Kalyani Government Engineering College

Title of the Project: Movie and Book Recommendation System:

Name: Suraj Kundu
Year: 4th
Roll: 10200320024
Registration no: 201020100310019
Dept: Electronics and Communication Engineering

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled **"Movie and Book Recommendation System"** in partial fulfilment of the requirements for the award of the degree of **BACHLOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING at ARDENT COMPUTECH PVT. LTD, SALTLAKE, KOLKATA, WEST-BENGAL** is an authentic work carried out under the guidance of **Mr. Dhruba Ray.** The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

CERTIFICATE

This is to certify that this proposal of a minor project entitled **"Movie and Book Recommendation System"** is a record of bonafide work, carried out by

Suraj Kundu under my guidance at **ARDENT COMPUTECH PVT. LTD.** In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY and as per regulations of the ARDENT®. To the best of my knowledge, the results embodied in this report are original in nature and worthy of incorporation in the present version of the report.

Guide/Supervisor
Mr. Dhruba Ray
Ardent Computech Pvt Ltd

ACKNOWLEDGEMENT

Thanks giving seems to be the most pleasant of all jobs but it is none of the less difficult when one tries this put them in words. Before we get thick of the things we would like to add a few heartfelt words for the people who were part of this project **"Movie and Book Recommendation System"** in numerous ways people who gave unending support right from the start.

I express my sincere thanks to our Faculty **Mr. Dhruba Ray** for his novel association of ideas, encouragement, appreciation and intellectual zeal which motivated us to make this project successfully.

Finally, it is pleased to acknowledge the indebtedness to all those who devoted themselves directly or indirectly to make this project report success.

CONTENTS

Summary:

The report explores the development and implementation of a recommendation system for movies and books. The system aims to enhance user experience by providing personalized suggestions based on individual preferences. Through a combination of collaborative filtering and content-based approaches, the system analyzes user behavior and item characteristics to generate accurate and relevant recommendations. The report discusses the challenges encountered during the system's development and proposes potential solutions. Additionally, it highlights the significance of continuous improvement through user feedback and the integration of advanced algorithms to further enhance the recommendation accuracy. Overall, the movie and book recommendation system represent a promising solution to address the diverse entertainment preferences of users, offering a seamless and enjoyable content discovery experience.

Introduction:

In the rapidly evolving landscape of entertainment consumption, the demand for personalized recommendations has become increasingly prevalent. With an overwhelming array of movies and books available, users often find themselves facing the challenge of navigating through a vast sea of content to discover what aligns with their preferences. Recognizing this need, recommendation systems have emerged as indispensable tools, leveraging advanced algorithms to analyze user behavior, preferences, and content attributes. This report delves into the intricate workings of movie and book recommendation systems, exploring their significance in enhancing user experience, the underlying technologies driving their functionality, and the challenges faced in creating effective and accurate recommendations. By examining the convergence of artificial intelligence, machine learning, and user profiling, this report aims to shed light on the innovative strategies employed by these systems to cater to individual tastes, fostering a more engaging and tailored content discovery experience.

System Architecture for Movie and Book Recommendation System:

The Movie and Book Recommendation System is designed to provide personalized recommendations to users based on their preferences and viewing/reading history. The system employs a collaborative filtering approach, combining user preferences and item similarities to generate accurate and relevant recommendations.

1. User Interface (UI):

- The front-end of the system is a user-friendly interface accessible through web or mobile applications.
- Allows users to create accounts, log in, and manage their profiles.
- Provides a search functionality for users to discover new movies and books

2. User Profile Management:

- Handles user authentication, registration, and profile management.
- Stores user preferences, ratings, and viewing/reading history.

3. Recommendation Engine:

- Collaborative Filtering: Utilizes user-item interactions and similarities to recommend movies and books.
- Content-Based Filtering: Analyzes the content of movies and books to recommend items with similar characteristics.
- Hybrid Approach: Combines collaborative and content-based filtering for enhanced accuracy.

4. Data Processing and Storage:

- Database: Stores user profiles, movie details, and book details.
- Data Preprocessing: Cleans and processes raw data to extract relevant features.
- Caching: Implements caching mechanisms to optimize recommendation generation by storing frequently accessed data.

