Movie Recommendation System

A Project Report Submitted
In Partial Fulfillment
for award of Bachelor of Technology

In
COMPUTER SCIENCE

By

Archit Vijay Singh(2101330120017)
Raghav Swaroop(2101330120041)
Shreyansh Pandey(2101330120051)
Prasant Srivastav(2101330120037)

Under the Supervision of
Ms. Pooja Sharma
Assistant Professor, Computer Science

Department of Computer Science
School of Computer Science and Emerging Technologies
NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY, GREATER NOIDA
(An Autonomous Institute)
Affiliated to
DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW
May, 2024

DECLARATION

We hereby declare that the work presented in this report entitled "**Movie Recommendation System**", was carried out by us. We have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute. We have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. We have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

We affirm that no portion of our work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, we shall be fully responsible and answerable.

Name : Archit Vijay Singh
Roll Number : 2101330120017

Name : Raghav Swaroop
Roll Number : 2101330120041

Name : Shreyansh Pandey
Roll Number : 2101330120051

Name : Prasant Srivastav
Roll Number : 2101330120037

CERTIFICATE

Certified that **Archit Vijay Singh** (2101330120017) has carried out the research work presented in this Project Report entitled "**Movie Recommendation System"** for the award of **Bachelor of Technology**, **Computer Science** from Dr. APJ Abdul Kalam Technical University, Lucknow under our supervision. The Project Report embodies results of original work, and studies are carried out by the students herself/himself. The contents of the Project Report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Signature

Ms Pooja Sharma

Asst. Professor
Computer Science
NIET Greater Noida

Date:

CERTIFICATE

Certified that **Raghav Swaroop** (2101330120041) has carried out the research work presented in this Project Report entitled "**Movie Recommendation System"** for the award of **Bachelor of Technology**, **Computer Science** from Dr. APJ Abdul Kalam Technical University, Lucknow under our supervision. The Project Report embodies results of original work, and studies are carried out by the students herself/himself. The contents of the Project Report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Signature

Ms Pooja Sharma

Asst. Professor
Computer Science
NIET Greater Noida

Date:

CERTIFICATE

Certified that **Shreyansh Pandey** (2101330120051) has carried out the research work presented in this Project Report entitled "**Movie Recommendation System"** for the award of **Bachelor of Technology**, **Computer Science** from Dr. APJ Abdul Kalam Technical University, Lucknow under our supervision. The Project Report embodies results of original work, and studies are carried out by the students herself/himself. The contents of the Project Report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Signature

Ms Pooja Sharma

Asst. Professor
Computer Science
NIET Greater Noida

Date:

CERTIFICATE

Certified that **Prasant Srivastav** (2101330120037) has carried out the research work presented in this Project Report entitled "**Movie Recommendation System"** for the award of **Bachelor of Technology**, **Computer Science** from Dr. APJ Abdul Kalam Technical University, Lucknow under our supervision. The Project Report embodies results of original work, and studies are carried out by the students herself/himself. The contents of the Project Report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Signature

Ms Pooja Sharma

Asst. Professor
Computer Science
NIET Greater Noida

Date:

ACKNOWLEDGEMENTS

We would like to express our sincere appreciation to Ms. Pooja Sharma for her valuable feedback and suggestions, which have significantly improved the quality of our project. We are also grateful for her diligent review of our work during its development. Furthermore, we extend our thanks to the Department of Computer Science at Noida Institute of Engineering and Technology for their support and encouragement, which inspired us to undertake this research paper. Our thanks and appreciations to respected HOD Dr. Ritesh Rastogi, for their motivation and support throughout our project.

Name of Students

Archit Vijay Singh

(2101330120017)

Raghav Swaroop

(2101330120041)

Shreyansh pandey

(2101330120051)

Prasant Srivastav

(2101330120037)

Date:

Place:

ABSTRACT

In our social lives, movie recommendations play a vital role in enhancing entertainment experiences. Movies have a unique ability to provide enjoyment and captivate audiences based on their interests and the overall popularity of the films. A well-designed recommendation system can offer users a curated selection of movies tailored to their preferences.Given the vast array of movie options available, consumers often face challenges in discovering new releases or hidden gems due to time constraints. This creates a need for efficient recommendation systems that not only assist users in choosing movies but also contribute to boosting movie sales and overall viewer satisfaction. By suggesting relevant movies, these systems bridge the gap between consumer preferences and available content.

Our study is centered on exploring a hybrid approach to recommendation systems, specifically blending content-based and collaborative filtering techniques. This innovative perspective aims to optimize movie recommendations by combining the strengths of both methods. By examining this hybrid technique, we seek to enhance the effectiveness and accuracy of movie recommendations, ultimately improving the movie-watching experience for users and supporting the film industry.A movie recommendation system is crucial in our social lives as it enhances entertainment more effectively than other forms of leisure. Such systems provide users with curated movie selections based on their interests or film popularity. They serve the purpose of suggesting products for purchase or viewing, especially important since consumers cannot keep up with all new releases due to limited time. This scenario poses challenges for the movie industry, underscoring the need for efficient movie recommendations to aid users in quickly finding enjoyable content.

This study emphasizes the importance of hybrid techniques, combining content-based and collaborative filtering approaches from a fresh perspective. By leveraging insights from extensive research and scholarly articles, we aim to enhance the effectiveness of movie recommendations and improve user experience in choosing movies to watch.

Keywords—Movie recommendation, Filtering method, Hybrid Method

TABLE OF CONTENTS

(Signature of the Candidate)
Name: Archit Vijay Singh
Enrollment No. :2101330120017

(Signature of the Candidate)
Name: Raghav Swaroop
Enrollment No. :2101330120041

(Signature of the Candidate)
Name: Shreyansh Pandey
Enrollment No. :2101330120051

(Signature of the Candidate)
Name: Prasant Srivastav
Enrollment No. :2101330120037

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Full Form
UI	User Interface
GPS	Google Positioning System
SDK	Software Development Kit
ER	Entity-Relationship
DFD	Data Flow Diagram
IFS	Information Filtering System
RS	Recommendation System
CF	Collaborative Filtering
KNN	K- Nearest Neighbour
ICF	Item Based Collaborative Filtering

CHAPTER 1

INTRODUCTION:

The recommendation system is an integral part of daily life, assisting individuals in decision-making based on their preferences and interests. Collaborative filtering models analyze a user's past purchases and ratings provided by similar users to predict their future preferences or ratings for various items. Despite the advancements in recommendation techniques, challenges like accuracy and customization persist, leading to continued use of simpler methods like search-based recommendations in many applications. Techniques such as Alternating Least Squares, Singular Value Decomposition, K-Nearest Neighbor, and Normal Predictor algorithms have been employed to address these challenges. Collaborative filtering can be memory-based, incorporating all past ratings, or model-based, using machine learning models to recommend new items based on user-item interactions. However, there is still a need for further improvement in recommendation systems to enhance accuracy and effectiveness. Movies are a popular source of entertainment, and with the proliferation of streaming services and online platforms, users often rely on recommendation systems to discover new content aligned with their preferences. The use of categorization methods and machine learning techniques plays a crucial role in organizing and recommending movies based on user preferences and behaviors, contributing to a more personalized and engaging viewing experience.

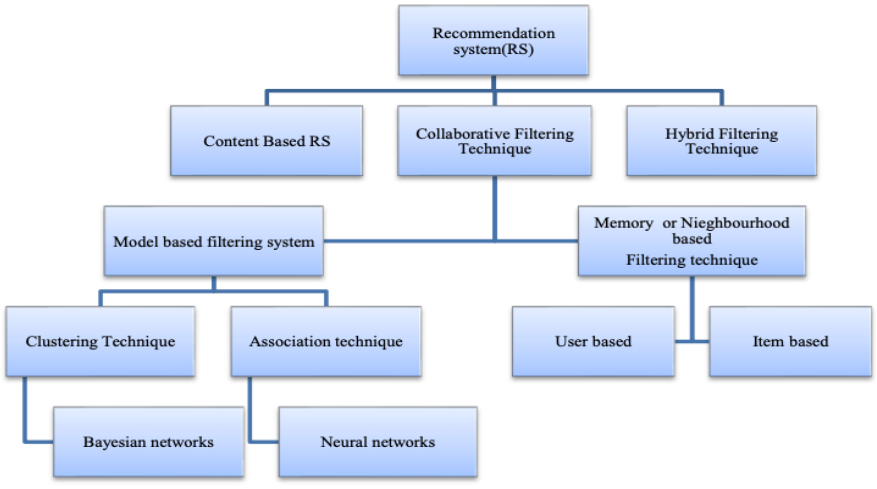


Fig. 1.0

Information filtering systems can be categorized based on two main criteria:

a. Operation initiative: This categorization differentiates between active and passive IF systems.

b. Operation location: This classification distinguishes between systems located at the information source, filtering servers, and user sites.

c. Recommendation System (RS) : The evolution of recommendation system starts as follows:

The development of recommendation systems can be outlined as follows:

1. Recommender Systems are a specialized type of information filtering method.
2. They offer technology to assist users in discovering relevant content on the internet.
3. These systems aid users in navigating through internet information to filter out irrelevant or incorrect content.
4. Recommendation Systems play a direct role in guiding users to make decisions that align with their current needs.
5. They gather information on user preferences for various items such as movies, songs, or books.
6. Recommendation Systems can be defined as efficiently and effectively navigating through information.
7. They are automated mechanisms designed to locate relevant and novel information.

Content Based Recommendation System

The content-based recommendation system suggests items by comparing the content of the items with a user's profile. It recommends items that are similar to ones the user has previously shown a preference for. This type of recommender system operates based on user-provided data, which can be explicit (like ratings) or implicit (like clicks on links). This data is used to generate a user profile, which is then utilized to make personalized suggestions to the user.

Collaborative Filtering System

Collaborative filtering (CF) is an automated method for predicting user interests by gathering preferences or taste information from multiple users in a collaborative manner within a recommendation system (RS). In collaborative filtering, there are two main types: user-based recommenders and item-based recommenders.

Hybrid Recommendation System

Recent research indicates that a hybrid approach, which combines collaborative filtering and content-based filtering, can be more effective in certain scenarios. Hybrid methods can be implemented in various

ways, such as making separate predictions using content-based and collaborative-based techniques and then combining these predictions. Another approach involves enhancing collaborative-based methods with content-based capabilities, and vice versa.

Several empirical studies have compared the performance of hybrid methods with pure collaborative and content-based methods, demonstrating that hybrid methods often offer more accurate recommendations. Additionally, these hybrid approaches can address common challenges in recommender systems, including the cold start problem and sparsity issues. A notable example of a successful hybrid recommender system is seen on Netflix. The platform generates recommendations by analyzing the viewing and searching behaviors of similar users (using collaborative filtering). It also suggests movies that share characteristics with films highly rated by a user (employing content-based filtering). This combination of techniques enhances the overall recommendation quality and user experience on the platform.

1.1. Motivation

The rapid expansion of web technologies has resulted in an overwhelming volume of information for users to navigate online, making it challenging to find content that aligns with their interests efficiently. The increasing dependence on the internet has compounded this issue, making it difficult for individuals to access relevant information tailored to their needs. To address this problem, recommendation systems have emerged, providing users with automated mechanisms to discover relevant and new information. These systems have become critical for the success of E-commerce and the IT industry, gaining widespread adoption in various applications such as YouTube, Google, and Amazon. For instance, when browsing videos on YouTube, users encounter a section dedicated to recommended videos, prompting curiosity about how this functionality operates.

1.2. Background

In the past, content-based recommendation systems were used to recommend items based solely on a user's browsing history, catering to individual users but not suitable for recommending a single item to multiple users simultaneously. Collaborative filtering emerged to address this limitation.

Collaborative filtering, in a more specific context, is an automated method of predicting user interests by gathering explicit rating or taste information from numerous users who collaborate in providing data. The underlying assumption of collaborative filtering is that if person X shares similar opinions with person Y

on one topic, they are likely to share similar opinions on other topics compared to a randomly chosen person.

For example, in a collaborative filtering recommendation system for television shows, predictions can be made about which TV show a user might like based on their existing likes or dislikes. These predictions are personalized to each user but draw insights from a collective dataset of user preferences. This approach contrasts with simpler methods that assign a generic score (e.g., average rating) to each item of interest based solely on its popularity or number of votes.

1.3. Identification of Client /Need / Relevant Contemporary issue

In the current digital landscape characterized by vast amounts of online content, users face challenges in navigating relevant information. To address this, recommendation systems have become pivotal, leveraging advanced web technologies to automate the discovery of pertinent content. These systems play a critical role in E-commerce and IT industries, with prominent applications like YouTube, Google, and Amazon incorporating recommendation features. Initially, content-based approaches used user browsing history but had limitations in recommending across diverse user profiles. Collaborative filtering (CF) subsequently emerged, leveraging collective user preferences and ratings to predict individual interests. CF operates on the principle that users with similar preferences on certain items will likely share preferences on other items. This method, encompassing user-based and item-based approaches, relies on explicit user data to generate personalized recommendations. Today, these techniques underpin personalized recommendation systems across various digital platforms, enhancing information retrieval and content delivery for users

1.4. Identification of Problem

The project's primary objective is to create a robust movie recommendation system that serves both customer-driven and business-driven goals. Customer-driven objectives involve recommending movies based on user satisfaction metrics like post-viewing ratings, while business-driven goals focus on generating recommendations that boost user engagement and prompt subsequent selections. Through an analysis of recommendation system architectures like Mosaic, Netscape, and YouTube, it was noted that content-based systems have limitations, leading to a shift towards collaborative filtering methods. Collaborative filtering, a widely implemented technique, leverages user similarities to predict and recommend items based on collective preferences. This approach has been successfully applied in various

domains such as news aggregation (Group Lens) and music recommendations (Ringo). Major platforms like Amazon use collaborative filtering alongside topic diversification to enhance scalability and recommendation accuracy. The project specifically aims to develop a recommendation system for posted materials, ensuring that users are presented with relevant content based on shared interests and topics, thereby enhancing overall user experience and satisfaction.

1.5. Identification of Tasks

The primary aim of my project is to address key challenges encountered in recommendation systems, specifically focusing on mitigating issues like cold start and sparsity through the implementation of collaborative filtering techniques. By utilizing collaborative filtering, the system will assist users in discovering items aligned with their interests, ultimately enhancing the accuracy and quality of movie recommendations. Additionally, the project aims to improve scalability to accommodate growing datasets and user bases, ultimately enhancing overall user experience. The overarching goal is to create a recommendation system that not only effectively suggests relevant movies but also optimizes user satisfaction and engagement by leveraging collaborative filtering methods. Features need to be included are:-

- **User Login:** Users can log in to the application using their registered username and password. The login functionality checks the validity of the credentials and provides appropriate error messages if the credentials are incorrect.
- **User Registration:** The signup feature allows users to create accounts, providing essential information like username, email, and password. Upon registration, users gain access to personalized movie recommendations based on their preferences and viewing history, enhancing their experience on the platform.
- **Home:** The "Home" feature in the movie recommendation project serves as the main landing page or dashboard for users. It provides personalized movie recommendations based on user preferences and viewing history. Additionally, it offers options for movie search, genre exploration, and access to trending or recommended films.
- **Rating:** The rating feature allows users to assign numerical scores or feedback to movies based on their viewing experience. This data is used by collaborative filtering algorithms to identify user preferences and recommend similar movies to enhance personalized movie suggestions in the recommendation system.
- **Recommendation:** The recommendation feature uses collaborative and content-based filtering, leveraging machine learning algorithms to suggest movies based on user preferences and viewing

history. It predicts user interests by analyzing past interactions and ratings, helping users discover relevant films amid a vast selection, enhancing their movie-watching experience.

