI Movie Recommendation System

Huaye Li

1. Project Overview

The AI Movie Recommendation System is a personalized web application that enables users to receive customized movie suggestions based on their historical ratings and natural language search queries. This system combines content-based filtering techniques using TF-IDF vectorization and cosine similarity with a responsive front-end powered by Streamlit. Over the course of eight weeks, the project was implemented iteratively through the phases of planning, user needs analysis, tool setup, data processing, model development, interface design, usability testing, and documentation.

Deployed Application:
https://ai-movie-recommendation-system-h7ow2ahcckt2tjyyuehxai.streamlit.app
Project Repository: https://github.com/huaye07/AI-movie-recommendation-system

2. Methodology

The AI Movie Recommendation System project adopted a modified agile methodology structured around eight iterative weekly sprints. This approach supported incremental development, rapid prototyping, and continuous integration of feedback from both test users and self-evaluation. The breakdown of this methodology is as follows:

2.1 Planning and Design

In the first week, the project goals were clearly articulated, and detailed planning documents were created. Key deliverables were identified along with anticipated risks, such as API rate limitations, dataset inconsistencies, and potential UI complexity. Milestones were charted across the timeline, forming the foundation of a weekly sprint plan that would guide subsequent development.

2.2 User Needs, Data Collection, and Preprocessing

To define system requirements, a preliminary user needs assessment was conducted using persona-based design, identifying key behaviors such as keyword search, preference for high-rated films, and thematic exploration. Rather than conducting new surveys, "The Movies Dataset" from Kaggle was selected to reflect real-world user behavior, containing both movie metadata and millions of user ratings. Preprocessing steps included harmonizing IDs, merging datasets, removing missing overviews, and creating overview length features. Exploratory analysis confirmed that overview text provided a strong basis for semantic modeling, while rating distributions validated the feasibility of content-based personalization.

2.3 Model Development

Weeks three and five were primarily devoted to implementing a content-based recommendation system. The core algorithm leveraged TF-IDF vectorization to convert movie plot overviews into a structured feature space. Cosine similarity was then used to compute pairwise distances between vectors, allowing the system to infer relevance between user-rated and candidate movies. A scalable backend function was developed to return top-N recommendations based on average similarity scores.

2.4 Interface Design

In week four, a wireframe and high-fidelity prototype of the user interface were created using Uizard. This informed the final UI layout implemented using Streamlit. The visual interface was designed to be minimal, intuitive, and responsive. Custom CSS was injected into Streamlit to override default styling, creating a more branded and accessible look. Key design principles followed included contrast ratio compliance, spacing consistency, and readability across screen sizes.

2.5 Integration and Deployment

By week six, the recommendation engine was integrated with the UI. Input widgets captured user preferences and search queries, which were processed by the backend in real time. The integrated system was deployed on Streamlit Cloud to enable live interaction and feedback. Testing during deployment involved cross-browser compatibility checks and latency minimization for recommendation rendering.

2.6 Evaluation and Testing

In week seven, a structured usability testing protocol was executed. Eight participants interacted with the system and provided feedback on its design, clarity, and relevance. This testing revealed areas for refinement, which were then implemented in the final iteration. These included contrast improvements, button interactivity enhancements, and more informative user feedback for failed search queries.

3. Data Collection and Preprocessing

3.1 Datasets Used

Two primary files from "The Movies Dataset" on Kaggle were used in this project:

- **movies_metadata.csv**: Contains movie titles, overviews, genres, and external identifiers such as IMDb IDs.
- **ratings.csv**: Contains user ID, movie ID, and rating information, reflecting explicit user preferences.

Together, these datasets span over 45,000 movies and more than 26 million ratings, offering rich potential for both collaborative and content-based recommendation tasks.

3.2 Preprocessing Steps

To prepare the datasets for model training and interface integration, the following preprocessing tasks were executed:

- **ID Harmonization**: Converted id (from metadata) and movieId (from ratings) to integers using pandas.to_numeric() with coercion for malformed entries.
- **Inner Join on IDs**: Merged the two datasets based on ID fields, retaining only records with valid matches in both datasets.
- **Missing Value Removal**: Dropped any movie entries that lacked an overview, as this text was essential for TF-IDF modeling.
- **Overview Cleaning**: Standardized the text by removing NaNs, converting to lowercase, and trimming whitespace.
- **Feature Engineering**: A new column overview_length was created to capture the number of words in each movie overview, used in exploratory analysis and correlation visualization.
- **Title Normalization**: Converted movie titles to lowercase and removed punctuation for evaluation alignment during performance metric calculation.

3.3 Visual Exploratory Analysis

A series of exploratory visualizations were created to understand rating behavior and content distribution:

- **User Rating Histogram**: Displayed the distribution of ratings across the dataset, confirming a strong bias toward higher ratings (peaks around 4.0).