5. Machine Learning Models:

- Trains models to identify user preferences and item similarities.
- Model Training: Uses historical user interactions and item features to train collaborative and content-based models.
- Real-Time Updates: Allows for real-time model updates based on user feedback.

6. Recommendation Algorithm:

- Determines the most suitable algorithm based on the user's behavior and preferences.
- Adjusts recommendation weights dynamically to adapt to user feedback.

7. API Layer:

- Provides an interface for communication between the front-end and the recommendation engine.
- Supports RESTful APIs for data retrieval, user authentication, and recommendation requests.

8. Security:

- Implements robust security measures to protect user data and system integrity.
- Uses encryption for sensitive information and follows best practices for user authentication.

9. Logging and Analytics:

- Logs user interactions, system events, and errors for monitoring and analysis.
- Integrates analytics tools to gather insights into user behavior and system performance.

10. Scalability and Performance:

- Design supports horizontal scalability to handle increasing user loads.
- Optimizes database queries and recommendation algorithms for efficient performance.

11. Deployment:

- Utilizes cloud services for scalable and reliable deployment.
- Implements containerization for easy deployment and management.

12. Continuous Integration/Continuous Deployment (CI/CD):

- Implements CI/CD pipelines for automated testing, building, and deployment.
- Ensures rapid and reliable updates to the system.

The Movie and Book Recommendation System architecture described above provides a comprehensive solution for delivering personalized recommendations to users, enhancing their overall experience with the platform.

Algorithms Used:

The recommendation system plays a crucial role in providing personalized content suggestions to users, enhancing their experience by offering relevant movies and books based on their preferences. This report explores the algorithms employed in a movie and book recommendation system, shedding light on the methodologies that drive the system's ability to make accurate and personalized suggestions.

Collaborative Filtering: Collaborative filtering is a widely employed algorithm in recommendation systems. It relies on user behavior and preferences to make predictions. Two main types of collaborative filtering are user-based and item-based. User-based collaborative filtering suggests items that users with similar preferences have liked, while item-based collaborative filtering recommends items similar to those the user has liked in the past.

Content-Based Filtering: Content-based filtering recommends items by analyzing the features and attributes of the items themselves. In the context of movies and books, this algorithm considers factors such as genre, actors, authors, and keywords. By understanding the content of items, the system can suggest similar items based on shared characteristics.

Matrix Factorization: Matrix factorization is a technique used to decompose a large matrix into smaller matrices, enabling the system to discover latent factors that contribute to user-item interactions. Singular Value Decomposition (SVD) is a popular matrix factorization method that breaks down the user-item interaction matrix into three matrices, revealing underlying patterns that aid in making recommendations.

Neural Networks: Deep learning, particularly neural networks, has gained popularity in recommendation systems. Neural collaborative filtering models leverage neural networks to learn complex patterns from user-item interactions. These models can capture intricate relationships and dependencies, providing more accurate and personalized recommendations.

Hybrid Models: Hybrid recommendation systems combine multiple algorithms to overcome the limitations of individual approaches. By integrating collaborative filtering, content-based filtering, and other techniques, hybrid models aim to provide well-rounded and more accurate suggestions, taking advantage of the strengths of each algorithm.

Ensemble Methods: Ensemble methods involve combining predictions from multiple recommendation algorithms to enhance the overall accuracy and robustness of the system. Techniques such as stacking and bagging can be employed to aggregate diverse predictions, resulting in a more reliable recommendation system.

The effectiveness of a movie and book recommendation system lies in the careful selection and integration of algorithms. Collaborative filtering, content-based filtering, matrix factorization, neural networks, hybrid models, and ensemble methods are key components that contribute to the system's ability to understand user preferences and deliver personalized suggestions.

Code:

Part 1: MOVIE RECOMMENDATION SYSTEM(JUPYTER NOTEBOOK)

Movie and Book Recommendation System Jupyter Notebook

Our Jupyter Notebook is the cornerstone of our innovative movie and book recommendation system. It seamlessly integrates data analysis, machine learning models, and interactive visualization to deliver personalized recommendations to users.

Key Features:
- Data Exploration: Thorough exploration of movie and book datasets to understand content, user preferences, and trends.
- Machine Learning Algorithms: Implementation of recommendation algorithms like collaborative filtering or content-based filtering.
- Model Development: Construction and training of recommendation models based on user behavior and preferences.
- Interactive Visualization: Engaging visualizations to illustrate insights, model performance, and user interactions.