- **Content Based Search:** The search feature in the movie recommendation project enables users to find specific movies based on titles or keywords. It enhances user experience by allowing targeted searches, assisting users in discovering movies aligned with their preferences quickly and efficiently within the recommendation system.

Use-Case model

Users can watch movies on a music streaming web application where movie information, including details about actors, directors, production, artists, action times, user ratings, and channels, is collected and stored in a database. Each movie has unique attributes such as actors, producers, directors, artists, action times, users, rating values, and channels. When users watch movies on this platform, practical data is generated and stored.The system utilizes collaborative filtering based on refined datasets to suggest movies to users. The primary function of the system is to display these recommendations using recommendation algorithms. When users access the recommendation section within the application, they are presented with important project recommendations and can select tracks based on their previous choices.Interagent provides the dataset, or "big data," through collaboration with the music streaming and downloading application. This web application collects 1 million data points daily, which Interagent then delivers to our web service. These updates are crucial for enhancing the accuracy of recommendations.Upon completion of the recommendation system project, Interagent integrates the web service into the music streaming and downloading application. This integration allows users to access our web service and receive personalized recommendations directly through the web application.

A use case represents an objective that a user aims to achieve with a system. Use cases are typically named using a verb or a verb followed by a noun phrase. The model illustrated in the figure is usually concise yet adequately descriptive to outline a user's objective. Users engage in use cases to accomplish observable goals. For example, in an online hotel reservation system, "Make reservation" is a use case because it describes what the user intends to achieve with the system. Similarly, the function of searching for a hotel on an online map is another use case reflecting the user's needs. This provides an overview of the user's goals without detailing how to achieve them. To determine the steps required to achieve a goal, we can document the scenario and interactions between the user and the system, including the main flow, exception flow, and conditional flow.

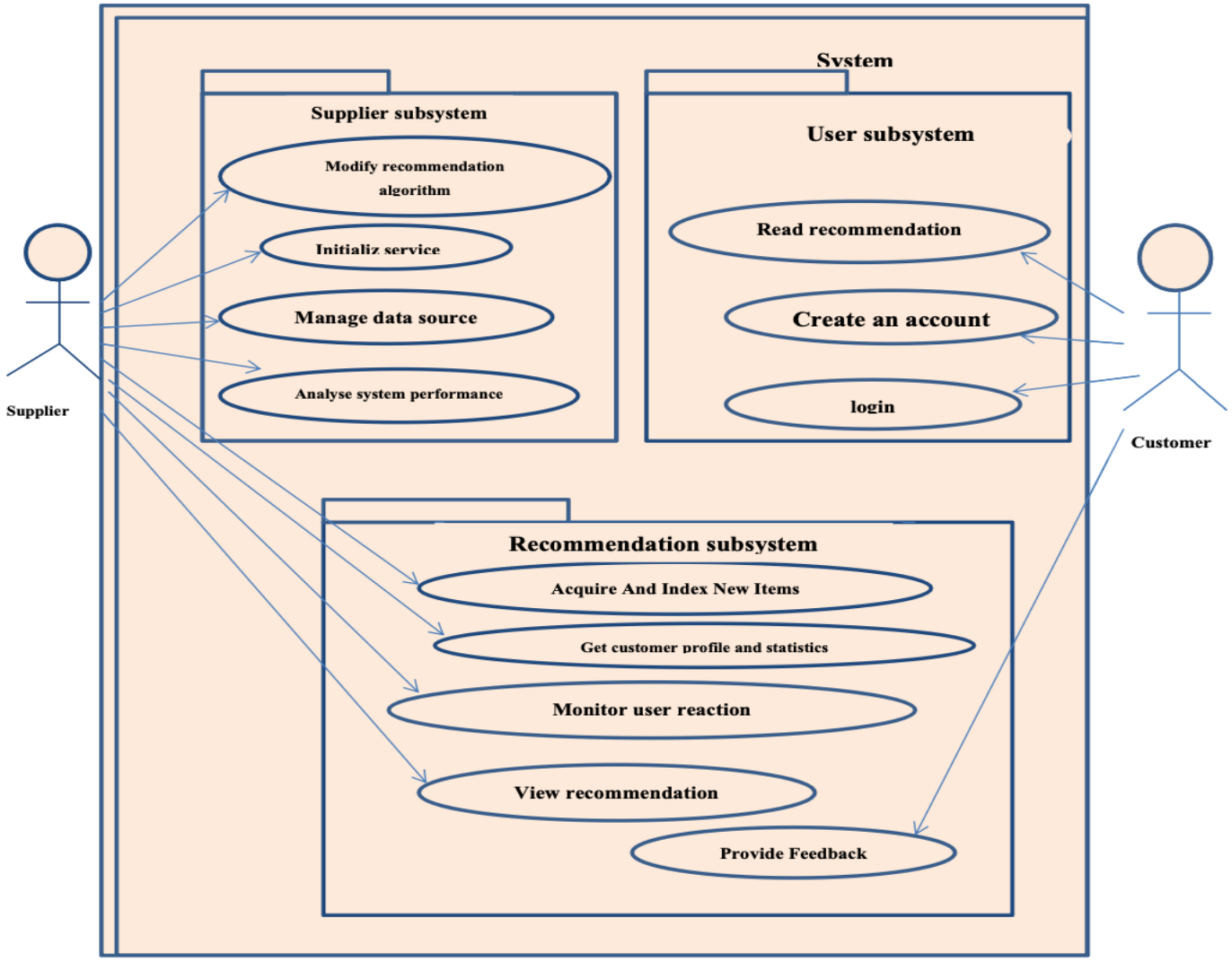


Fig. 1.2

A use case is an objective user's wants to achieve with a system. Use cases are named with verb or verb + noun phrase. The model in Fig is usually short yet description enough to describe a user objective. User performs use case to yield observable goal. Take online hotel reservation system as an example. "Make reservation" is a use case as this is what user wants to achieve with the system. The function of looking up a hotel on an online map can also be what a user needs. It is intended to provide an overview of what the user wants without knowing how to achieve the goal. To identify how to achieve a goal, we can also document its scenario and steps (i.e. interaction) involved between user and system, with main flow, exception flow, conditional flow, etc.

1.6. Phases Of Development

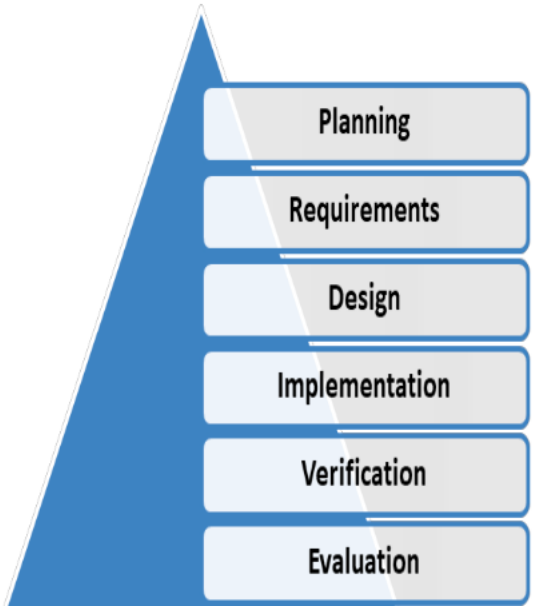


Fig. 1.3

Planning

As with most any development project, the first step is go through an initial planning stage to map out the specification documents, establish software or hardware requirements, and generally prepare for the upcoming stages of the cycle.

Requirements

In this phase, requirements are gathered from customers and check by an analyst whether requirements will fulfill or not. Analyst checks that need will achieve within budget or not. After all of this, the software team skips to the next phase.

Design

Once planning is complete, an analysis is performed to nail down the appropriate business logic, database models, and the like that will be required at this stage in the project .In the design phase, team design the software by the different diagrams like Data Flow diagram, activity diagram, class diagram, state transition diagram, etc.

Implementation

With the planning and analysis out of the way, the actual implementation and coding process can now begin. All planning, specification, and design docs up to this point are coded and implemented into this initial iteration of the project.

Verification

Once this current build iteration has been coded and implemented, the next step is to go through a series of testing procedures to identify and locate any potential bugs or issues that have cropped up.

Evaluation

Once all prior stages have been completed, it is time for a thorough evaluation of development up to this stage. This allows the entire team, as well as clients or other outside parties, to examine where the project is at, where it needs to be, what can or should change, and so on.

1.7. Timeline

Task	Estimate Start Date	Estimate Finish Date	Description	Estimate Duration
Project Proposal	29-01-2024	12-02-2024	Working and submission of project proposal	1-2 Weeks
Project Report-1	15-02-2024	04-03-2024	Brief case study on the topic and implementation	2-3 Weeks
Project Report-2	11-03-2024	08-04-2024	Testing and debugging	3 Weeks
Final Project Report	15-04-2024	13-05-2024	Submission of project	3-4 Weeks

Table. 1.4

1.8. Project Report Organization

	January				February				March				April				May			
Requirement Gathering																				
Analysis																				
Design																				
Coding																				
Testing																				
Implement																				
	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4

Table 1.5

CHAPTER 2

LITERATURE REVIEW:

Michael Fleischman et al. equated the problem of limited information about user preferences to one of content similarity measurement with the help of Natural Language Processing. They described a naïve word-space approach and a more sophisticated approach as their two algorithms and evaluated their performance compared to commercial, gold and baseline approaches [1]. A. Saranya et al. implemented a system based on probabilistic matrix factorization and provide users to solve cold start problems. Cosine similarity is used to judge similarity among users. They also implemented NB tree which is used to generate the user link formation. Their system considers various ranges of movie ratings which are given by review experts [2]. Hans Byström demonstrated a system in their paper by implementing k-means clustering and SoftMax regression classification. The baseline predictor showed an RMSE value as 0.90 and the system best achieved result was with RMSE value as 0.884 which is 1.81% improvement to baseline value [3]. Eyrun A. Eyjolfsdottir et al. introduced a system implemented using clustering and machine learning over a hybrid recommendation approach. Their system inputs user's personal information and with the help of the well-trained support vector machine (SVM) models predicts user's movies preferences. Using SVM prediction their system selects and clusters the movies from the dataset and generates questions. Then user answers those questions and their system refines its movie dataset and outputs movies recommendation for the users [4]. Arnab Kundu et al. developed a system by using collaborative filtering approach. In their system clustering is achieved among users and their preferences with the help of Expectation Maximization Algorithm [5]. Manoj Kumar et al. used collaborative filtering approach in their paper MOVREC. This approach makes use of the information provided by users. Their system then analyses the information and movies with highest rating using K-means algorithm is recommended to the user. In their system, user is provisioned to select attributes for the recommended movies [6]. Kaivan Wadia et al. implemented recommendation system based on self-organizing map (SOM). A SOM is defined as a neural network technique which comes in the domain of supervised learning. They used content based approach as well as SOM in their system for the categorization and retrieval of information [7]. Karzan Wakil et al. implemented their system which is based on human emotions. They used a hybrid approach which comprises of content based filtering, collaborative filtering and emotion detection algorithm. The output of their system provides better recommendation as it enables users to understand connec- tion between their emotional states and the recommended movies [8]. Roberto

Mirizzi et al. implemented a Facebook application that semantically recommends movies with the knowledge of user's profile. To detect similarity among movies Semantic version of Vector Space Model (SVM) is implemented by them [9]. Zan Wang et al. described improvements of the typical collaborative filtering methods in their paper. Their system uses collaborative filtering for extracting like-minded user's preferences. Then they used hybrid model of clustering which is a combi- nation of Genetic Algorithms k-means (GA-KM) and Principal Component Analysis (PCA). They implemented PCA to reduce the sparseness in the feature space and GA-KM to reach to the global optimal solution [10]. Rupali Hande et al. implemented using hybrid filtering approach in their paper MOVIEMENDER. Various techniques like clustering, similarity and classification are implemented in their system to result in better recommendation. Their system resulted with high precision and accuracy as well as reduction in MAE score [11].

2.1. Existing Solutions

In the landscape of recommendation systems, existing solutions encompass a range of methodologies aimed at addressing the challenges of personalized content delivery and user engagement. One prominent approach is collaborative filtering, which stands out as a mature and widely adopted technique. Collaborative filtering operates by leveraging similarities among users or items to make recommendations. This method considers user interactions, such as ratings or purchase histories, to identify patterns and predict preferences. Techniques like user-based collaborative filtering recommend items to a user based on the preferences of similar users, while item-based collaborative filtering suggests items based on similarities between items themselves. Collaborative filtering has been successfully implemented in diverse applications, from e-commerce platforms like Amazon to content streaming services like Netflix, demonstrating its efficacy in enhancing recommendation accuracy and user satisfaction.In addition to collaborative filtering, content-based filtering and hybrid approaches have been explored as existing solutions in recommendation systems. Content-based filtering focuses on analyzing item attributes or characteristics to make recommendations that match a user's profile or preferences. This approach is useful for suggesting items similar to those a user has interacted with previously. Hybrid approaches combine collaborative and content-based filtering to leverage the strengths of both methods, aiming to overcome individual limitations and improve recommendation quality.

2.2. Review Summary

[1] Michael Fleischman, Eduard Hovy. “Recommendations without User Preferences: A Natural Language Processing Approach”. In proceedings of the 8th international conference on Intelligent user interfaces, pages 242–244. Miami, Florida, USA, 2003.

[2] Saranya, A. Hussain. “User Genre Movie Recommendation System Using NB Tree”. In proceedings of the International Journal of Innovative Research in Science, Engineering and Technology. Vol. 4, Issue 7, July 2015.

[3] Hans Byström. “Movie Recommendations from User Ratings”. Stanford University, 2013.

[4] Eyrun A. Eyjolfsdottir, Gaurangi Tilak and Nan Li. MovieGEN: “A Movie Recommendation System”. Technical report, Computer Science Department, University of California Santa Barbara, 2010.

[5] Debadrita Roy, Arnab Kundu. “Design of movie recommendation system by means of collaborative filtering”. Int. J. Emerg. Technol. and Adv. Eng, 2013.

[6] Kumar, M., Yadav, D.K., Singh A., Gupta and V.K. “A movie recommender system: Movrec”. In International Journal of Computer Applications, ACL 2002 Conference on Empirical Methods in Natural Language Processing, vol. 124, pp. 0975–8887, 2015.

[7] Kaivan Wadia, Pulkit Gupta. “Movie Recommendation System based on Self-Organizing Maps”. The University of Texas at Austin, Austin, Texas, 2011.

[8] Karzan Wakil, Rebwar Bakhtyar, Karwan Ali and Kozhin Alaadin. “Improving Web Movie Recommender System Based on Emotions”, International Journal of Advanced Computer Science and Applications, vol. 6, no. 2, 2015.

[9] Roberto Mirizzi, Tommaso Di Noia, Azzurra Ragone. “Movie recommendation with dbpedia”. In CEUR Workshop Proceedings, vol. 835, 2012.

[10] Wang Z, Yu X, Feng N and Wang Z. “An improved collaborative movie recommendation system using computational intelligence”. J Vis Lang Comput 25:667–675, 2014.

[11] Rupali Hande, Ajinkya Gutti, Kevin Shah, Jeet Gandhi and Vrushal Kamtikar. “MOVIEMENDER - A MOVIE RECOMMENDER SYSTEM”, International Journal of Engineering Sciences & Research Technology (IJESRT) (Vol. 5, No. 11), 2016

2.3. Problem Definition

The project aims to develop an efficient movie recommendation system that aligns with either customer-driven or business-driven objectives. Customer-driven goals involve identifying movies that customers are likely to enjoy based on their post-viewing ratings, aiming to maximize user satisfaction. Alternatively, business-driven objectives focus on creating movie recommendations that encourage user engagement, ensuring that at least one recommended movie becomes the user's next viewing selection. By understanding these goals, the recommendation system can tailor its outputs to either enhance user satisfaction or drive user engagement, depending on the desired metric.