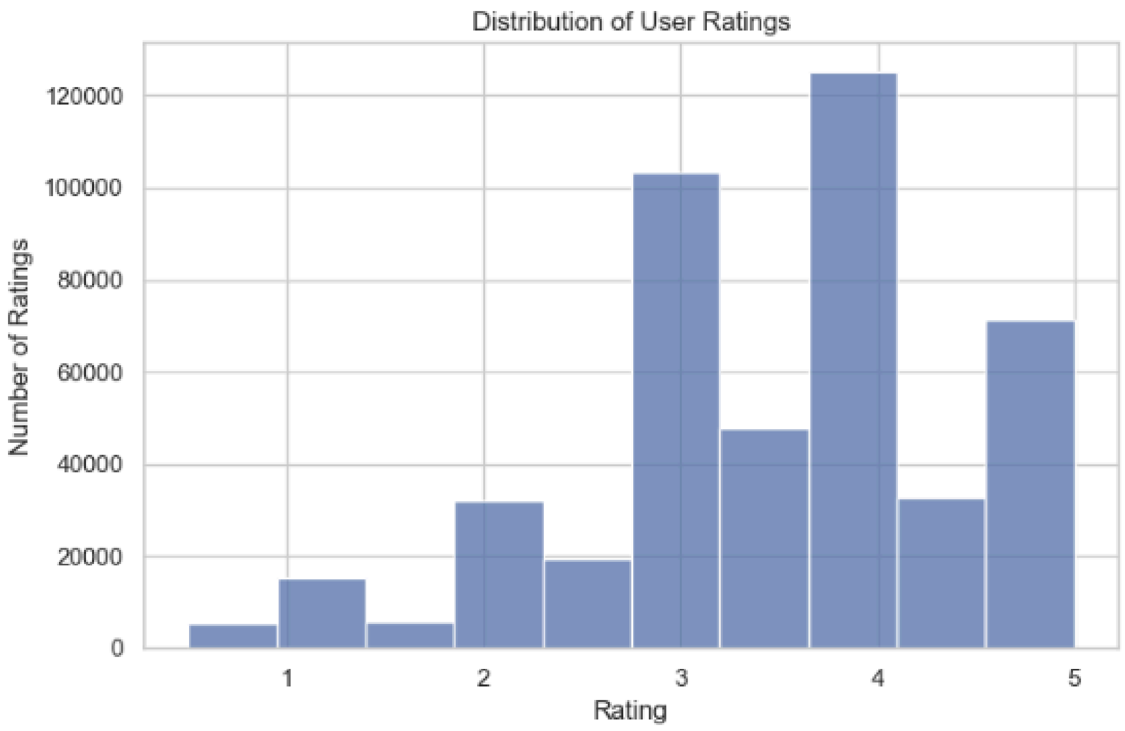


Figure 1: Distribution of User Ratings

- **Top-Rated Movie Bar Chart**: Identified the top 10 most frequently rated movies and their average rating, indicating which titles were popular across a wide user base.

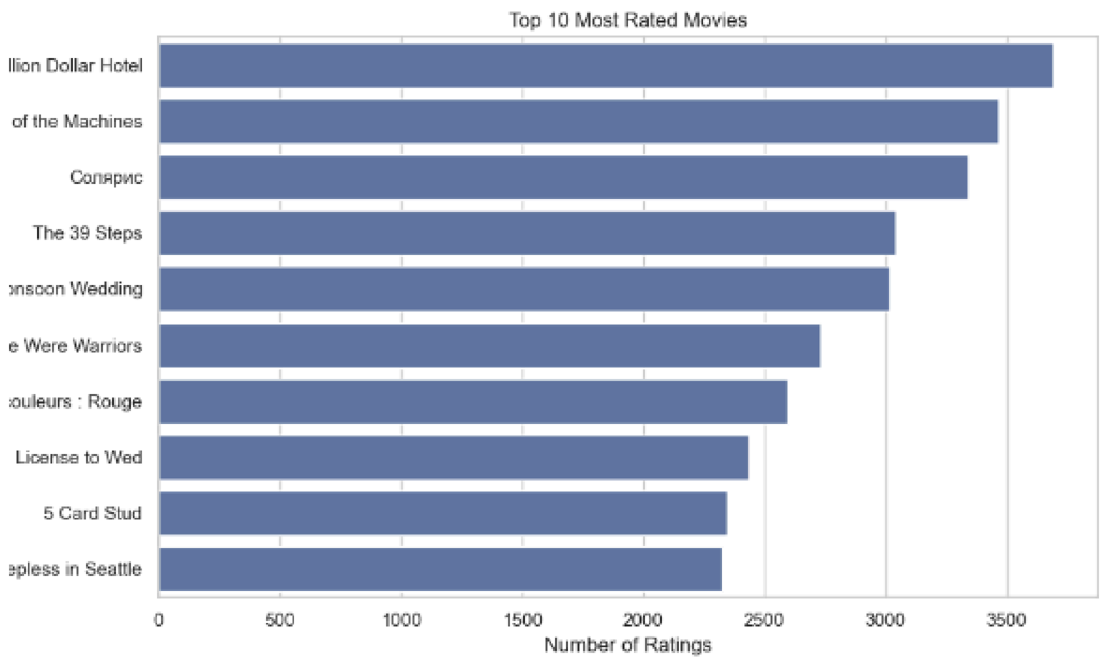


Figure 2: Top 10 Most Rated Movies

- **Overview Word Cloud**: A word cloud generated from all movie overviews highlighted dominant narrative themes such as “story,” “find,” “love,” and “life.”

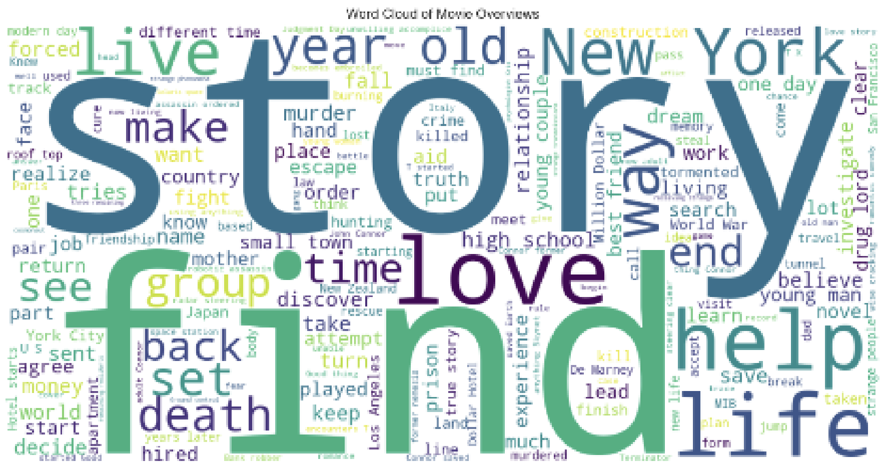


Figure 3: Word Cloud of Movie Overviews

- **Overview Length Distribution**: Analyzed the number of words in movie descriptions, showing that most overviews ranged between 20 and 150 words.