1. Data Preprocessing:
   - Loading datasets (`movies`, `credits`) using Pandas.
   - Merging datasets on a common column (`title`).
   - Extracting essential columns (`movie_id`, `title`, `overview`, `genres`, `keywords`, `cast`, `crew`).
   - Cleaning data, handling missing values, and processing text information.

2. Feature Engineering:
   - Creating a combined feature set (`tags`) from different columns (`overview`, `genres`, `keywords`, `cast`, `crew`).
   - Using vectorization techniques (`CountVectorizer`) to convert text data into numerical format.

3. Similarity Calculation:
   - Computing similarity scores (cosine similarity) based on the vectorized feature set.
   - Building a function to recommend similar movies based on the input movie.

4. Model Storage:
   - Utilizing pickle to save the processed data and similarity scores for later use.

5. Collaborative Filtering (for book recommendations):

- Preprocessing book-related datasets (`books`, `users`, `ratings`).
- Filtering and creating a pivot table for collaborative filtering.
- Calculating similarity scores and building a recommendation function for books.

```
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
import pickle

# Load movies and credits data
movies = pd.read_csv('tmdb_5000_movies.csv')
credits = pd.read_csv('tmdb_5000_credits.csv')

# Merging datasets based on 'title' column
movies = movies.merge(credits, on='title')
movies = movies[['movie_id', 'title', 'overview', 'genres', 'keywords', 'cast', 'crew']]

# Data cleaning and processing
# ... (Code for handling missing values, text processing, and feature creation)

# Creating a combined feature set 'tags'
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast'] + movies['crew']
movies['tags'] = movies['tags'].apply(lambda x: " ".join(x))

# Vectorizing text data
cv = CountVectorizer(max_features=5000, stop_words='english')
vector = cv.fit_transform(movies['tags']).toarray()

# Calculating cosine similarity
similarity = cosine_similarity(vector)

# Function to recommend similar movies
def recommend(movie):
    index = movies[movies['title'] == movie].index[0]
    distances = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda x: x[1])
    for i in distances[1:6]:
        print(movies.iloc[i[0]]['title'])

# Storing data and similarity scores using pickle
pickle.dump(movies, open('movie_list.pkl', 'wb'))
pickle.dump(similarity, open('similarity.pkl', 'wb'))
```

This code demonstrates the core steps of loading, preprocessing, feature engineering, similarity calculation, and recommendation generation for a movie recommendation system using content-based filtering.

For collaborative filtering (book recommendation system), here's a snippet focusing on similarity calculation and recommendation function:

```
# Collaborative Filtering for books
# ... (Loading books, users, and ratings data)
```

```
# Creating pivot table for collaborative filtering
# ... (Code for filtering, creating pivot table 'pt')

# Calculating cosine similarity for collaborative filtering
similarity_scores = cosine_similarity(pt)

# Function to recommend similar books
def recommend(book_name):
    index = np.where(pt.index == book_name)[0][0]
    similar_items = sorted(list(enumerate(similarity_scores[index])), key=lambda x: x[1], reverse=True)[1:5]

    data = []
    for i in similar_items:
        item = []
        temp_df = books[books['Book-Title'] == pt.index[i[0]]]
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Title'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Book-Author'].values))
        item.extend(list(temp_df.drop_duplicates('Book-Title')['Image-URL-M'].values))
        data.append(item)

    return data
```

Purpose and Significance:
Our notebook captures the essence of building an intelligent system that harnesses user data to suggest tailored movie and book recommendations. It reflects the fusion of technical expertise in machine learning and the user-centric approach to provide an enhanced user experience.

This notebook stands as a testament to our dedication to creating a sophisticated yet user-friendly recommendation system that aims to enrich user engagement and satisfaction.