In our exploration of various recommendation systems like Mosaic, Netscape, and YouTube, we identified the limitations of content-based recommendation approaches and the transition towards collaborative filtering systems. Collaborative filtering leverages user similarity to predict or recommend items, harnessing the collective preferences of similar users. This method allows for personalized recommendations based on user behavior rather than just content features, making it more effective in addressing information overload. Collaborative filtering techniques, including GroupLens and Amazon's recommendation algorithms, have been successfully implemented across different domains to assist users in discovering relevant content.

Our current focus is on developing a recommendation system tailored to posted materials related to specific subjects. For example, if users like John and Mike both post documents on artificial intelligence with similar tags and labels, our system aims to recommend these documents to users searching for content on this topic. By leveraging tags, labels, and content similarities, our recommendation engine enhances user experience by providing relevant and personalized suggestions within a diverse collection of posted materials. This approach not only assists users in discovering content of interest but also fosters engagement and interaction within the blogging platform based on shared interests and topics.

2.4. Goals/Objectives

The primary focus of this project is to address prevalent challenges encountered in recommendation systems, particularly the issues of cold start and sparsity, through the implementation of collaborative filtering techniques. Cold start refers to the difficulty of making recommendations for new users or items

with limited historical data, while sparsity pertains to the lack of sufficient data points for accurate recommendations due to the vastness of item-user interactions. Collaborative filtering methods aim to alleviate these challenges by leveraging collective user preferences and item similarities to make personalized recommendations. By harnessing the power of collaborative filtering, this project aims to assist users in discovering items aligned with their interests, thereby enhancing the accuracy and quality of the recommendation system.

Furthermore, the project seeks to improve the overall user experience by providing relevant and engaging recommendations that cater to individual preferences. Enhancing recommendation accuracy directly contributes to the quality of the movie recommendation system, ensuring that users receive suggestions that resonate with their tastes and preferences. Scalability is also a key consideration, as the system needs to efficiently handle growing datasets and user bases without compromising performance. Ultimately, the goal is to create a recommendation system that not only helps users find items of interest but also enriches their overall experience by delivering personalized and high-quality recommendations, thus fostering user satisfaction and engagement.

CHAPTER 3

REQUIREMENTS AND ANALYSIS:

The movie recommendation system implemented using Django, Python, and SQLite leverages both content-based and collaborative filtering techniques to offer personalized movie suggestions to users. Django, a high-level Python web framework, facilitates rapid development and clean design, while SQLite, a lightweight database, stores movie data, user information, and interaction history. Python's versatility is used for backend logic and implementing machine learning algorithms.

Content-based filtering relies on movie attributes like genres, directors, actors, and descriptions to recommend similar movies to those a user has liked. The implementation begins with feature extraction, where movie data is parsed to derive relevant features. Genres, for example, are represented using one-hot encoding, and text-based features, like descriptions, are vectorized using techniques such as TF-IDF (Term Frequency-Inverse Document Frequency). Cosine similarity measures the similarity between these vectorized attributes. Thus, for a given movie a user likes, the system recommends other movies with high cosine similarity scores. In Django, this involves creating models for movies with fields for relevant features and designing views and templates to display recommended movies.

Collaborative filtering, on the other hand, bases its recommendations on user interactions like ratings or viewing history rather than the content itself. This method uses a user-item matrix, where rows represent users and columns represent movies, with cell values indicating the user's rating. Techniques like Singular Value Decomposition (SVD) factorize this matrix into lower-dimensional representations, capturing underlying patterns in user preferences and movie attributes. Neighborhood-based methods further enhance this by either recommending movies liked by similar users (user-based collaborative filtering) or suggesting movies similar to those the user has liked (item-based collaborative filtering). In Django, user interactions are stored in the database, with recommendation algorithms triggered by user actions and recommendations presented through views.

Combining both approaches results in a hybrid recommendation system, which leverages the strengths of content-based and collaborative filtering. A weighted hybrid model combines the scores from both

methods, adjusted by their respective confidences. Alternatively, a switching hybrid uses collaborative filtering when sufficient interaction data is available, defaulting to content-based filtering otherwise. Feature augmentation incorporates content-based features into the collaborative filtering matrix. In Django, data management involves setting up models for users, movies, and interactions, with SQLite handling data storage. Views capture user interactions, while recommendation views present personalized suggestions. Backend logic, implemented through Python scripts and Django management commands, runs the recommendation algorithms, with periodic tasks scheduled to update recommendations based on new data.

Evaluating the system involves performance metrics like precision and recall, which measure the accuracy and completeness of recommendations, and error metrics like Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), which compare predicted ratings to actual user ratings. User engagement metrics track interactions with recommended movies, providing insight into the system's effectiveness. User feedback mechanisms are also essential for refining algorithms and improving the system over time.

In conclusion, a movie recommendation system built with Django, Python, and SQLite integrates content-based and collaborative filtering to provide effective, personalized movie suggestions. Each method has its strengths and weaknesses, but a hybrid model often yields the best performance. Using Django's robust framework and Python's versatile libraries, developers can create a scalable and user-friendly recommendation system that enhances user experience through tailored movie recommendations.

3.1 Requirements Specification

[A]Functional Requirements

Functional requirements are product features that developers must implement to enable the users to achieve their goals. They define the basic system behavior under specific conditions.

1. **User Authentication and Account Management**:
    - Users can register, log in, and manage their accounts.
    - Personalized preferences are associated with user accounts.
2. **Data Collection and Preprocessing**:
    - Collect movie data (titles, descriptions, images, etc.) from reliable sources or APIs.

- Preprocess the data (cleaning, deduplication) before storing it in the SQLite database.

3. **Database Schema and Models**:
    - Define models for movies and related entities (e.g., amenities, genres).
    - Establish relationships (many-to-many) between movies and amenities.
    - Store movie details (name, description, image URL, price) in the database.
4. **Context-Based Filtering**:
    - Consider user context (time, location) for movie recommendations.
    - Suggest relevant movies based on context (e.g., horror during Halloween).
5. **Collaborative Filtering**:
    - Implement collaborative filtering techniques (user-based or item-based).
    - Recommend movies similar to those liked by the user.
6. **Web Interface**:
    - Create web pages for user interaction (movie search, recommendations).
    - Display movie details, images, and personalized recommendations.
7. **Testing and Optimization**:
    - Thoroughly test the system to ensure accurate recommendations.
    - Optimize database queries for efficient data retrieval.

[B]Non-Functional Requirements

A non-functional requirement is a requirement that stipulates standards that can be used to judge the process of a system. Fairly than exact performance. They compared it with functional requirements that describe exact performance or function.

1. **Scalability**:
    - Design the system to handle a growing user base.
    - Optimize performance for large datasets.
2. **Security and Privacy**:
    - Ensure data privacy and security.
    - Comply with relevant data protection laws (e.g., GDPR, CCPA).
3. **Usability and User Experience**:
    - Create an intuitive and user-friendly interface.
    - Provide clear instructions and error handling.

4. **Performance**:
    - Minimize latency in recommendation generation.
    - Optimize response times for user interactions.
5. **Maintainability**:
    - Write clean, well-documented code.
    - Plan for future enhancements and updates.
6. **Legal and Ethical Considerations**:
    - Avoid recommending inappropriate or illegal content.
    - Respect copyright and licensing agreements.
    -

[A]Software Requirements

• Python

• Numpy

• Django

[B]3.3.2 Hardware Requirements

• A PC with Windows/Linux OS

• Processor with 1.7-2.4gHz speed

• Minimum of 8gb RAM

3.2 Design Flow

The design flow for the movie recommendation system using Python, Django, and SQLite with a combination of content-based and collaborative filtering involves several key steps. First, the system will gather and preprocess movie data, extracting features such as genre, cast, and director for content-based recommendations. Next, it will build a user-item interaction matrix based on user ratings and preferences to enable collaborative filtering. The system will then implement collaborative filtering algorithms like k-Nearest Neighbors or matrix factorization to generate movie recommendations by leveraging user similarities. These recommendations will be integrated into a Django web application where users can register, rate movies, and receive personalized recommendations based on their preferences and

similarities with other users. Finally, the system will use SQLite as the relational database to store user profiles, movie data, and interaction matrices for efficient data management and retrieval within the Django application.

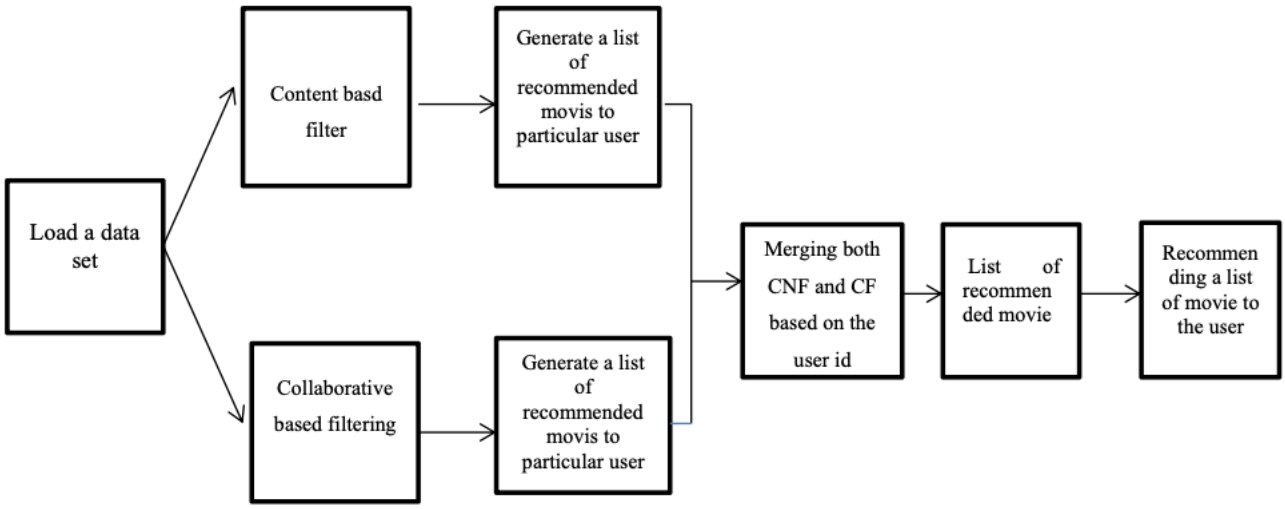


Fig. 3.1

Initially load the data sets that are required to build a model the data set that are required in this project are movies.csv, ratinfg.csv, users.csv all the data sets are available in the movielens dataset. Basically, two models are built in this work content based and collaborative filtering each produce a list of movies to a particular user by combining both based on the useid a single final list of movies are recommended to the particular user.

3.3 Preliminary Product Description

Description for Movie Recommendation System

The recommendation system is an integral part of daily life, where individuals rely on information to make decisions about their preferences and interests. Collaborative filtering models analyze a user's previous interactions and incorporate judgments from other users with similar behaviors to predict preferences or

ratings. Despite existing approaches, challenges persist in achieving accurate recommendations. Methods such as Alternating Least Squares, Singular Value Decomposition, K-Nearest Neighbor, and Normal Predictor algorithms have been explored to address these challenges.

Two main types of collaborative filtering methods are memory-based and model-based. Memory-based methods leverage all past ratings directly, ensuring real-time updates, while model-based methods, like neural networks, develop models from user-item data to recommend new items. To enhance recommendation systems, continued improvements are needed to meet user needs and deliver relevant product suggestions.

Movies are a popular form of entertainment, but individuals often struggle to keep up with new releases due to time or financial constraints. Online platforms, including streaming services and video-on-demand apps, provide convenient access to movies. Machine learning techniques, such as data organization and classification, play a key role in categorizing and recommending movies based on user preferences.

In summary, recommendation systems play a crucial role in satisfying consumer needs for personalized suggestions, especially in the context of movie preferences and entertainment choices, where technology enables easy access to a wide range of content.

The hybrid Content-Based and Collaborative Filtering movie recommendation system integrates two distinct approaches to enhance recommendation accuracy and diversity.

Content-BasedFiltering:

Utilizes item (movie) attributes like genre, keywords, cast, and director to identify similarities between movies.

Computes similarity scores using the Cosine Similarity algorithm, which measures the angle between item feature vectors to determine how alike movies are.

Recommends movies based on a user's historical preferences and likes, focusing on content similarities with previously liked items.

Provides personalized recommendations aligned with specific user preferences but may lack diversity as it predominantly suggests similar content.

CollaborativeFiltering:

Incorporates user interactions and preferences to recommend movies based on similarities between users.

Implements User-to-User Collaborative Filtering, identifying users with similar movie preferences and predicting ratings for unrated movies based on similar users' ratings.

Employs algorithms like K Nearest Neighbors (KNN) to calculate similarities between users and predict movie ratings.

Handles user biases by adjusting ratings to account for individual user tendencies, improving prediction accuracy.

Hybrid Approach:

Combines Content-Based and Collaborative Filtering techniques to leverage both item attributes and user preferences for more accurate and diverse recommendations.

Enhances recommendation quality by considering movie content similarities and user behavior patterns simultaneously.

Both content-based and collaborative filtering approaches have inherent limitations. Content-based filtering often results in limited diversity, as it primarily focuses on items similar to those the user has previously liked. On the other hand, collaborative filtering depends heavily on having a sufficient number of user-item interactions, and it can face challenges with data sparsity, where there may not be enough information to make accurate recommendations.

Offers personalized recommendations that align with user tastes while introducing variety by incorporating insights from similar users' preferences.

Modules and their Description

- **Login:**
  Login to the application using his ID and Password.
- **SignUp:**
  If user is a new user, he will enter his personal details and he will user Id and password through which he can login to the application.
- **Search:**
  The search module allows users to find movies within the recommendation system.
- **Rating:**
  The rating module enables users to rate movies, providing valuable feedback to enhance the accuracy of recommendations.
- **Recommendation:**
  The Recommend module suggests personalized movie selections using hybrid content-based and collaborative filtering algorithms for users.
- **Home:**
  The Home module displays personalized movie recommendations and trending titles for users upon system access.
- **Movie Details:**
- The Movie Details module displays comprehensive information about selected movies, including title, genre, ratings, and cast details.

CHAPTER 4

PROPOSED METHODOLOGY:

In this chapter, we explore the development of Movie Recommendation Systems using different approaches aimed at enhancing recommendation quality progressively. Initially, we delve into the Content-Based Recommendation System, which operates without leveraging other users' preferences. This method recommends items based solely on user preferences, focusing on similarities in content such as genre, keywords, cast, or director. A key algorithm used in this context is Cosine Similarity, which measures the similarity between items by calculating the cosine of the angle between their feature vectors. This approach is effective for recommending similar items based on specific user preferences but lacks diversity in recommendations.

To address the limitations of content-based systems, we then move to Collaborative Filtering Recommendation Systems. This technique leverages user interactions and preferences to recommend items, particularly focusing on the User-to-User Collaborative Filtering approach. This method identifies similar users based on shared item interactions and predicts ratings for items a user has not yet seen based on similar users' ratings. Algorithms such as K Nearest Neighbors (KNN) and Matrix Factorization are employed in Collaborative Filtering. KNN calculates similarities between users and predicts ratings based on weighted averages, while Matrix Factorization addresses sparsity issues in rating matrices by decomposing them into low-dimensional matrices representing latent features.