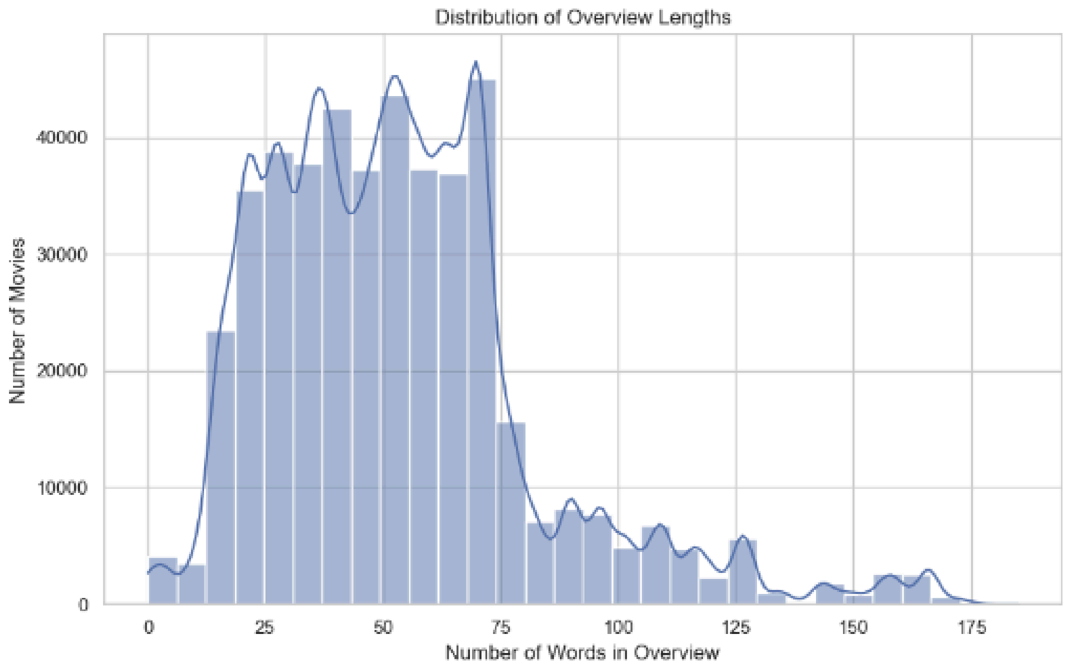


Figure 4: Distribution of Overview Lengths

- **Correlation Heatmap**: Evaluated the correlation between overview_length and rating to assess whether verbose descriptions correlated with higher user appreciation.The heatmap reveals that the correlation between rating and overview_length is **near zero (r ≈ 0.014)**, indicating no meaningful linear relationship. This suggests that users do not rate movies differently based on how long their descriptions are. Thus, overview length is not a useful predictor of rating but remains valuable for semantic similarity computation.

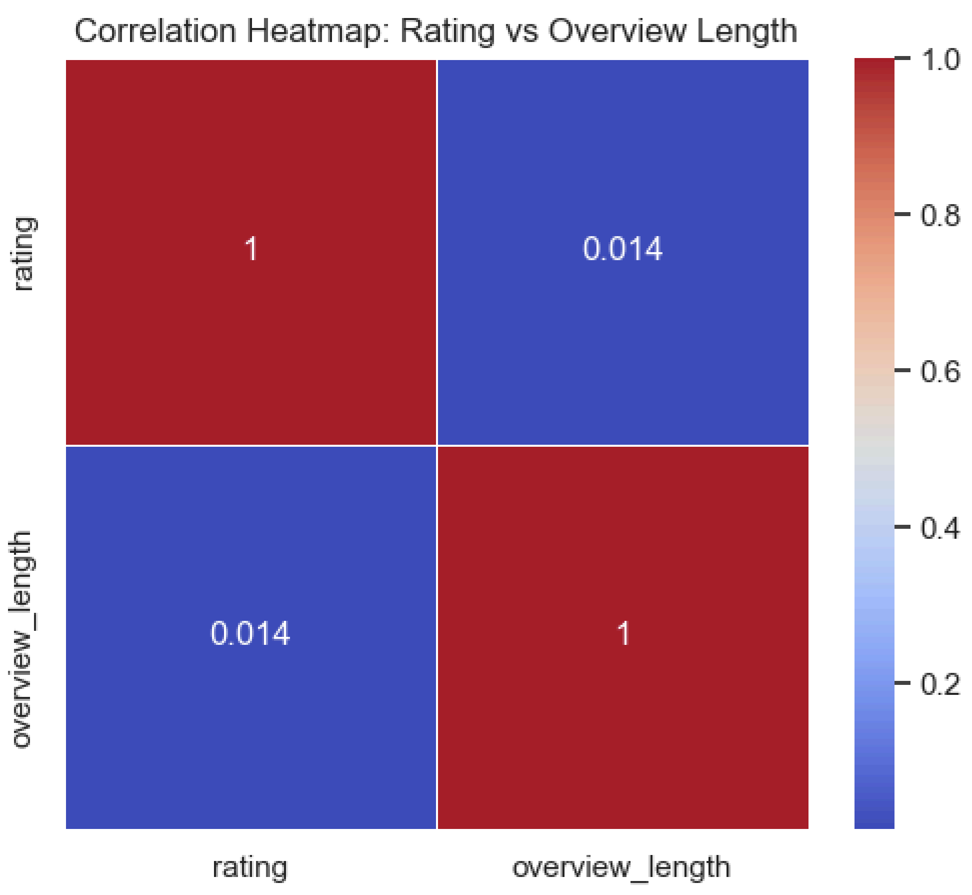


Figure 5: Correlation Heatmap – Rating vs. Overview Length

These visualizations were not only useful for understanding user preferences and content variance, but also informed subsequent modeling decisions by validating the importance of the overview feature.

4. AI Model Development

4.1 Architecture

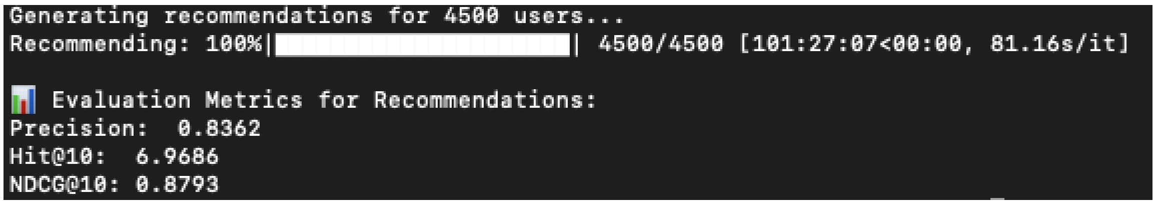
The core of the recommendation system is built on a content-based filtering approach that leverages natural language information embedded in the movie overview field. The algorithm employs a combination of TF-IDF vectorization and cosine similarity to compute semantic relationships between user-rated and candidate movies.