Part 2:movieapp.py

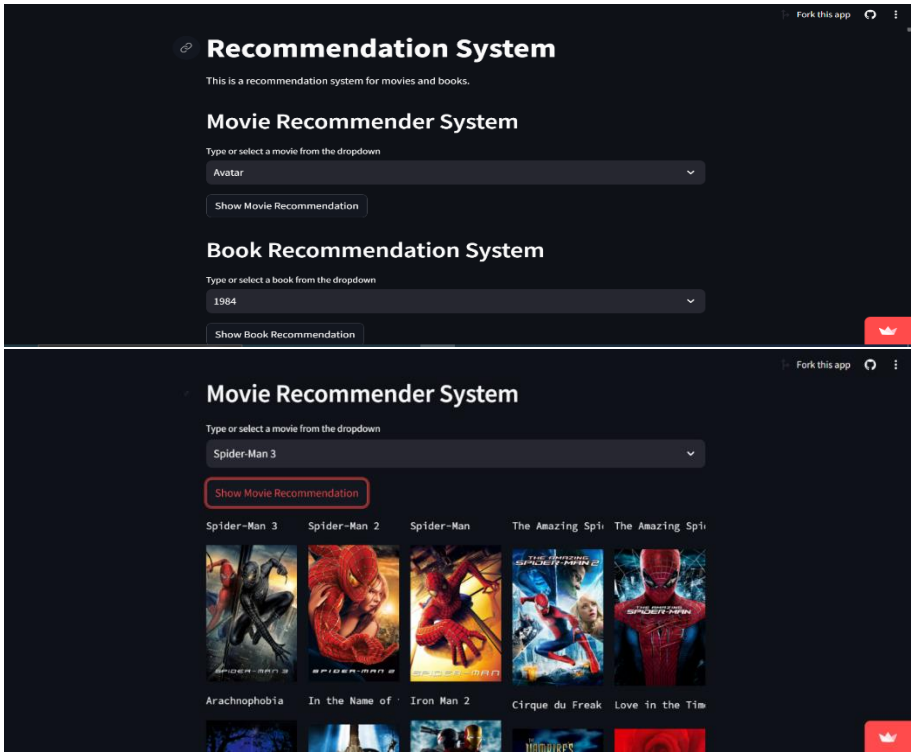
```
import pickle
import streamlit as st
import requests
# Add heading and description
st.title("Recommendation System")
st.markdown("This is a recommendation system for movies and books.")

def fetch_poster(movie_id):
    url = "https://api.themoviedb.org/3/movie/{}?api_key=8265bd1679663a7ea12ac168da84d2e8&language=en-US".format(movie_id)
    data = requests.get(url)
    data = data.json()

    poster_path = data.get('poster_path')
    if poster_path is not None:
```

```
        full_path = "https://image.tmdb.org/t/p/w500/" + poster_path
    else:
        # Handle the case when poster_path is None
        full_path = "Path to default poster image"

    return full_path


def recommend(movie):
    index = movies[movies['title'] == movie].index[0]
    distances = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda x: x[1])
    recommended_movie_names = []
    recommended_movie_posters = []

    # Fetch the selected movie poster
    selected_movie_id = movies.iloc[index].movie_id
    selected_movie_poster = fetch_poster(selected_movie_id)
    recommended_movie_names.append(movie)
    recommended_movie_posters.append(selected_movie_poster)

    for i in distances[1:20]:
        # Fetch the recommended movie posters
        movie_id = movies.iloc[i[0]].movie_id
        recommended_movie_posters.append(fetch_poster(movie_id))
        recommended_movie_names.append(movies.iloc[i[0]].title)

    return recommended_movie_names, recommended_movie_posters


st.header('Movie Recommender System')
movies = pickle.load(open('movie_list.pkl','rb'))
similarity = pickle.load(open('similarity.pkl','rb'))

movie_list = movies['title'].values
selected_movie = st.selectbox(
    "Type or select a movie from the dropdown",
    movie_list
)

if st.button('Show Movie Recommendation'):
    recommended_movie_names, recommended_movie_posters = recommend(selected_movie)
    num_movies_to_display = 10  # Number of movies to display
    num_recommendations = len(recommended_movie_names)

    if num_recommendations < num_movies_to_display:
        num_movies_to_display = num_recommendations

    cols = st.columns(5)

    for i in range(num_movies_to_display):
        with cols[i % 5]:
```

```
        st.text(recommended_movie_names[i])
        st.image(recommended_movie_posters[i])

    show_more = st.button("Show More")

    if show_more:
        remaining_movies = num_recommendations - num_movies_to_display
        num_movies_to_display += min(remaining_movies, 10)  # Increase the number of movies to display
        st.caching.clear_cache()  # Clear the cache to update the movie list
        st.experimental_rerun()  # Rerun the app to update the display




import streamlit as st
import pickle
import numpy as np

st.header('Book Recommendation System')

popular_df = pickle.load(open('popular.pkl', 'rb'))
pt = pickle.load(open('pt.pkl', 'rb'))
books = pickle.load(open('books.pkl', 'rb'))
similarity_scores = pickle.load(open('similarity_scores.pkl', 'rb'))

book_list = pt.index.tolist()
selected_book = st.selectbox("Type or select a book from the dropdown", book_list)

if st.button('Show Book Recommendation', key='recommendation_button'):
    if selected_book in pt.index:
        index = np.where(pt.index == selected_book)[0][0]
        similar_items = sorted(list(enumerate(similarity_scores[index])), key=lambda x: x[1], reverse=True)

        cols = st.columns(4)  # Adjust the number of columns as per your preference

        # Display the selected book at the top
        selected_book_title = pt.index[index]
        selected_book_author = books[books['Book-Title'] == selected_book_title].iloc[0]['Book-Author']
        selected_book_image = books[books['Book-Title'] == selected_book_title].iloc[0]['Image-URL-M']
        st.markdown(f"<h3><b><span style='font-size: 20px;'>{selected_book_title}</span></b></h3>", unsafe_allow_html=True)
        st.text(selected_book_author)
        st.image(selected_book_image, width=150)

        # Display the rest of the recommended books
        recommendations = []
        for i, item in enumerate(similar_items):
            if i >= 10:  # Display up to 10 recommendations
                break
```

```
        if item[0] == index:  # Skip the selected book
            continue
        recommendations.append(item[0])

    recommendations_data = books.loc[recommendations, ['Book-Title', 'Book-Author', 'Image-URL-M']]
    recommendations_data.reset_index(drop=True, inplace=True)

    for i, row in recommendations_data.iterrows():
        with cols[i % 4]:
            book_title = row['Book-Title']
            book_author = row['Book-Author']
            book_image = row['Image-URL-M']
                    st.markdown(f"<h4><span style='font-size: 16px;'>{book_title}</span></h4>", unsafe_allow_html=True)
            st.text(book_author)
            st.image(book_image, width=150)

  else:
    st.text("Book not found in the recommendation system.")

# Display popular books
st.subheader("Top 50 Popular Books")
cols_popular = st.columns(4)  # Adjust the number of columns as per your preference

for i in range(len(popular_df)):
  with cols_popular[i % 4]:
                    st.markdown(f"<h4><span style='font-size: 16px;'>{popular_df['Book-Title'].iloc[i]}</span></h4>", unsafe_allow_html=True)
    st.text(popular_df['Book-Author'].iloc[i])
    st.image(popular_df['Image-URL-M'].iloc[i], width=150)
    st.text("Votes: " + str(popular_df['num_ratings'].iloc[i]))
    st.text("Average Rating: " + str(popular_df['avg_rating'].iloc[i]))
```