Moreover, the chapter discusses strategies to mitigate Cold Start Problems in recommendation systems. For new users, meta-information such as location, age, gender, browser, and device can be used to make initial recommendations. Similarly, for new movies lacking user interactions, meta-information such as genre, cast, and crew can guide recommendations to target users. These strategies aim to improve the system's ability to provide meaningful recommendations even in scenarios with limited user data or new items, enhancing the overall effectiveness and user experience of Movie Recommendation Systems

4.1 Algorithm Used

I. K- NEAREST NEIGHBORS

II. COSINE SIMILARITY

K- NEAREST NEIGHBORS

The standard method of Collaborative Filtering, known as the Nearest Neighbor algorithm, involves working with an n × m matrix of ratings, where each row represents a user (u) and each column represents an item (p). The goal is to predict the rating (r) for a target user (i) on an item (j) they have not yet rated. This is done by first calculating the similarities between the target user (i) and all other users, selecting the top X similar users based on these similarities, and then taking a weighted average of their ratings for the item (j), using the similarities as weights.

One challenge in collaborative filtering is that users have different rating baselines. Some users may tend to give higher ratings overall, while others might be more conservative in their ratings despite their satisfaction with the items. To address this bias, we adjust the ratings by subtracting each user's average rating across all items before computing the weighted average, and then add back the target user's average rating.

In the context of the K-Nearest Neighbors (KNN) algorithm:

1. Choose a value for k, representing the number of nearest neighbors to consider.

2. Calculate the distance between the unknown data point and all training examples in the feature space.

3. Identify the k training examples closest to the unknown data point based on distance.

4. Determine the nearest neighbor by selecting the training data point with the smallest distance to the unknown point.

In KNN-regression, the approach is similar, but instead of classifying based on nearest neighbors, the prediction is made by averaging the target values of the nearest training data points. This method predicts the result based on the average of the total sum of distances or similarities between the training points and the sample points.

COSINE SIMILARITY

We used the cosine similarity algorithm to determine how similar content is between two items. The dot product of two vectors is calculated as the projection of one vector onto the other. For identical vectors,

the dot product equals the squared magnitude of the vectors. Conversely, if the vectors have no shared direction, the dot product will be zero.

The dot product plays a crucial role in defining similarity because it directly influences the similarity metric. The similarity between two vectors u and v is defined as the ratio of their dot product to the product of their magnitudes. This ratio will be 1 if the vectors are identical (pointing in the same direction), and it will be 0 if the vectors are orthogonal (perpendicular, indicating no similarity).

	Item 1	Item 2	Item 3	Item 4
User A	4	?	3	5
User B	?	5	4	?
User C	5	4	2	?
User D	2	4	?	3
User E	3	4	5	?

Table 4.0

4.2 Feasibility Study

The proposed movie recommendation system aims to combine context-based filtering and collaborative filtering to provide personalized movie suggestions. Leveraging Python, Django, and SQLite, the system will collect and preprocess movie data, considering user context (such as time and location) for recommendations. Collaborative filtering techniques will be used to suggest movies based on user behavior and preferences. The project's viability includes scalability, market demand, monetization strategies, and legal considerations. Continuous improvement based on user feedback and market trends is essential for success.

System Architecture Of Proposed System

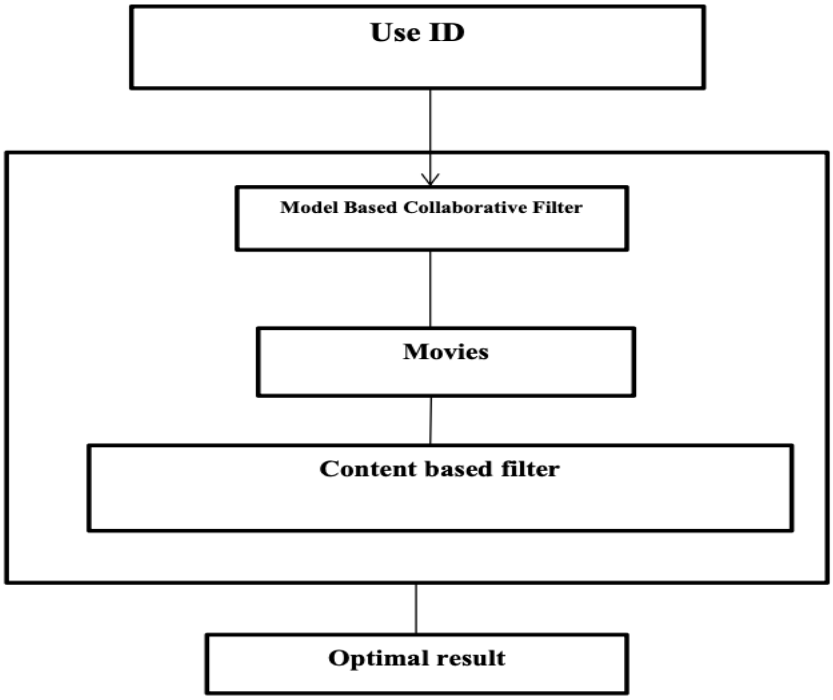


Fig. 4.1

Each user receives personalized movie recommendations upon logging in or entering their user ID. This is achieved through two distinct approaches implemented in the project. Each approach suggests a specific set of movies for the individual user. The hybrid model combines these two sets of movie recommendations based on the user's preferences, resulting in a unified list of recommended movies tailored to each user.

DataSet

We utilized the widely recognized MovieLens dataset (available at http://grouplens.org/datasets/movielens/100k/) to evaluate our proposed system's performance. This dataset comprises 100,000 ratings given by 943 users for 1,682 movies, with ratings ranging from 1 to 5. We obtained the dataset online and divided it into 80% for training and 20% for testing to validate our system's effectiveness. In our project, we utilized the open-source MovieLens dataset provided by GroupLens. This dataset contains 100,000 data points representing various movies and users, enabling us to conduct comprehensive analyses and experiments for our system. The movies in this dataset are categorized into 19 genres, including action, animation, horror, comedy, and more.

We will use three columns from the data:

• Userid

• Movieid

• Rating

	userId	movieId	rating
0	1	1	4.0
1	1	3	4.0
2	1	6	4.0
3	1	47	5.0
4	1	50	5.0
...	...	...	...
100831	610	166534	4.0
100832	610	168248	5.0
100833	610	168250	5.0
100834	610	168252	5.0
100835	610	170875	3.0

Fig. 4.2

ER Diagram

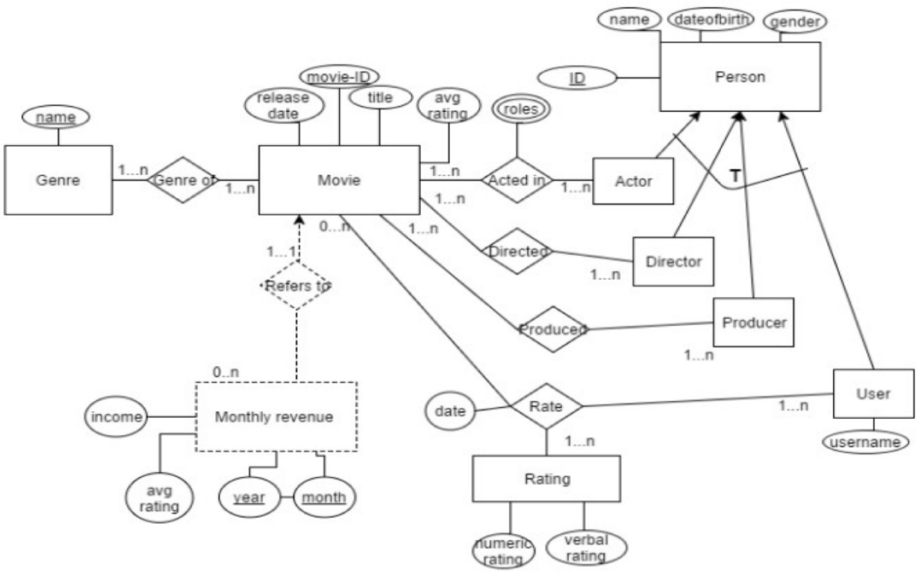


Fig. 4.3

Sequence Diagram

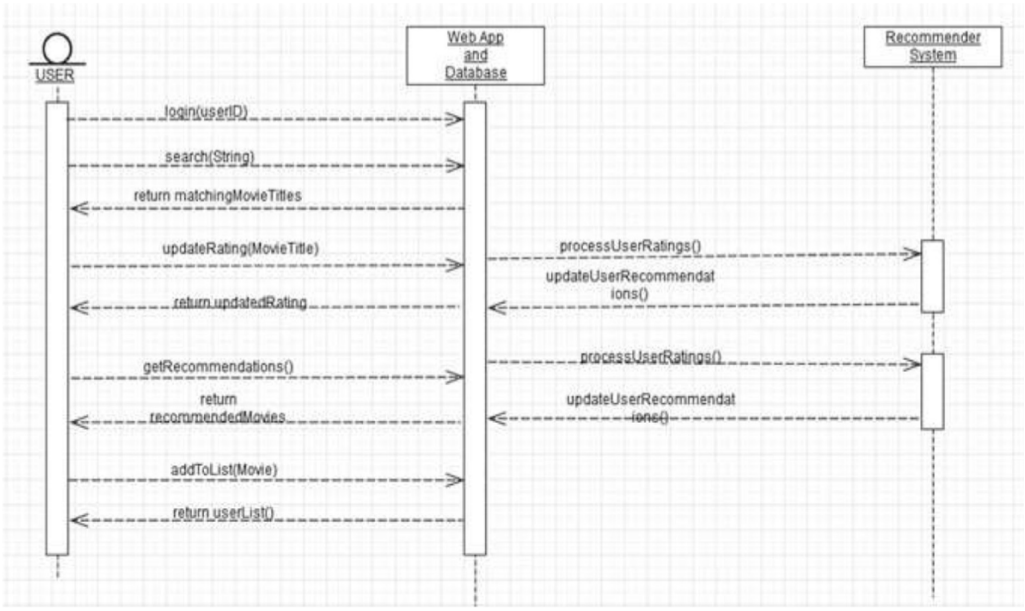


Fig. 4.4

DFD (Data Flow Diagram)

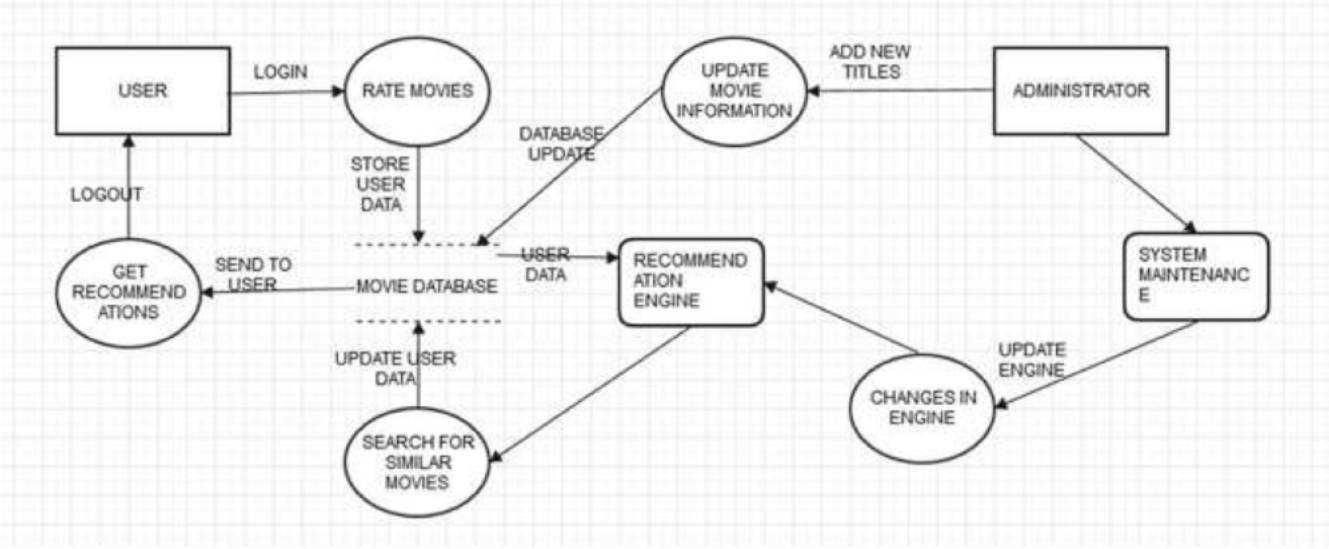


Fig. 4.4

CF assign weight

	High	Low	Moderate
High	1	3.9	5.6
Low	0.2564	1	3.2
Moderate	0.1785	0.3125	1
	1.435	5.213	9.800