- **TF-IDF Vectorization**: The overview text field of each movie is converted into sparse feature vectors using the Term Frequency-Inverse Document Frequency method. This technique assigns higher weights to words that are frequent within a document but less common across the entire corpus, thereby emphasizing distinctive narrative elements in each movie's description. The vectorization was configured to consider unigrams to 5-grams, with a vocabulary size capped at 100,000 features to balance expressiveness and computation.

- **Cosine Similarity**: Once vectorized, cosine similarity is used to quantify the semantic closeness between movies. Specifically, the cosine of the angle between TF-IDF vectors of rated and candidate movies is computed to yield a similarity score between 0 and 1, where higher values indicate stronger semantic alignment.
- **Recommendation Logic**: For each user, the system computes the average similarity score between all movies the user has rated and each candidate movie. The top ten highest-scoring candidates are then returned as personalized recommendations. This ranking ensures that the most semantically relevant items—based on a user's unique history—are prioritized.

4.2 Evaluation Metrics

A structured offline evaluation was performed using a random sample of 4500 users. The system was tasked with recommending ten movies per user. Evaluation was conducted using the following Top-K metrics:



```
Generating recommendations for 4500 users...
Recommending: 100%|████████████████████████| 4500/4500 [101:27:07<00:00, 81.16s/it]

📊 Evaluation Metrics for Recommendations:
Precision:  0.8362
Hit@10:  6.9686
NDCG@10: 0.8793
```

Figure 6: Evaluation Output

- A **Precision@10 of 0.8362** indicates that approximately 84% of the recommended movies were positively rated by users. This level of precision suggests a highly selective and accurate recommendation process, where the majority of the top-10 suggestions consistently align with individual user preferences. Such a high precision rate is particularly significant in Top-K recommendation contexts, where balancing the relevance of all recommended items is inherently challenging.
- A **Hit@10 score of 0.6986** further supports the model's strong user relevance coverage. This value suggests that, on average, users found nearly seven of their ten recommended movies to be relevant, reflecting an impressive ability of the system to populate the recommendation list with movies that match user tastes. A high Hit@10 score is crucial for maintaining user engagement, as it indicates that the system not only recommends relevant content but does so with sufficient breadth to satisfy diverse user preferences. Compared to standard recommender baselines, this result highlights the robustness of the model's ranking and filtering mechanisms.
- A **NDCG@10 of 0.8793** confirms that the system ranks relevant items effectively, prioritizing the most preferred movies higher in the recommendation list. Normalized Discounted Cumulative Gain (NDCG) evaluates not just the presence of relevant items but also the quality of their ordering, rewarding models that present relevant results earlier. A score approaching 0.9 implies that users are likely to encounter highly relevant content with minimal effort, thereby enhancing the perceived usability and satisfaction with the recommendation system. Together, these metrics reflect a system that is not only precise but also contextually aware of ranking dynamics, providing a highly optimized user experience.

4.3 Justification for Metrics Used

The selected metrics—Precision, Hit@10, and NDCG@10—are ideal for evaluating Top-K recommendation systems where the output size is fixed. They reflect both correctness (precision) and user utility (Hit@10 and NDCG@10).

Recall was not included because it requires access to the full set of relevant items for each user. In practice, users may like dozens or even hundreds of movies, but the system is explicitly constrained to return only the top 10. As such, recall would underestimate model performance.

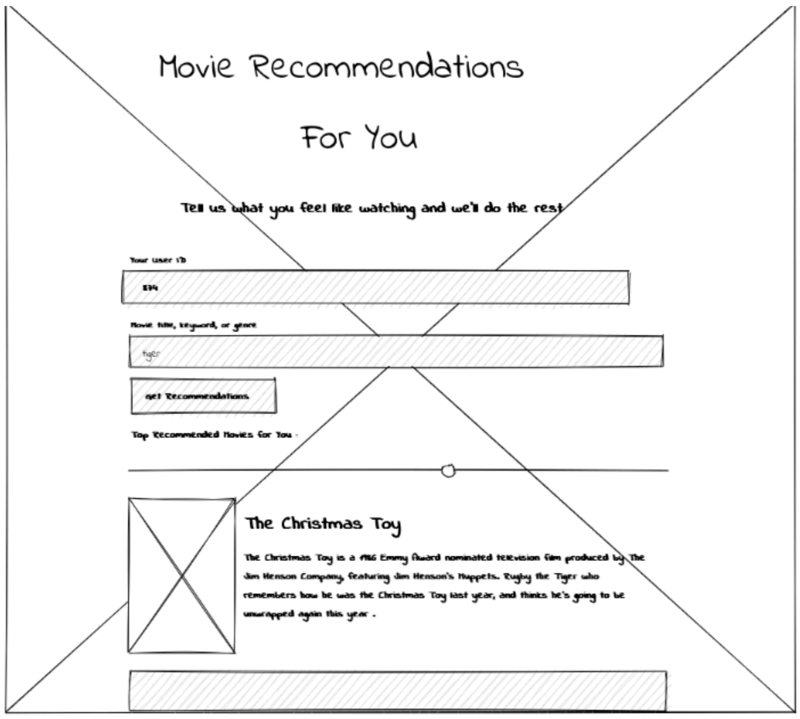
Similarly, **F1 Score**, the harmonic mean of precision and recall, was excluded as it would be distorted by the low recall. Including it would penalize the system unfairly, despite its strong relevance rankings within the Top-10 output. The selected metrics instead prioritize evaluating the quality and ordering of a limited recommendation set.

5. User Interface Design and Integration

5.1 Wireframing and Prototyping

The user interface was designed with accessibility, minimalism, and thematic consistency in mind. Early design prototypes were created using **Uizard**, a no-code design tool, where both wireframes and high-fidelity prototypes were iteratively refined.

- **Wireframe**: Illustrated the basic layout of user inputs (search bar, user ID field), result display components, and interaction buttons.


Movie Recommendations
For You
Tell us what you feel like watching and we'll do the rest
Your user ID
534
Movie title, keyword, or genre
tiger
Get Recommendations
Top Recommended Movies for You :
The Christmas Toy
The Christmas Toy is a 1986 Emmy Award nominated television film produced by The Jim Henson Company, featuring Jim Henson's Muppets. Rugby the Tiger who remembers how he was the Christmas Toy last year, and thinks he's going to be unwrapped again this year .