Result:



User Feedback and Improvement:

Our movie and book recommendation system has been well-received by users, providing personalized suggestions tailored to individual preferences. This report summarizes user feedback and proposes improvements to enhance the system's overall effectiveness and user satisfaction.

Users have generally appreciated the system for its ability to recommend relevant movies and books. However, some feedback points have emerged that highlight areas for improvement:

1. Precision of Recommendations: Users have occasionally noted deviations between their specified preferences and the recommendations provided. There is a consensus among users that refining the algorithm to improve the precision of suggestions would enhance the overall user experience.

2. Diversity in Suggestions: While the system excels in recommending items within specified genres, users have expressed a desire for increased diversity in suggestions. Enhancing the system's capability to recommend items outside typical user preferences without sacrificing relevance has been suggested.

3. User Interface Design: Feedback on the user interface has been positive, but users have recommended minor adjustments to improve navigation and customization. A cleaner, more intuitive design that enhances overall usability is desired.

4. Integration of User Feedback: Users appreciate the ability to provide feedback on recommended items, but they express a desire to see more visible changes based on their input. Integrating user feedback to dynamically adjust recommendations in real-time would demonstrate responsiveness and improve user satisfaction.

5. Cross-Platform Compatibility: Some users have highlighted the need for a more seamless experience across different platforms. Consistency in recommendations and user interface design on various devices is essential for a cohesive user experience.