Table. 4.6

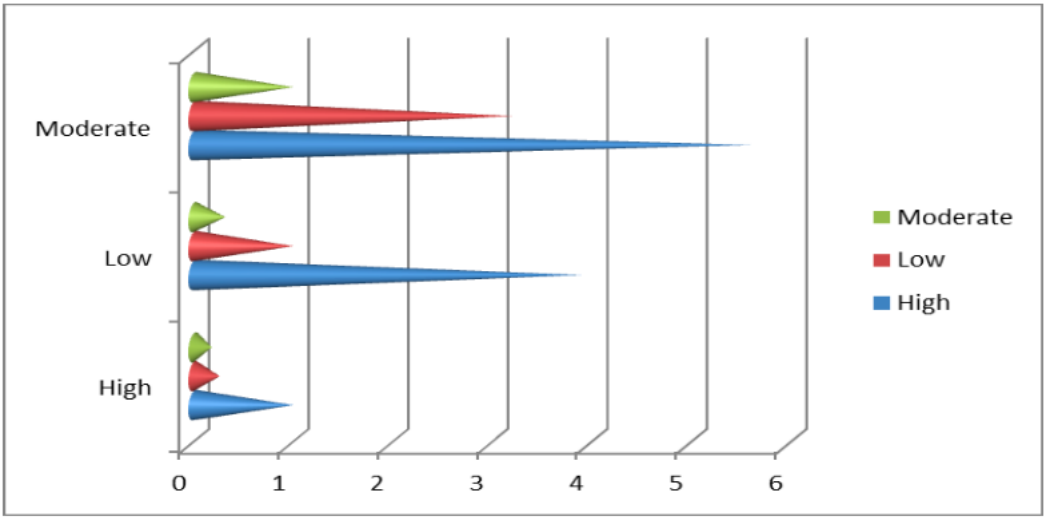
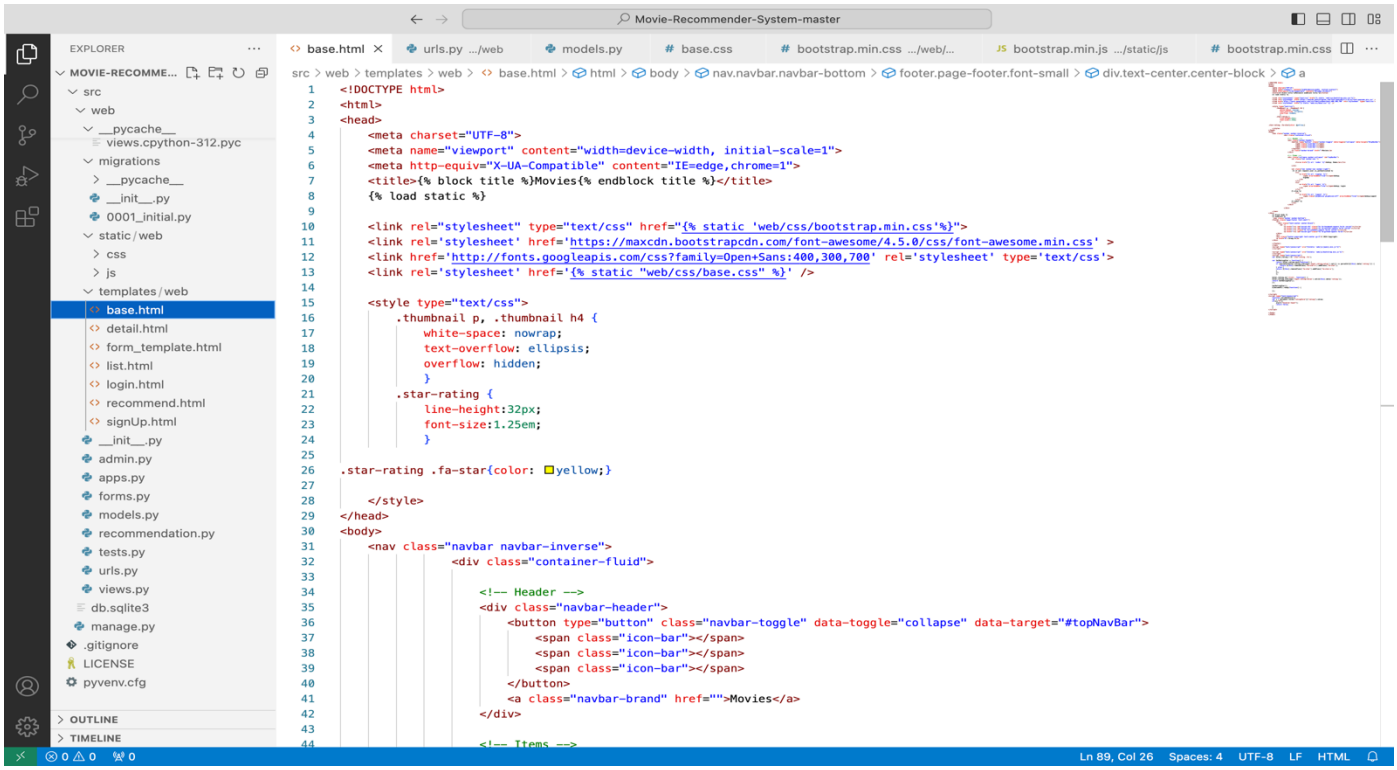
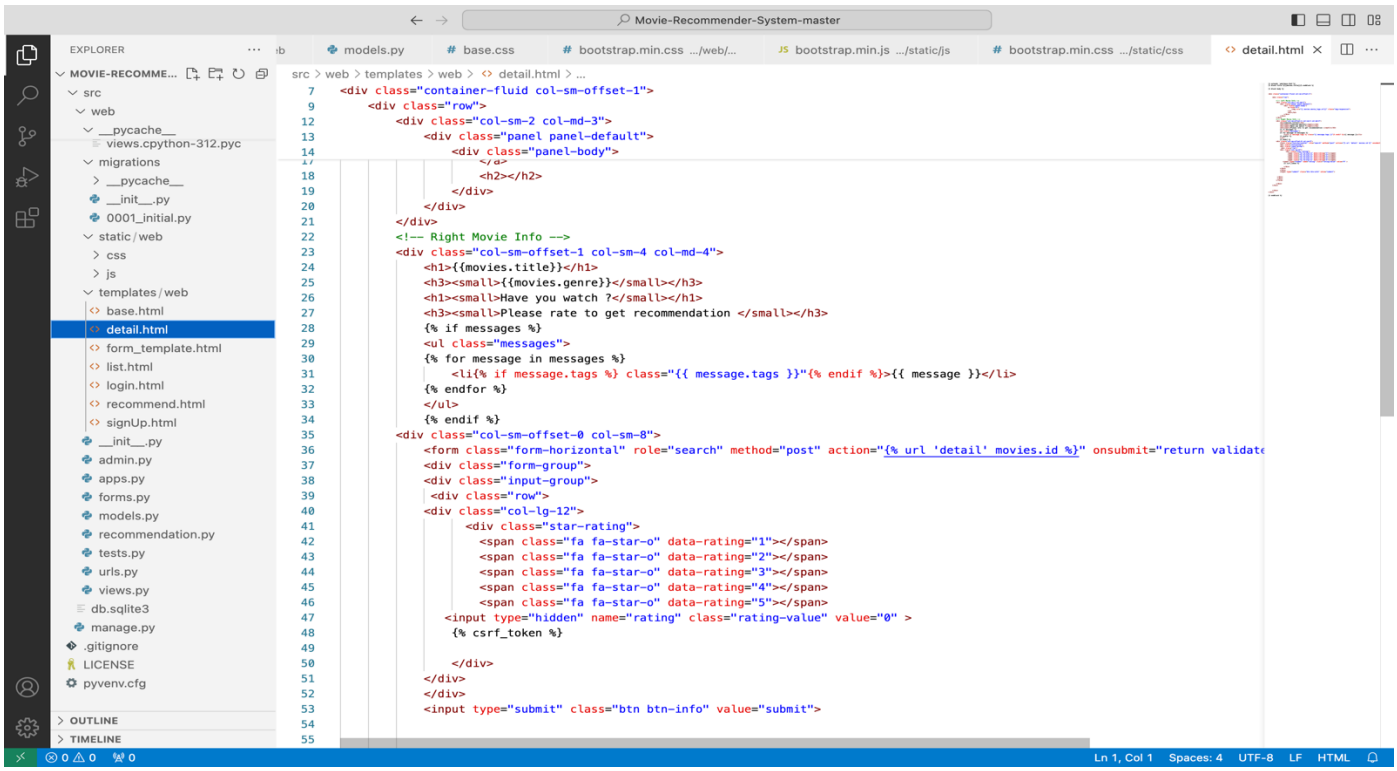
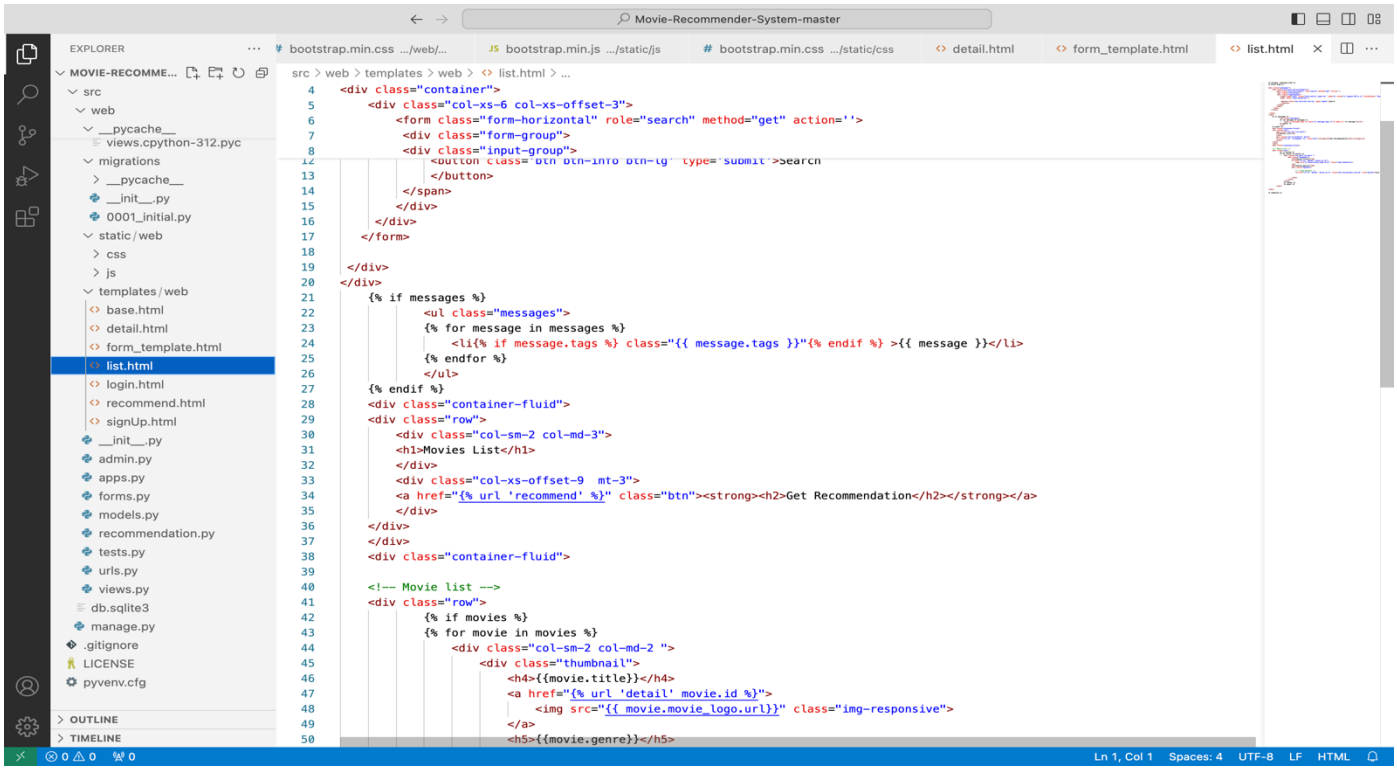


Fig. 4.7

Implemented Code






Movie-Recommender-System-master
EXPLORER
bootstrap.min.css .../web/...
bootstrap.min.js .../static/js
bootstrap.min.css .../static/css
detail.html
form_template.html
list.html
src > web > templates > web > list.html > ...
MOVIE-RECOMME...
src
web
__pycache__
views.cpython-312.pyc
migrations
__pycache__
__init__.py
0001_initial.py
static / web
css
js
templates / web
base.html
detail.html
form_template.html
list.html
login.html
recommend.html
signUp.html
__init__.py
admin.py
apps.py
forms.py
models.py
recommendation.py
tests.py
urls.py
views.py
db.sqlite3
manage.py
.gitignore
LICENSE
pyvenv.cfg
OUTLINE
TIMELINE

<form class="form-horizontal" role="search" method="get" action=''>

<button class="btn btn-info btn-lg" type="submit">search
</button>

</form>

{% if messages %}
<ul class="messages">
{% for message in messages %}
<li{% if message.tags %} class="{{ message.tags }}"{% endif %} >{{ message }}
{% endfor %}

{% endif %}

<h1>Movies List</h1>

<h2>Get Recommendation</h2>

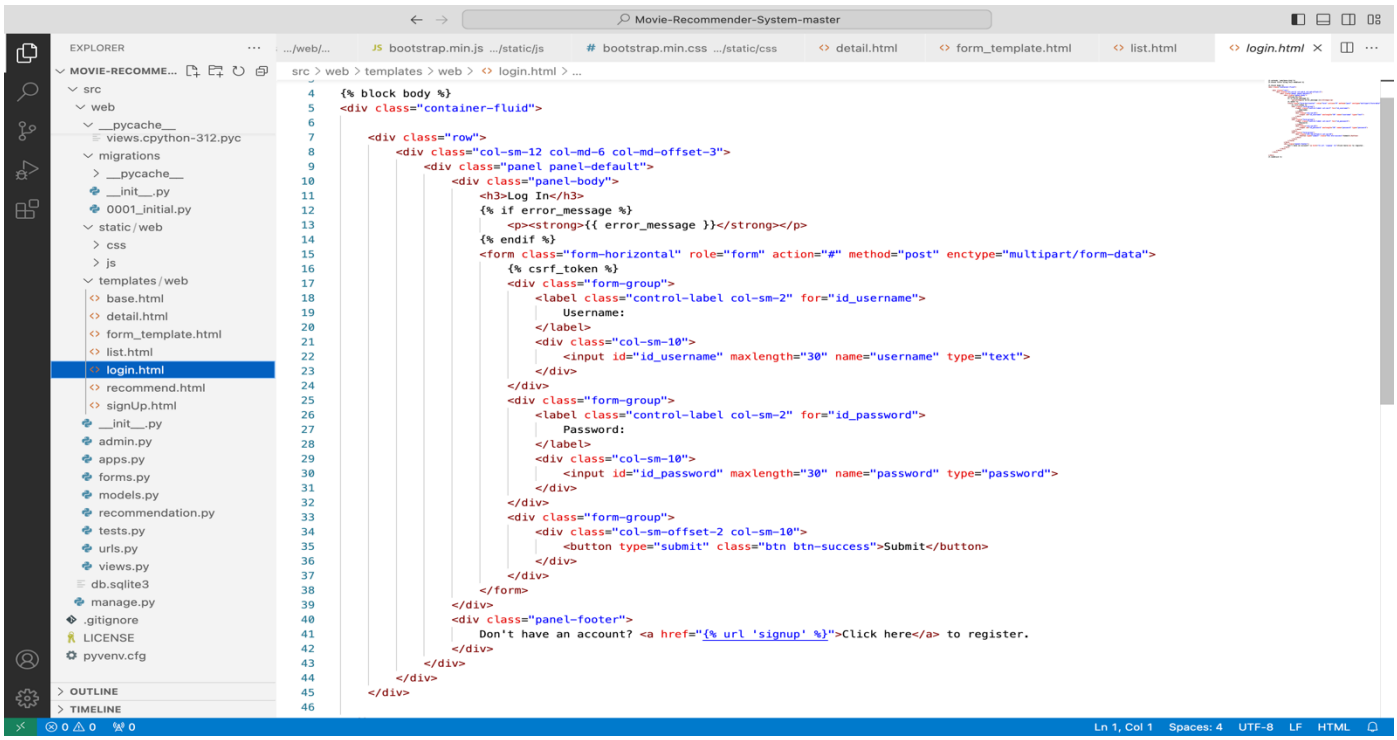
<!-- Movie list -->

{% if movies %}
{% for movie in movies %}

<h4>{{movie.title}}</h4>

<img src="{{ movie.movie_logo.url}}" class="img-responsive">

<h5>{{movie.genre}}</h5>
Ln 1, Col 1
Spaces: 4
UTF-8
LF
HTML


Movie-Recommender-System-master
EXPLORER
login.html
src > web > templates > web > login.html > ...
MOVIE-RECOMME...
{% block body %}

<h3>Log In</h3>
{% if error_message %}
{{ error_message }}
{% endif %}
<form class="form-horizontal" role="form" action="#" method="post" enctype="multipart/form-data">
{% csrf_token %}

<label class="control-label col-sm-2" for="id_username">
Username:
</label>

<input id="id_username" maxlength="30" name="username" type="text">

<label class="control-label col-sm-2" for="id_password">
Password:
</label>

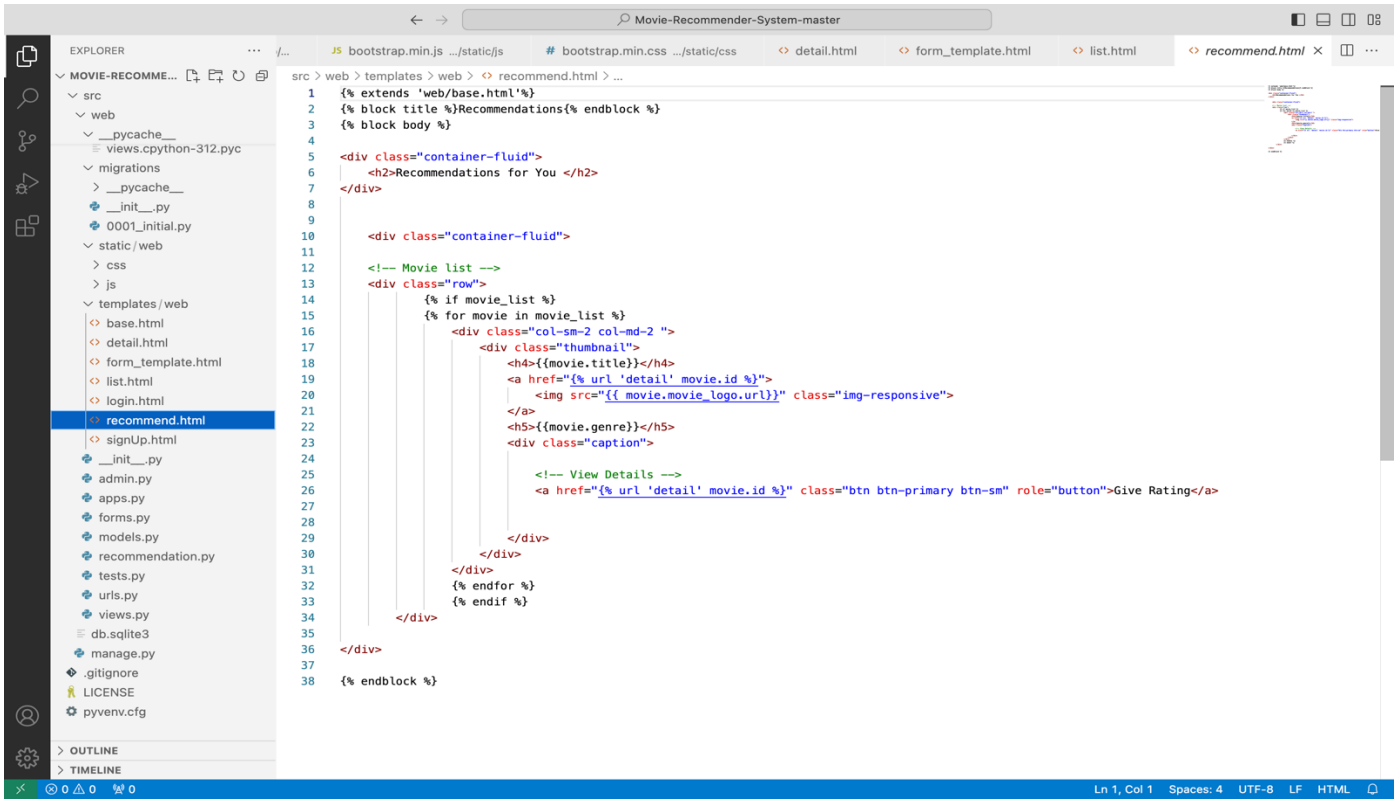
<input id="id_password" maxlength="30" name="password" type="password">

<button type="submit" class="btn btn-success">Submit</button>

</form>

Don't have an account? Click here to register.