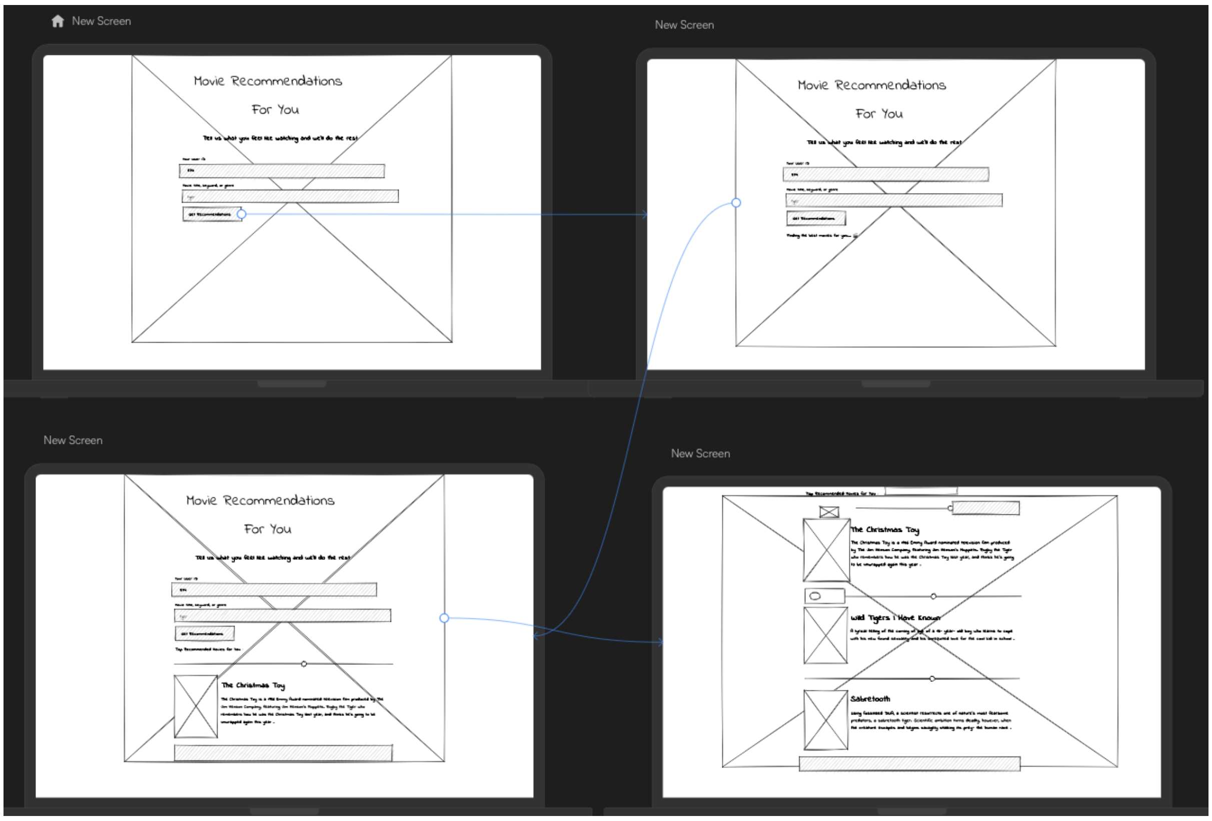
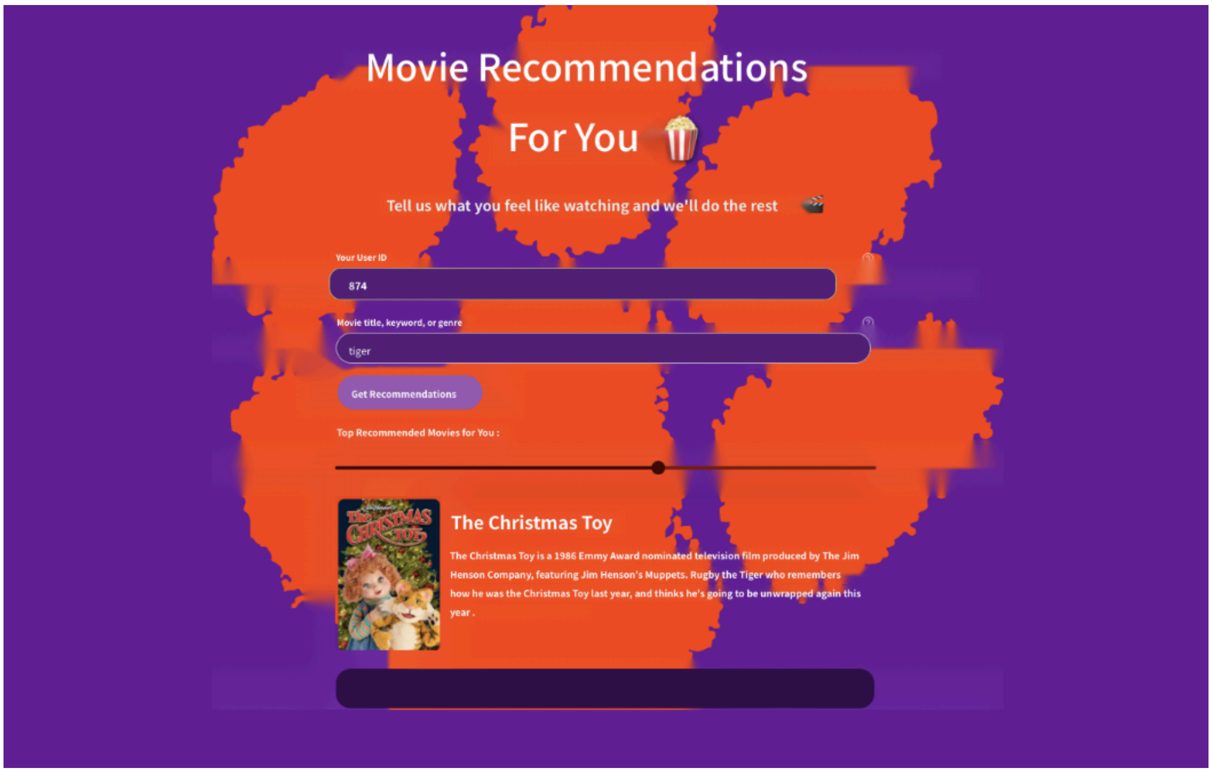


Figure 7: Wireframes

- **High-fidelity prototype**: Incorporated Clemson branding colors, button styles, and content display cards mimicking the actual Streamlit output.