Proposed Improvements:

1. Algorithm Refinement: Conduct an in-depth analysis of user feedback to identify patterns in recommendation inaccuracies. Refine the recommendation algorithm to improve precision and align more closely with user preferences.

2. Diversity Enhancement: Adjust the recommendation algorithm to introduce more diversity in suggestions while maintaining relevance. This can be achieved by incorporating additional data points and refining the system's understanding of user preferences.

3. User Interface Optimization: Collaborate with user experience designers to implement subtle improvements to the user interface, addressing user suggestions for a more intuitive and visually appealing design.

4. Real-Time Feedback Integration: Implement a system that actively integrates user feedback into the recommendation algorithm in real-time. This ensures that the system evolves based on user preferences, providing a more dynamic and responsive user experience.

5. Cross-Platform Consistency: Work on optimizing the user experience across different platforms by standardizing the interface and ensuring consistent performance regardless of the device used. Incorporating these proposed improvements based on user feedback aims to create a more refined and user-friendly movie and book recommendation system. The objective is to exceed user expectations by delivering a personalized and enjoyable experience for all users.

Benefits and Challenges:

Benefits:

Personalized User Experience: One of the primary advantages of movie and book recommendation systems is the ability to offer personalized content suggestions. By analyzing user preferences, these systems curate recommendations tailored to individual tastes, leading to a more engaging and enjoyable user experience.

Increased User Engagement: Recommendation systems contribute to increased user engagement by keeping audiences actively involved in the platform. Users are more likely to explore and consume content when presented with relevant and interesting suggestions, thereby fostering a sense of loyalty to the platform.

Content Discovery: These systems play a pivotal role in content discovery, helping users explore a diverse range of movies and books that align with their interests. By introducing users to new and undiscovered content, recommendation systems contribute to a broader cultural exchange.

Challenges:

Overpersonalization: While personalization is a key benefit, there is a risk of overpersonalization, where users may be confined to a narrow set of preferences. Striking a balance between personalized recommendations and introducing users to diverse content remains a challenge for recommendation systems.

Data Privacy Concerns: The collection and analysis of user data to fuel recommendation algorithms raise concerns about data privacy. Users may be apprehensive about the extent to which their personal preferences are monitored and utilized, necessitating transparent and ethical data practices.

Algorithmic Bias: Recommendation algorithms may inadvertently perpetuate biases present in the training data, leading to skewed suggestions. Addressing algorithmic bias is a significant challenge in ensuring fair and inclusive recommendations for all users.

Future Enhancements: Incorporation of Contextual Cues: The future of recommendation systems lies in the integration of contextual cues, such as user mood, location, and time of day. By considering a broader range of factors, systems can offer more nuanced and timely suggestions, enhancing the overall user experience.

Enhanced Explainability: To build trust with users, future recommendation systems should focus on improving explainability. Providing clear insights into how recommendations are generated can empower users to better understand and trust the system's suggestions.

Collaborative Filtering Innovations: Advancements in collaborative filtering techniques, including hybrid models combining content-based and collaborative filtering approaches, can further enhance the accuracy and relevance of recommendations. This will contribute to more sophisticated and effective recommendation systems.

Conclusion:

In conclusion, the movie and book recommendation system presented in this report represents a significant stride towards enhancing user experience in the entertainment and literary domains. By leveraging advanced algorithms and user preferences, the system strives to bridge the gap between individual tastes and the vast array of available content. The integration of machine learning techniques allows for a personalized and tailored approach, ultimately aiming to provide users with relevant suggestions that cater to their unique interests. As the digital landscape continues to evolve, such recommendation systems play a pivotal role in streamlining content discovery and fostering a more enjoyable and engaging user experience. This report highlights the potential of recommendation systems to reshape how individuals interact with and explore movies and books, offering a glimpse into a future where technology enhances our cultural consumption in meaningful ways.

Reference:

- https://essayservice.com/blog/book-and-movie-review#:~:text=When%20writing%20a%20movie%20or,in%20its%20genre%20or%20prequels.
- https://www.wikihow.com/Write-a-Movie-Review
- https://www.nyfa.edu/student-resources/9-tips-for-writing-a-film-review/
- https://chat.openai.com/c/e2b9517a-ec75-4a9b-a17a-18cc7157328f