Ln 1, Col 1
Spaces: 4
UTF-8
LF
HTML


Movie-Recommender-System-master
EXPLORER
MOVIE-RECOMME...
src
web
__pycache__
views.cpython-312.pyc
migrations
__pycache__
__init__.py
0001_initial.py
static/web
css
js
templates/web
base.html
detail.html
form_template.html
list.html
login.html
recommend.html
signUp.html
__init__.py
admin.py
apps.py
forms.py
models.py
recommendation.py
tests.py
urls.py
views.py
db.sqlite3
manage.py
.gitignore
LICENSE
pyvenv.cfg
OUTLINE
TIMELINE
bootstrap.min.js .../static/js
bootstrap.min.css .../static/css
detail.html
form_template.html
list.html
recommend.html
src > web > templates > web > recommend.html > ...
{% extends 'web/base.html'%}
{% block title %}Recommendations{% endblock %}
{% block body %}

<h2>Recommendations for You </h2>

<!-- Movie list -->

{% if movie_list %}
{% for movie in movie_list %}

<h4>{{movie.title}}</h4>

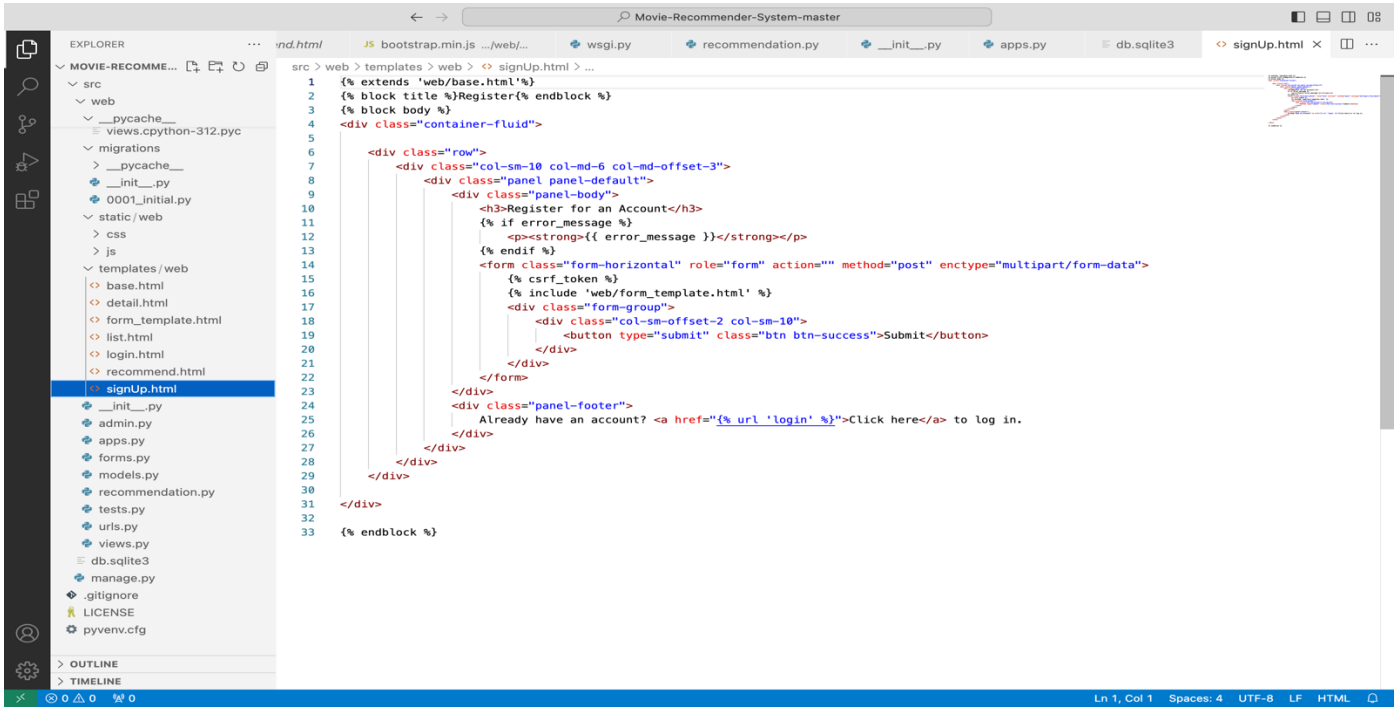
<img src="{{ movie.movie_logo.url}}" class="img-responsive">

<h5>{{movie.genre}}</h5>

<!-- View Details -->
Give Rating

{% endfor %}
{% endif %}

{% endblock %}
Ln 1, Col 1
Spaces: 4
UTF-8
LF
HTML


Movie-Recommender-System-master
EXPLORER
MOVIE-RECOMME...
src
web
__pycache__
views.cpython-312.pyc
migrations
__pycache__
__init__.py
0001_initial.py
static/web
css
js
templates/web
base.html
detail.html
form_template.html
list.html
login.html
recommend.html
signUp.html
__init__.py
admin.py
apps.py
forms.py
models.py
recommendation.py
tests.py
urls.py
views.py
db.sqlite3
manage.py
.gitignore
LICENSE
pyvenv.cfg
OUTLINE
TIMELINE
bootstrap.min.js .../web/...
wsgi.py
recommendation.py
__init__.py
apps.py
db.sqlite3
signUp.html
src > web > templates > web > signUp.html > ...
{% extends 'web/base.html'%}
{% block title %}Register{% endblock %}
{% block body %}

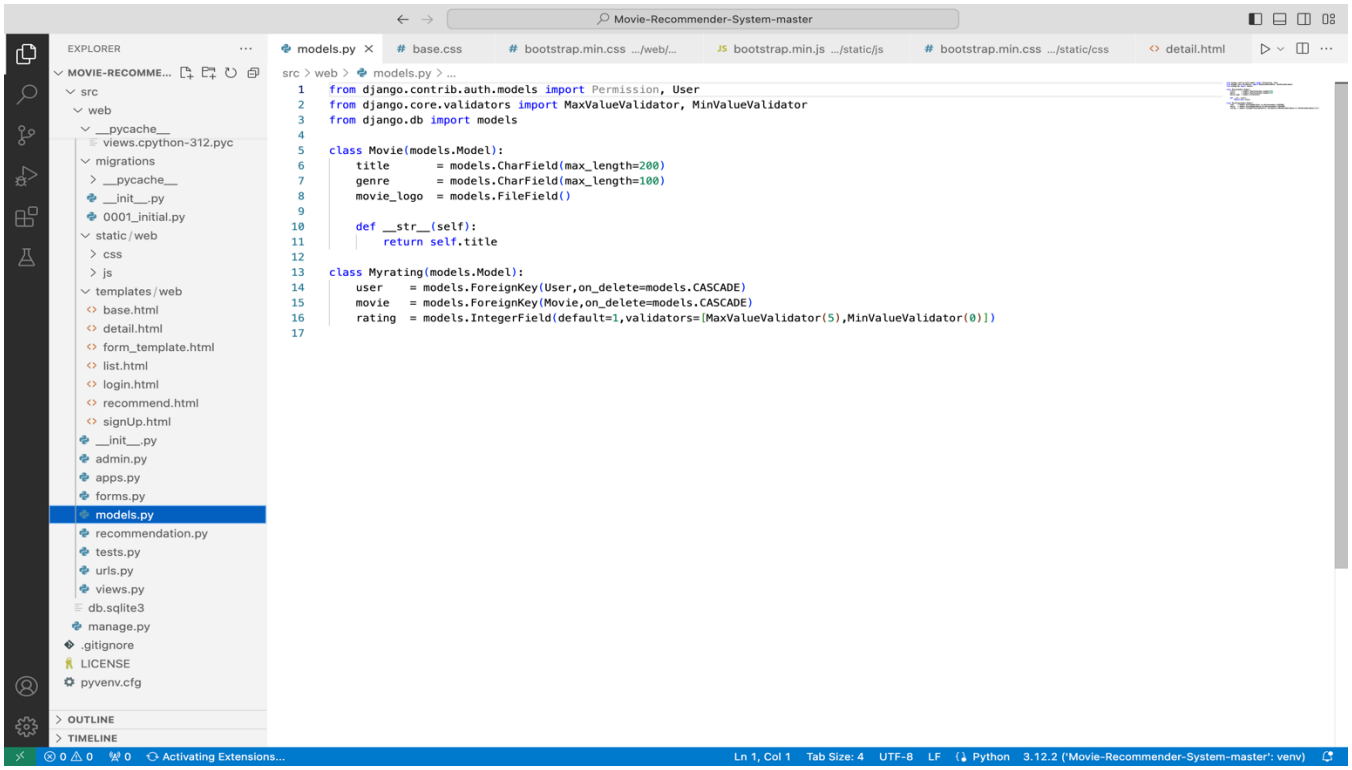
<h3>Register for an Account</h3>
{% if error_message %}
{{ error_message }}
{% endif %}
<form class="form-horizontal" role="form" action="" method="post" enctype="multipart/form-data">
{% csrf_token %}
{% include 'web/form_template.html' %}

<button type="submit" class="btn btn-success">Submit</button>

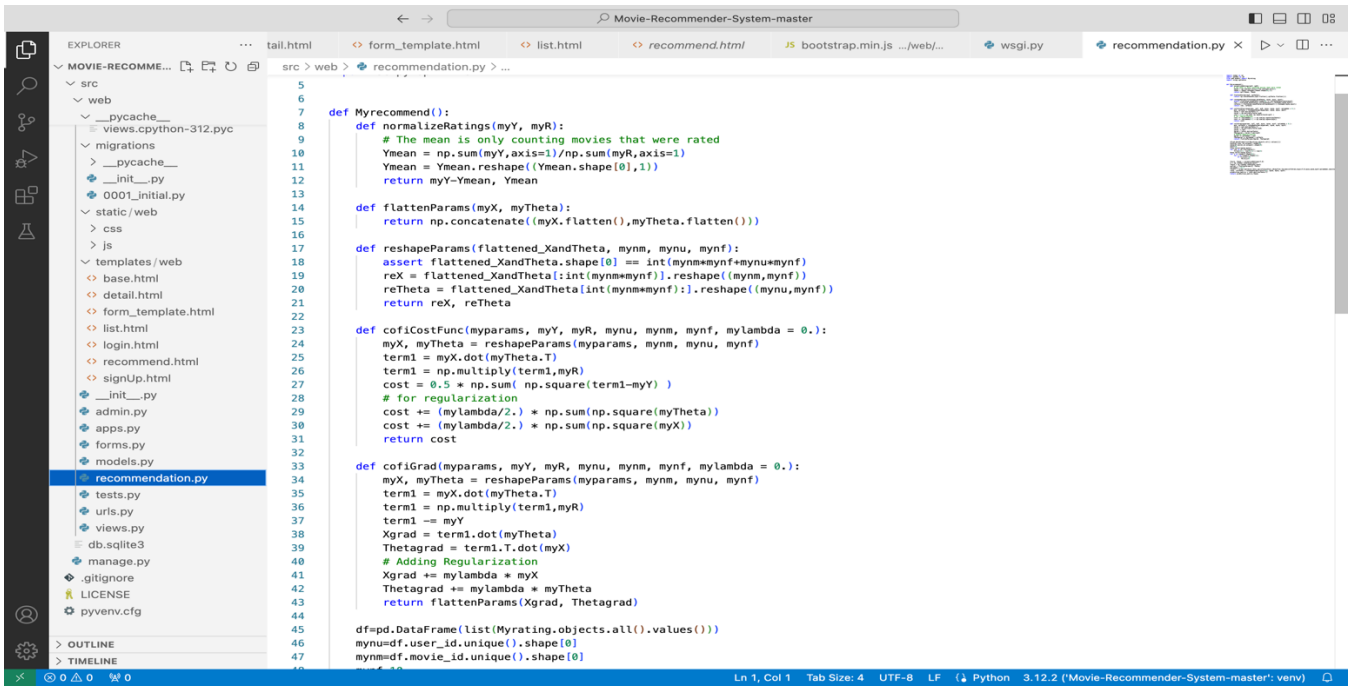
</form>

Already have an account? Click here to log in.