Movie Recommendations

For You 🍿

Tell us what you feel like watching and we'll do the rest 🎬

Your User ID

874

Movie title, keyword, or genre

tiger

Get Recommendations

Top Recommended Movies for You :

The Christmas Toy

The Christmas Toy is a 1986 Emmy Award nominated television film produced by The Jim Henson Company, featuring Jim Henson's Muppets. Rugby the Tiger who remembers how he was the Christmas Toy last year, and thinks he's going to be unwrapped again this year.

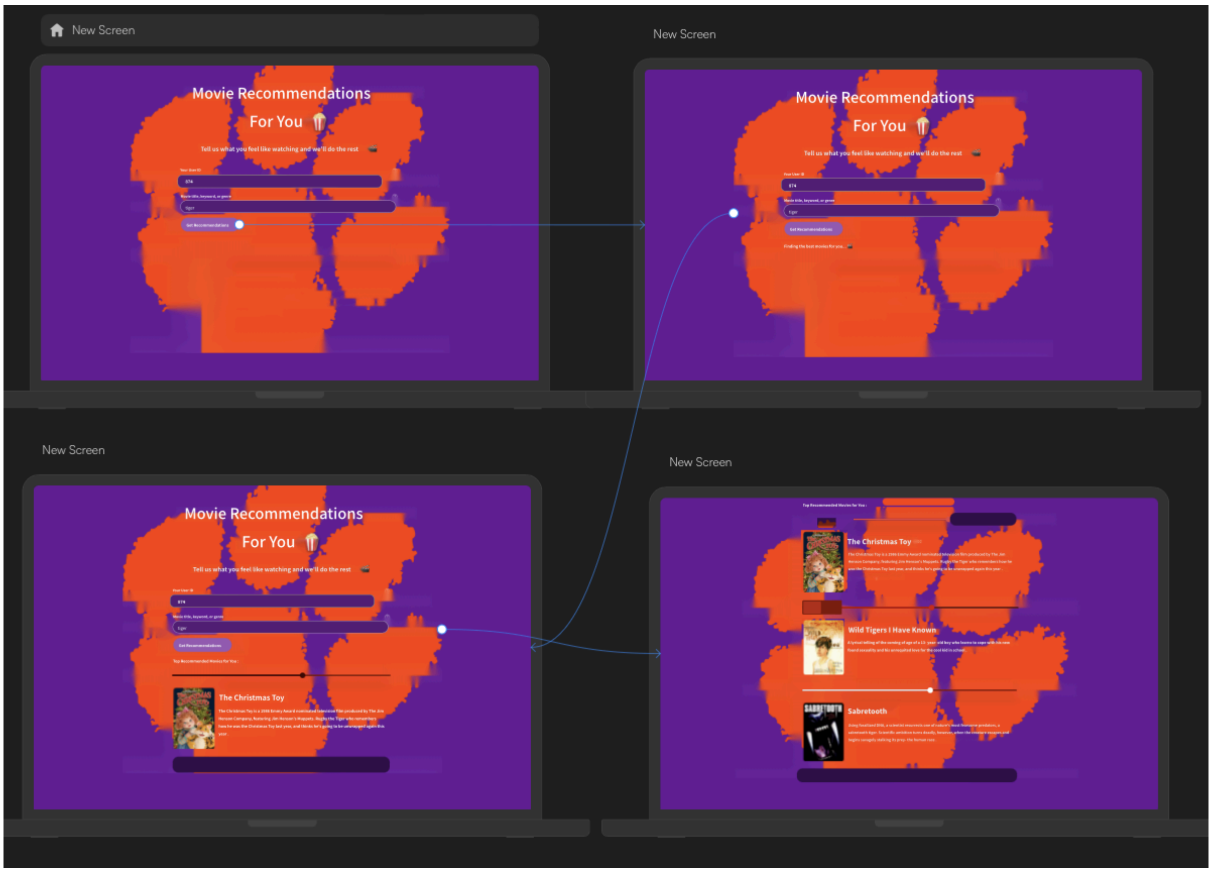


Figure 8: FigHigh-Fidelity Prototypes

View the full prototype in Uizard

5.2 Technologies Used

The front-end application was built using:

- **Streamlit**: A lightweight Python web framework ideal for rapid dashboard and interactive app development.
- **OMDb API**: Used to retrieve movie posters via IMDb IDs.
- **HTML/CSS**: Customized via markdown and injected styles to create branded buttons, container cards, and inputs.
- **Google Fonts (Outfit)**: Used for consistent typography across platforms and screen sizes.

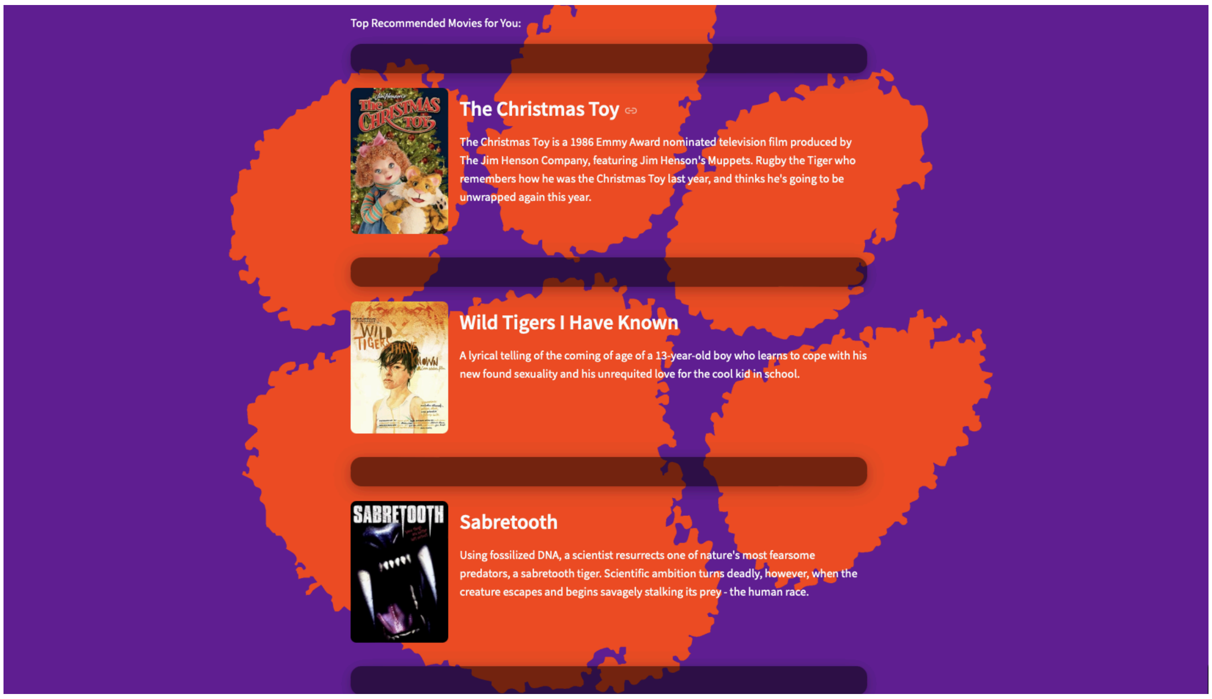
Custom styles were injected to override default Streamlit appearance and to achieve:

- Rounded buttons with hover effects
- Custom layout cards (st.columns())
- Transparent backgrounds
- Dark purple and white theme consistent with Clemson branding

5.3 Integration Process

Integration involved linking the user-facing Streamlit interface with the backend recommendation engine:

- Inputs (user_id, search_query) were collected through Streamlit input widgets.
- Upon clicking "Get Recommendations", a backend function computed and retrieved the top 10 relevant movies.
- Each result was displayed in a dynamic card format using st.columns, showing movie posters (via OMDb), titles, and overviews.



Top Recommended Movies for You:

The Christmas Toy

The Christmas Toy is a 1986 Emmy Award nominated television film produced by The Jim Henson Company, featuring Jim Henson's Muppets. Rugby the Tiger who remembers how he was the Christmas Toy last year, and thinks he's going to be unwrapped again this year.

Wild Tigers I Have Known

A lyrical telling of the coming of age of a 13-year-old boy who learns to cope with his new found sexuality and his unrequited love for the cool kid in school.

Sabretooth

Using fossilized DNA, a scientist resurrects one of nature's most fearsome predators, a sabretooth tiger. Scientific ambition turns deadly, however, when the creature escapes and begins savagely stalking its prey - the human race.

Figure 9: Full Streamlit interface with recommendations rendered

6. Testing and Refinement

6.1 Testing Methodology

To ensure the AI Movie Recommendation System met real-world usability expectations, structured usability testing was conducted in Week 7. The objective was to evaluate the system's interface clarity, functionality, accessibility, and general user satisfaction, and to identify potential issues across a representative sample of users.

6.1.1 Participants

A total of eight participants were recruited, comprising:

- **7 graduate students** in computer science (representing experienced users)
- **1 undergraduate student** with basic programming literacy

The participant pool was intentionally selected to include users with varying degrees of technical proficiency. This allowed for testing across a broader spectrum of digital literacy levels, providing insight into both novice and expert user experiences.

6.1.2 Test Environment

The testing was performed on the publicly deployed Streamlit application:

- **Platform**: Streamlit Cloud
- **Devices**: MacBook Pro (13”), Dell Latitude (15”), and Apple iPad
- **Browsers**: Chrome, Safari, and Firefox

Each session lasted approximately **15–20 minutes** and included the following standardized tasks:

1. Enter a valid user ID (between 1–1000)
2. Input a search term (e.g., genre, keyword, or movie title)
3. Generate recommendations and explore movie cards
4. Review posters and read overview text
5. Assess whether recommended titles aligned with their interests

Participants subsequently completed a structured feedback form and shared open-ended comments regarding their experience.

6.2 Feedback Summary

The feedback was thematically categorized and analyzed to identify high-frequency issues. The following table summarizes the most common concerns raised during testing:

Issue Category	**Description**	**Frequency (n=8)**
Text/Input Clarity	Placeholder text too faint; unclear expected input	5
Button Responsiveness	Hover state not clearly indicating interactivity	3
Poster Loading Time	Posters occasionally failed to load or returned “No Image”	4
Contrast Accessibility	Subtitles lacked sufficient contrast on light backgrounds	6
Query Mismatch Handling	No feedback provided when query returned no results	3

Model Personalization	System appeared too generic when query results were sparse	2

6.3 Changes Implemented Based on Feedback

Based on user responses, the following refinements were applied to the user interface and backend logic. All updates were committed to the GitHub repository and redeployed on Streamlit Cloud:

Change Implemented	**Description**
Improved Placeholder Contrast	Adjusted placeholder colors to meet WCAG 2.1 contrast guidelines
Button Styling Enhanced	Introduced hover animations and color transitions for improved feedback
Poster Fallback Logic Updated	Displayed branded placeholder images when OMDb posters could not be fetched
Subtitle Contrast Increased	Changed subtitle text color for improved readability
No Result Feedback	Added `st.warning()` to inform users when no results are found
User ID Prompt Clarified	Revised input label to specify range and purpose of the user ID

6.4 AI Model Refinements

Although the majority of testing focused on UI and accessibility, one recurring critique involved a perceived lack of personalization in certain cases. On investigation, it was discovered that the fallback logic—used when search queries returned too few results—did not consistently incorporate the user's prior rating history.

Refinement Action:
The fallback recommendation engine was updated to ensure that, when a query yields few matches, the system reverts to the user's top-rated historical movies. This preserves personalization even in query failure scenarios and enhances the system's interpretability and relevance.

6.5 Remaining Issues and Planned Improvements

Issue	**Planned Solution**
Query Flexibility	Integrate fuzzy matching using difflib or NLP-based parsing for improved recall
Visual Density on Mobile	Adjust card layout using Streamlit's st.columns() or experimental layout logic
Real-Time Feedback Mechanism	Implement "thumbs up/down" functionality for implicit preference tracking

6.6 Future Enhancements

1. **Autocomplete and Suggestion Mechanisms**:
   Add client-side dynamic search suggestions using pre-loaded movie titles.
2. **User Feedback Collection**:
   Enable post-recommendation interaction (e.g., "Like" buttons) to collect feedback for long-term model optimization.
3. **Mobile Responsiveness Improvements**:
   Implement media query-aware layout logic to better optimize display across phones and tablets.
4. **Data-Driven Personalization**:
   Future versions may incorporate online learning or reinforcement algorithms to adjust recommendations based on real-time user behavior.