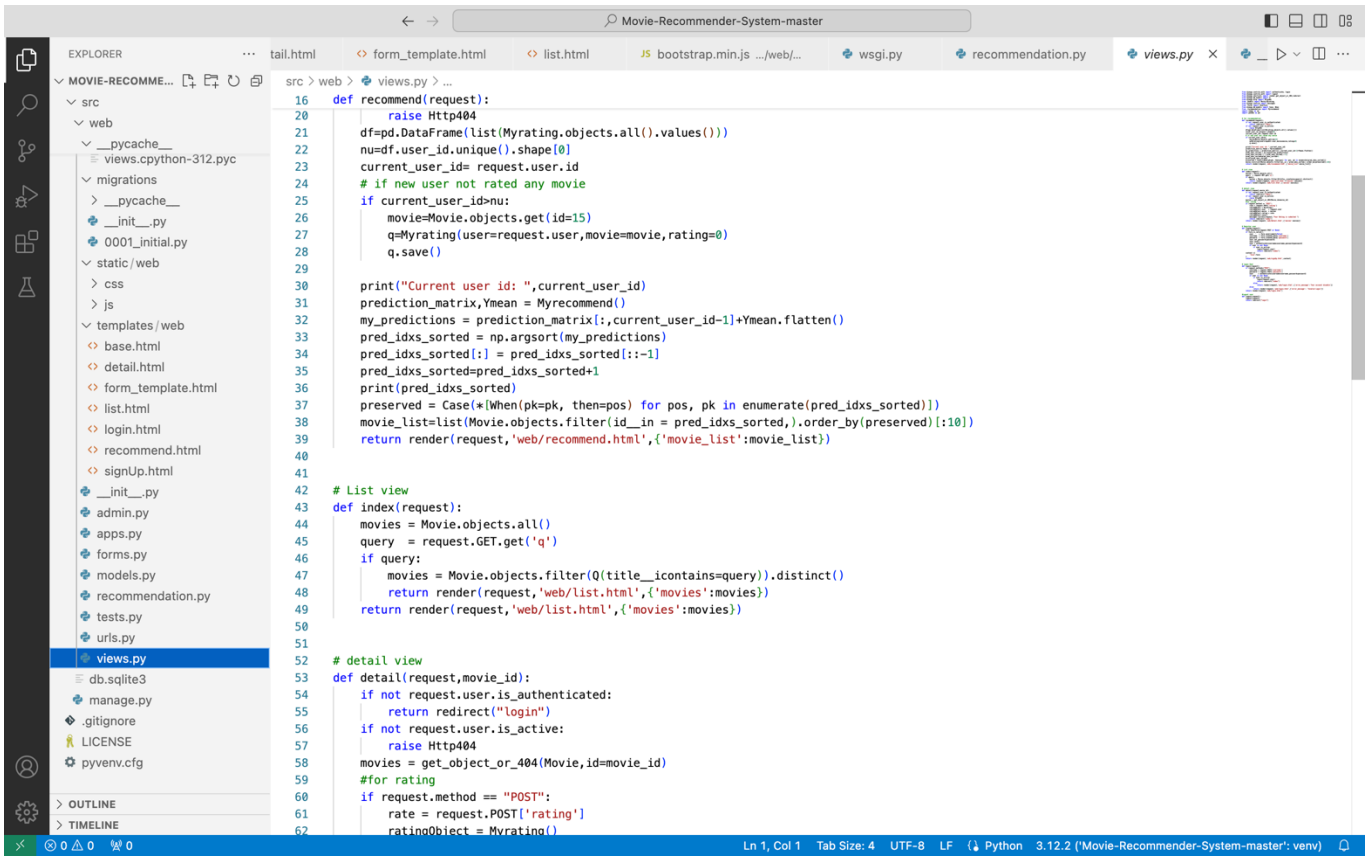
{% endblock %}
Ln 1, Col 1
Spaces: 4
UTF-8
LF
HTML

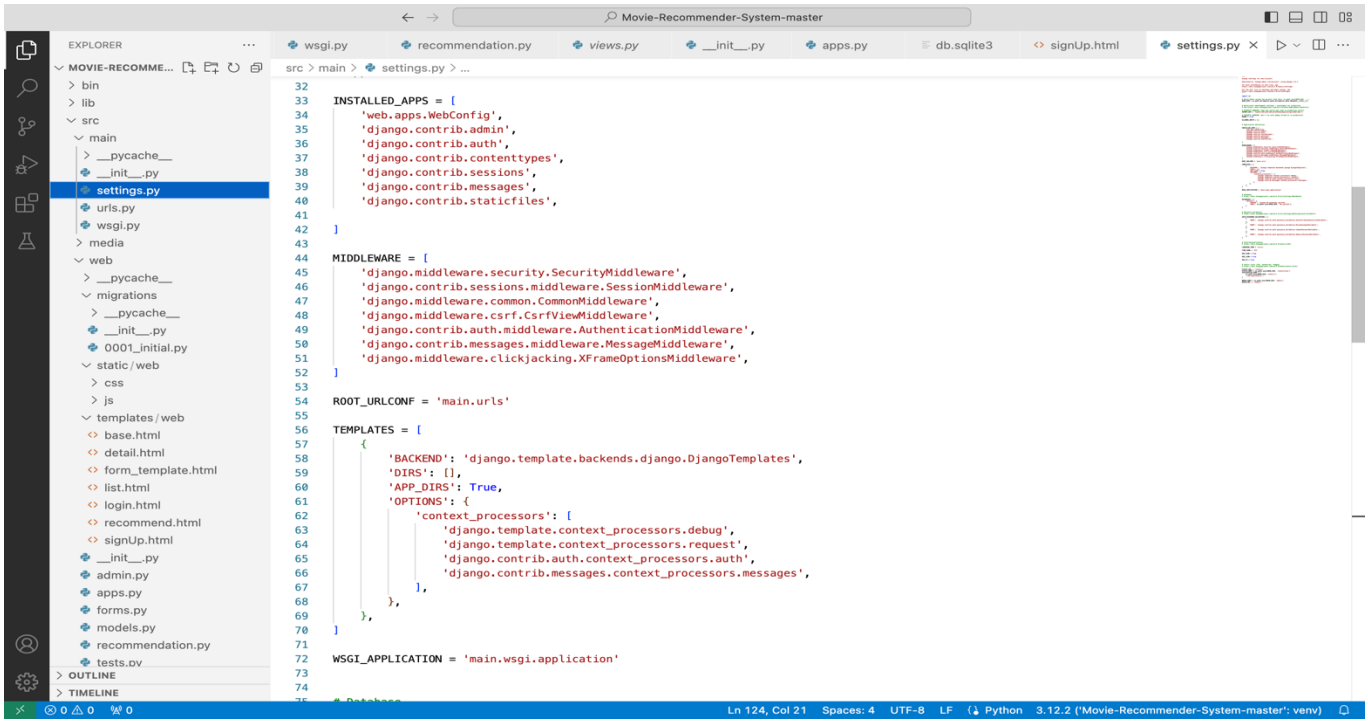


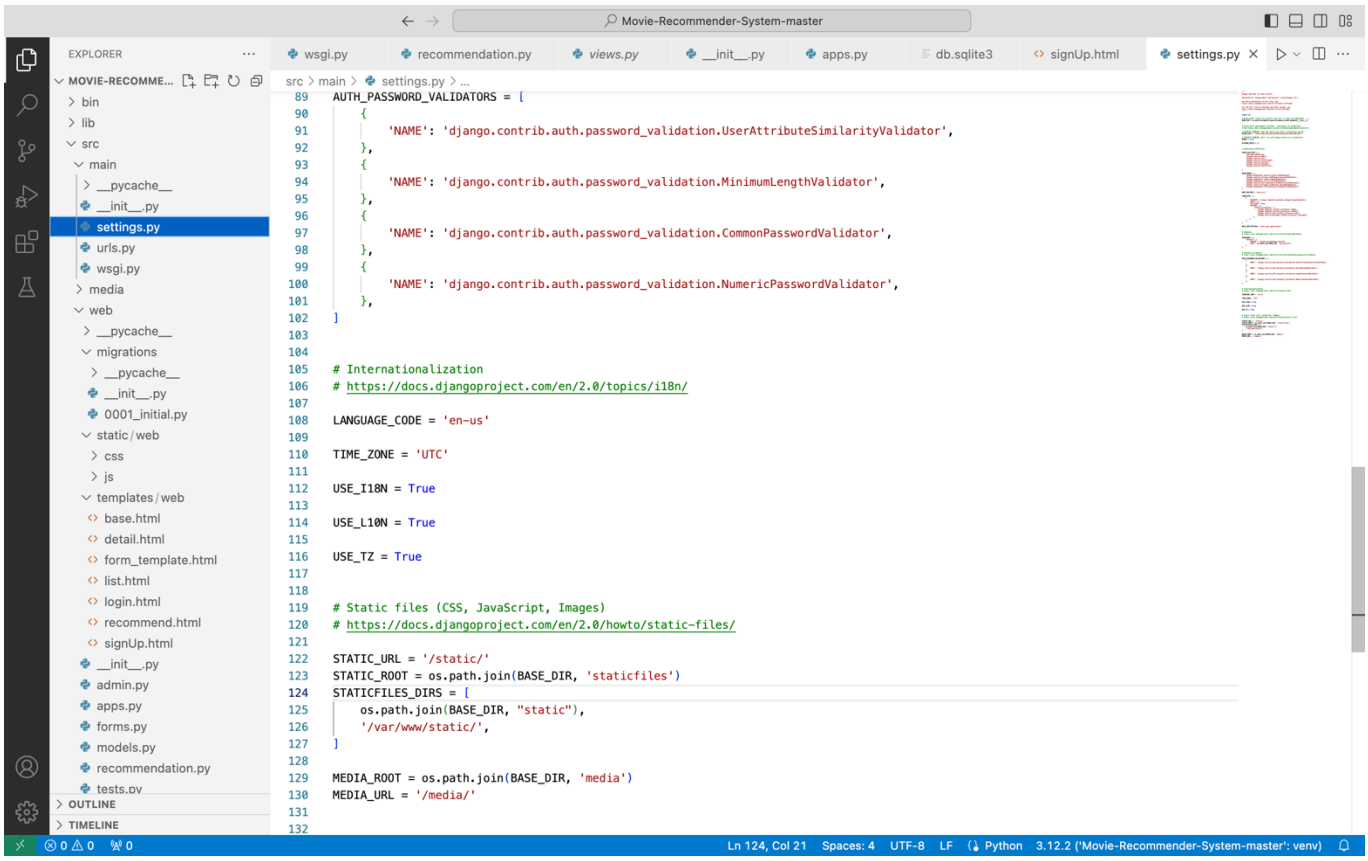

```python
from django.contrib.auth.models import Permission, User
from django.core.validators import MaxValueValidator, MinValueValidator
from django.db import models

class Movie(models.Model):
    title       = models.CharField(max_length=200)
    genre       = models.CharField(max_length=100)
    movie_logo  = models.FileField()

    def __str__(self):
        return self.title

class Myrating(models.Model):
    user    = models.ForeignKey(User,on_delete=models.CASCADE)
    movie   = models.ForeignKey(Movie,on_delete=models.CASCADE)
    rating  = models.IntegerField(default=1,validators=[MaxValueValidator(5),MinValueValidator(0)])

```

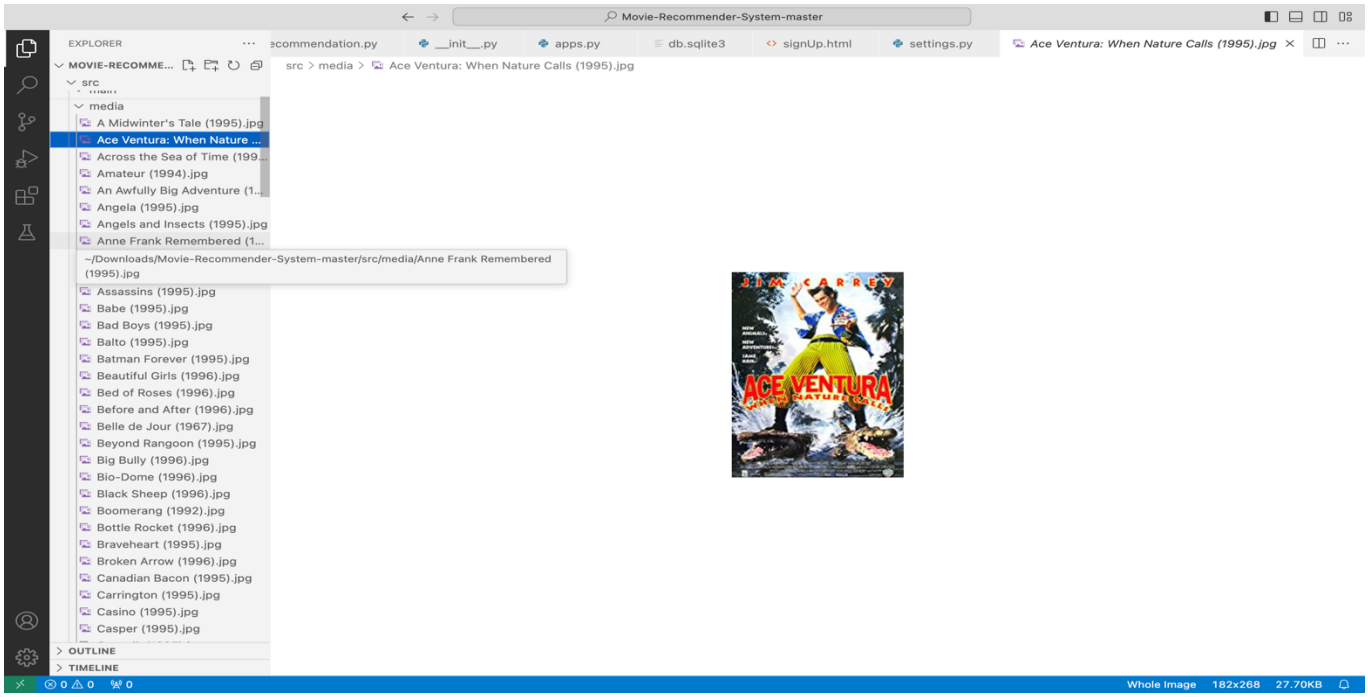



```python


def Myrecommend():
    def normalizeRatings(myY, myR):
        # The mean is only counting movies that were rated
        Ymean = np.sum(myY,axis=1)/np.sum(myR,axis=1)
        Ymean = Ymean.reshape((Ymean.shape[0],1))
        return myY-Ymean, Ymean

    def flattenParams(myX, myTheta):
        return np.concatenate((myX.flatten(),myTheta.flatten()))

    def reshapeParams(flattened_XandTheta, mynm, mynu, mynf):
        assert flattened_XandTheta.shape[0] == int(mynm*mynf+mynu*mynf)
        reX = flattened_XandTheta[:int(mynm*mynf)].reshape((mynm,mynf))
        reTheta = flattened_XandTheta[int(mynm*mynf):].reshape((mynu,mynf))
        return reX, reTheta

    def cofiCostFunc(myparams, myY, myR, mynu, mynm, mynf, mylambda = 0.):
        myX, myTheta = reshapeParams(myparams, mynm, mynu, mynf)
        term1 = myX.dot(myTheta.T)
        term1 = np.multiply(term1,myR)
        cost = 0.5 * np.sum( np.square(term1-myY) )
        # for regularization
        cost += (mylambda/2.) * np.sum(np.square(myTheta))
        cost += (mylambda/2.) * np.sum(np.square(myX))
        return cost

    def cofiGrad(myparams, myY, myR, mynu, mynm, mynf, mylambda = 0.):
        myX, myTheta = reshapeParams(myparams, mynm, mynu, mynf)
        term1 = myX.dot(myTheta.T)
        term1 = np.multiply(term1,myR)
        term1 -= myY
        Xgrad = term1.dot(myTheta)
        Thetagrad = term1.T.dot(myX)
        # Adding Regularization
        Xgrad += mylambda * myX
        Thetagrad += mylambda * myTheta
        return flattenParams(Xgrad, Thetagrad)

    df=pd.DataFrame(list(Myrating.objects.all().values()))
    mynu=df.user_id.unique().shape[0]
    mynm=df.movie_id.unique().shape[0]
```


Movie-Recommender-System-master
EXPLORER
MOVIE-RECOMME...
src
web
__pycache__
views.cpython-312.pyc
migrations
__pycache__
__init__.py
0001_initial.py
static / web
css
js
templates / web
base.html
detail.html
form_template.html
list.html
login.html
recommend.html
signUp.html
__init__.py
admin.py
apps.py
forms.py
models.py
recommendation.py
tests.py
urls.py
views.py
db.sqlite3
manage.py
.gitignore
LICENSE
pyvenv.cfg
OUTLINE
TIMELINE
tail.html
form_template.html
list.html
bootstrap.min.js .../web/...
wsgi.py
recommendation.py
views.py
src > web > views.py > ...
def recommend(request):
raise Http404
df=pd.DataFrame(list(Myrating.objects.all().values()))
nu=df.user_id.unique().shape[0]
current_user_id= request.user.id
# if new user not rated any movie
if current_user_id>nu:
movie=Movie.objects.get(id=15)
q=Myrating(user=request.user,movie=movie,rating=0)
q.save()
print("Current user id: ",current_user_id)
prediction_matrix,Ymean = Myrecommend()
my_predictions = prediction_matrix[:,current_user_id-1]+Ymean.flatten()
pred_idxs_sorted = np.argsort(my_predictions)
pred_idxs_sorted[:] = pred_idxs_sorted[::-1]
pred_idxs_sorted=pred_idxs_sorted+1
print(pred_idxs_sorted)
preserved = Case(*[When(pk=pk, then=pos) for pos, pk in enumerate(pred_idxs_sorted)])
movie_list=list(Movie.objects.filter(id__in = pred_idxs_sorted,).order_by(preserved)[:10])
return render(request,'web/recommend.html',{'movie_list':movie_list})
# List view
def index(request):
movies = Movie.objects.all()
query = request.GET.get('q')
if query:
movies = Movie.objects.filter(Q(title__icontains=query)).distinct()
return render(request,'web/list.html',{'movies':movies})
return render(request,'web/list.html',{'movies':movies})
# detail view
def detail(request,movie_id):
if not request.user.is_authenticated:
return redirect("login")
if not request.user.is_active:
raise Http404
movies = get_object_or_404(Movie,id=movie_id)
#for rating
if request.method == "POST":
rate = request.POST['rating']
Ln 1, Col 1
Tab Size: 4
UTF-8
LF
Python
3.12.2 ('Movie-Recommender-System-master': venv)