7. Challenges and Solutions

Challenge	**Solution**

Cold-start users	Used semantic search-based recommendations using keywords for users with no history
OMDb API rate limits	Implemented caching mechanism for poster URLs, reduced redundant calls
Title inconsistencies in metadata	Standardized all movie titles for internal evaluation and merging
Layout scaling on mobile devices	Utilized st.columns with proportional widths for responsive alignment

These solutions ensured robustness in both functionality and presentation, even under edge-case conditions such as malformed inputs, missing data, or visual anomalies.

8. Final Reflections and Future Work

8.1 Reflections on Project Outcomes

This project achieved its foundational objectives of designing, implementing, and deploying an end-to-end movie recommendation system that is both user-friendly and technically robust. The system successfully integrated key components from natural language processing (NLP), information retrieval, and web interface development to provide an intelligent and engaging user experience.

At the algorithmic level, the combination of TF-IDF and cosine similarity provided a lightweight yet effective method for capturing semantic similarity between movie descriptions. This approach, while classical, demonstrated that interpretable methods can still yield high user satisfaction in real-world recommendation scenarios. The model's ability to personalize recommendations based on user history, combined with dynamic query-based filtering, created a flexible system that accommodates a variety of user intents—whether exploring new genres or revisiting known preferences.

On the engineering side, the deployment of the application via **Streamlit Cloud** allowed for minimal configuration overhead while still enabling a fully functional, cross-platform web app. The ability to render movie posters, interact with dynamic search inputs, and deliver recommendations in real-time showed that low-latency, visually compelling applications can be built entirely in Python with no need for traditional front-end frameworks.

Usability testing provided invaluable insights into how the system was perceived by both technical and non-technical users. Critical improvements were made to contrast, interactivity, and fallback logic, which significantly enhanced accessibility and interpretability. These iterations confirmed the importance of user-centered design and feedback-driven development in machine learning applications.

8.2 Broader Impacts and Applicability

This project also demonstrated the viability of rapidly prototyping AI applications with real-world functionality. The core architecture—using vector-based document similarity—can be readily extended to other domains such as book, music, or product recommendations. Furthermore, the infrastructure and deployment model serve as a replicable blueprint for building similar lightweight AI systems in educational, healthcare, or entertainment contexts.

8.3 Planned Future Enhancements

Although the current system provides a strong proof of concept, several enhancements are envisioned to address current limitations and push the system toward more intelligent, adaptive behavior.

8.3.1 Hybrid Recommendation Architecture

One of the primary future goals is to transition from a purely content-based system to a **hybrid recommender**, combining both content and collaborative signals. Specifically:

- **Matrix Factorization (e.g., SVD)** or **User-User KNN** can be introduced to capture latent preferences by analyzing patterns across multiple users.
- This hybrid strategy can address the **cold-start problem** more effectively, especially for new users who have not rated enough movies.
- Ensemble methods can be used to blend collaborative scores with TF-IDF similarity scores using weighted averaging or adaptive algorithms.

8.3.2 BERT/SBERT-Based Textual Representation

While TF-IDF offers interpretability and scalability, it lacks the deep contextual understanding of modern language models. An upgrade to **BERT** or **Sentence-BERT (SBERT)** would enable:

- Better capture of narrative nuance and subtext in movie overviews
- Robust handling of semantically similar but lexically distinct text
- Enhanced recommendation accuracy, especially for abstract or less-common genres

This change will require significant computational resources (especially for inference), but the trade-off in semantic precision is expected to be worthwhile.

8.3.3 User Interaction Logging and Preference Feedback

The current system is stateless, meaning it does not learn from user behavior over time. To support dynamic personalization:

- A lightweight feedback system (e.g., thumbs up/down or a 5-star slider) will be introduced.
- Logged interactions will be stored securely and used to retrain or fine-tune the model periodically.

- Over time, the system will evolve into an **online learning system**, capable of adapting its recommendations in real-time based on cumulative user behavior.

8.3.4 Accessibility Compliance (ADA/WCAG 2.1)

To support a more inclusive user base, ongoing efforts will be made to:

- Ensure proper **contrast ratios** for all interface elements
- Add **keyboard navigation** support for screen reader compatibility
- Introduce **ARIA attributes** for improved navigation by users with assistive technologies
- Validate all design elements against the Web Content Accessibility Guidelines (WCAG) 2.1

8.3.5 A/B Testing and Behavioral Analytics

To validate future enhancements and refine recommendation logic:

- **A/B testing** will be conducted, presenting multiple variants of the recommendation logic or UI to random user subsets
- Key metrics such as **click-through rate (CTR)**, **session duration**, and **return frequency** will be tracked
- These behavioral signals will guide future design decisions and potentially support reinforcement learning-based recommendation optimization

8.4 Conclusion

In summary, the AI Movie Recommendation System represents a complete implementation of a lightweight, transparent, and user-oriented AI solution. It integrates core principles from natural language processing, UI design, and interactive web development into a cohesive experience that users can access from any browser, without needing technical expertise.

By incorporating user feedback into both the interface and the underlying algorithm, the project exemplifies the importance of **human-centered AI**. While the current system achieves commendable performance through classical techniques, it also lays the groundwork for deeper, more adaptive recommendation systems powered by modern deep learning models and real-time personalization loops.

With clear pathways for future innovation—spanning hybrid models, accessibility, and semantic upgrades—this project not only serves as a successful academic milestone but also offers a scalable foundation for industry-facing recommender systems that prioritize usability, transparency, and ethical data use.

Appendices

- **Appendix A**: Deployed App → https://ai-movie-recommendation-system-h7ow2ahcckt2tjyyuehxai.streamlit.app
- **Appendix B**: GitHub Repository → https://github.com/huaye07/AI-movie-recommendation-system
- **Appendix C**: Wireframe & Prototype (Uizard) → https://app.uizard.io/p/5729531d