Movie-Recommender-System-master
EXPLORER
MOVIE-RECOMME...
bin
lib
src
main
__pycache__
__init__.py
settings.py
urls.py
wsgi.py
media
web
__pycache__
migrations
__pycache__
__init__.py
0001_initial.py
static / web
css
js
templates / web
base.html
detail.html
form_template.html
list.html
login.html
recommend.html
signUp.html
__init__.py
admin.py
apps.py
forms.py
models.py
recommendation.py
tests.py
OUTLINE
TIMELINE
wsgi.py
recommendation.py
views.py
__init__.py
apps.py
db.sqlite3
signUp.html
settings.py
src > main > settings.py > ...
INSTALLED_APPS = [
'web.apps.WebConfig',
'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
]
MIDDLEWARE = [
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
ROOT_URLCONF = 'main.urls'
TEMPLATES = [
{
'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [],
'APP_DIRS': True,
'OPTIONS': {
'context_processors': [
'django.template.context_processors.debug',
'django.template.context_processors.request',
'django.contrib.auth.context_processors.auth',
'django.contrib.messages.context_processors.messages',
],
},
},
]
WSGI_APPLICATION = 'main.wsgi.application'
Ln 124, Col 21
Spaces: 4
UTF-8
LF
Python
3.12.2 ('Movie-Recommender-System-master': venv)


Movie-Recommender-System-master
EXPLORER
MOVIE-RECOMME...
bin
lib
src
main
__pycache__
__init__.py
settings.py
urls.py
wsgi.py
media
web
__pycache__
migrations
__pycache__
__init__.py
0001_initial.py
static / web
css
js
templates / web
base.html
detail.html
form_template.html
list.html
login.html
recommend.html
signUp.html
__init__.py
admin.py
apps.py
forms.py
models.py
recommendation.py
tests.py
OUTLINE
TIMELINE
wsgi.py
recommendation.py
views.py
__init__.py
apps.py
db.sqlite3
signUp.html
settings.py
src > main > settings.py > ...
AUTH_PASSWORD_VALIDATORS = [
{
'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
},
{
'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
},
{
'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
},
{
'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
},
]
# Internationalization
# https://docs.djangoproject.com/en/2.0/topics/i18n/
LANGUAGE_CODE = 'en-us'
TIME_ZONE = 'UTC'
USE_I18N = True
USE_L10N = True
USE_TZ = True
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.0/howto/static-files/
STATIC_URL = '/static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'staticfiles')
STATICFILES_DIRS = [
os.path.join(BASE_DIR, "static"),
'/var/www/static/',
]
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
Ln 124, Col 21
Spaces: 4
UTF-8
LF
Python
3.12.2 ('Movie-Recommender-System-master': venv)


Movie-Recommender-System-master
EXPLORER
MOVIE-RECOMME...
src
media
A Midwinter's Tale (1995).jpg
Ace Ventura: When Nature ...
Across the Sea of Time (199...
Amateur (1994).jpg
An Awfully Big Adventure (1...
Angela (1995).jpg
Angels and Insects (1995).jpg
Anne Frank Remembered (1...
~/Downloads/Movie-Recommender-System-master/src/media/Anne Frank Remembered (1995).jpg
Assassins (1995).jpg
Babe (1995).jpg
Bad Boys (1995).jpg
Balto (1995).jpg
Batman Forever (1995).jpg
Beautiful Girls (1996).jpg
Bed of Roses (1996).jpg
Before and After (1996).jpg
Belle de Jour (1967).jpg
Beyond Rangoon (1995).jpg
Big Bully (1996).jpg
Bio-Dome (1996).jpg
Black Sheep (1996).jpg
Boomerang (1992).jpg
Bottle Rocket (1996).jpg
Braveheart (1995).jpg
Broken Arrow (1996).jpg
Canadian Bacon (1995).jpg
Carrington (1995).jpg
Casino (1995).jpg
Casper (1995).jpg
OUTLINE
TIMELINE
ecommendation.py
__init__.py
apps.py
db.sqlite3
signUp.html
settings.py
Ace Ventura: When Nature Calls (1995).jpg
src > media > Ace Ventura: When Nature Calls (1995).jpg
JIM CARREY
ACE VENTURA
Whole Image
182x268
27.70KB

CHAPTER 5

RESULTS:

Our movie recommendation system project involves developing a hybrid approach that combines both content-based and collaborative filtering techniques. Each approach has its own set of advantages and disadvantages:

1. Content-Based Filtering:

Advantages: This method is straightforward to design and computationally efficient.

Disadvantages: Recommendations are limited to items similar to those the user has already rated or liked. The system may struggle to introduce new interests to users beyond their existing preferences.

2. Collaborative Filtering:

Advantages: Requires minimal domain knowledge as embeddings are automatically learned from user interactions. It can help users discover new interests based on similarities with other users.

Disadvantages: The model's prediction for a (user, item) pair relies on the dot product of corresponding embeddings. If an item is unseen during training, the system cannot create an embedding for it, leading to the cold-start problem.

The hybrid approach aims to overcome these limitations by combining the strengths of both content-based and collaborative filtering:

By leveraging content-based filtering, the system can recommend items based on a user's existing preferences.Incorporating collaborative filtering allows the system to suggest items based on similarities with other users, enabling the discovery of new interests.The hybrid model mitigates the cold-start problem by integrating both techniques, offering a more comprehensive and personalized recommendation experience for users.In summary, the hybrid approach merges content-based and collaborative filtering to capitalize on their respective advantages and address their shortcomings, resulting in a more robust and effective movie recommendation system.

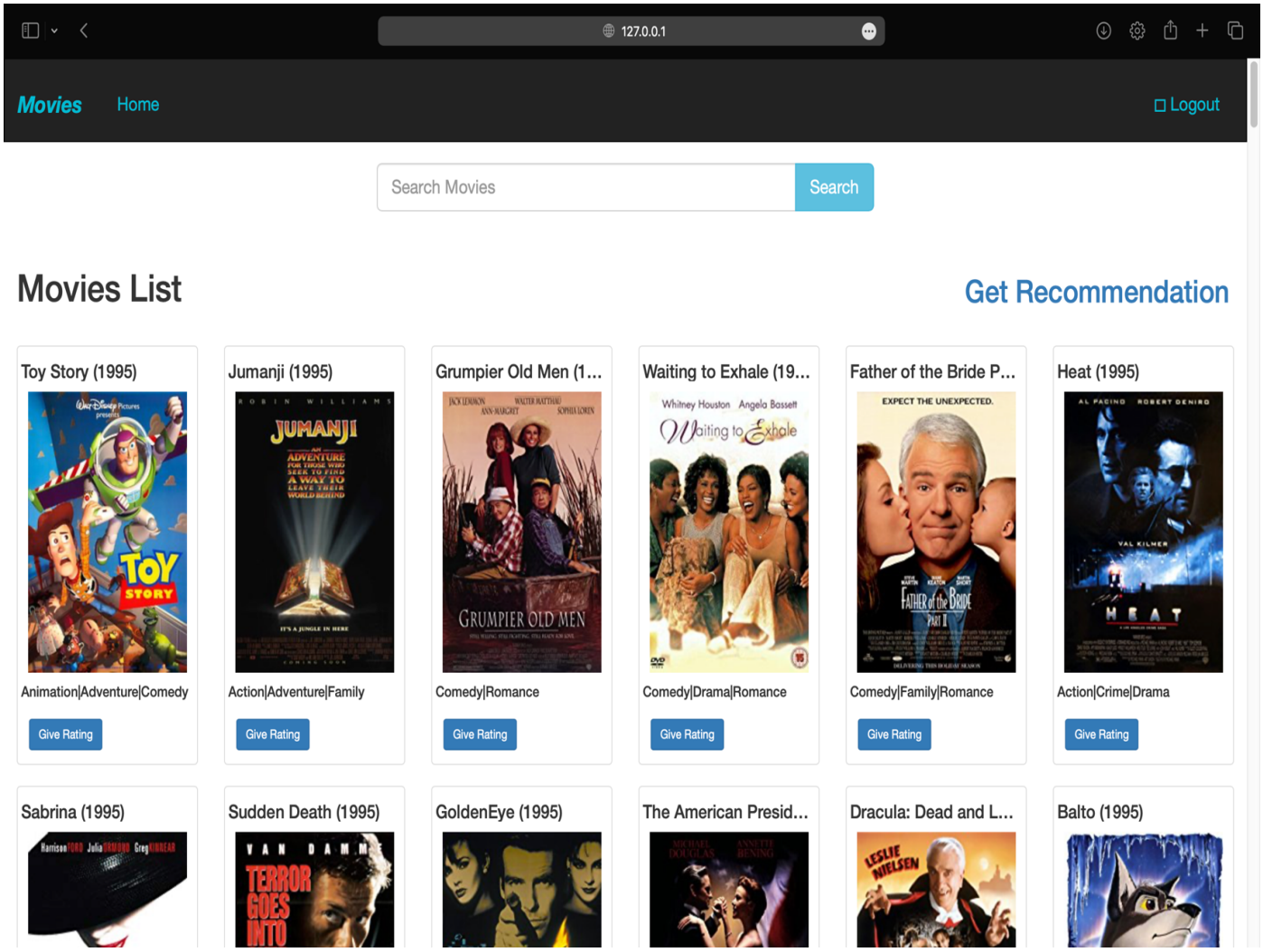

127.0.0.1
Movies
Home
Logout
Search Movies
Search
Movies List
Get Recommendation
Toy Story (1995)
Animation|Adventure|Comedy
Give Rating
Jumanji (1995)
Action|Adventure|Family
Give Rating
Grumpier Old Men (1...
Comedy|Romance
Give Rating
Waiting to Exhale (19...
Comedy|Drama|Romance
Give Rating
Father of the Bride P...
Comedy|Family|Romance
Give Rating
Heat (1995)
Action|Crime|Drama
Give Rating
Sabrina (1995)
Sudden Death (1995)
GoldenEye (1995)
The American Presid...
Dracula: Dead and L...
Balto (1995)

Fig. 5.0

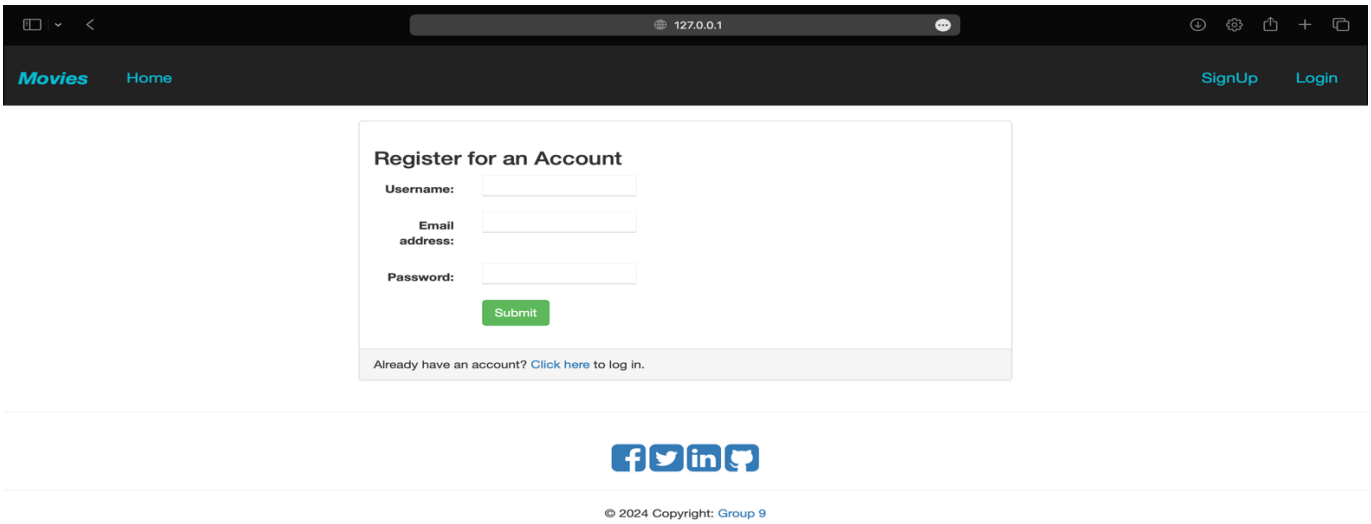

127.0.0.1
Movies
Home
SignUp
Login
Register for an Account
Username:
Email address:
Password:
Submit
Already have an account? Click here to log in.
© 2024 Copyright: Group 9

Fig. 5.1

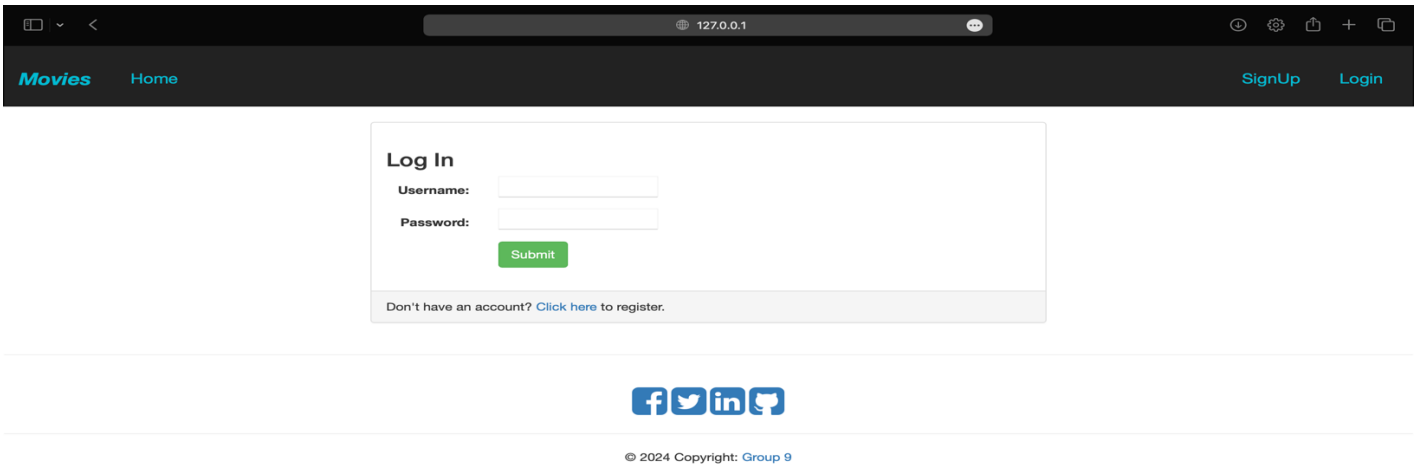

127.0.0.1
Movies
Home
SignUp
Login
Log In
Username:
Password:
Submit
Don't have an account? Click here to register.
© 2024 Copyright: Group 9

Fig. 5.2

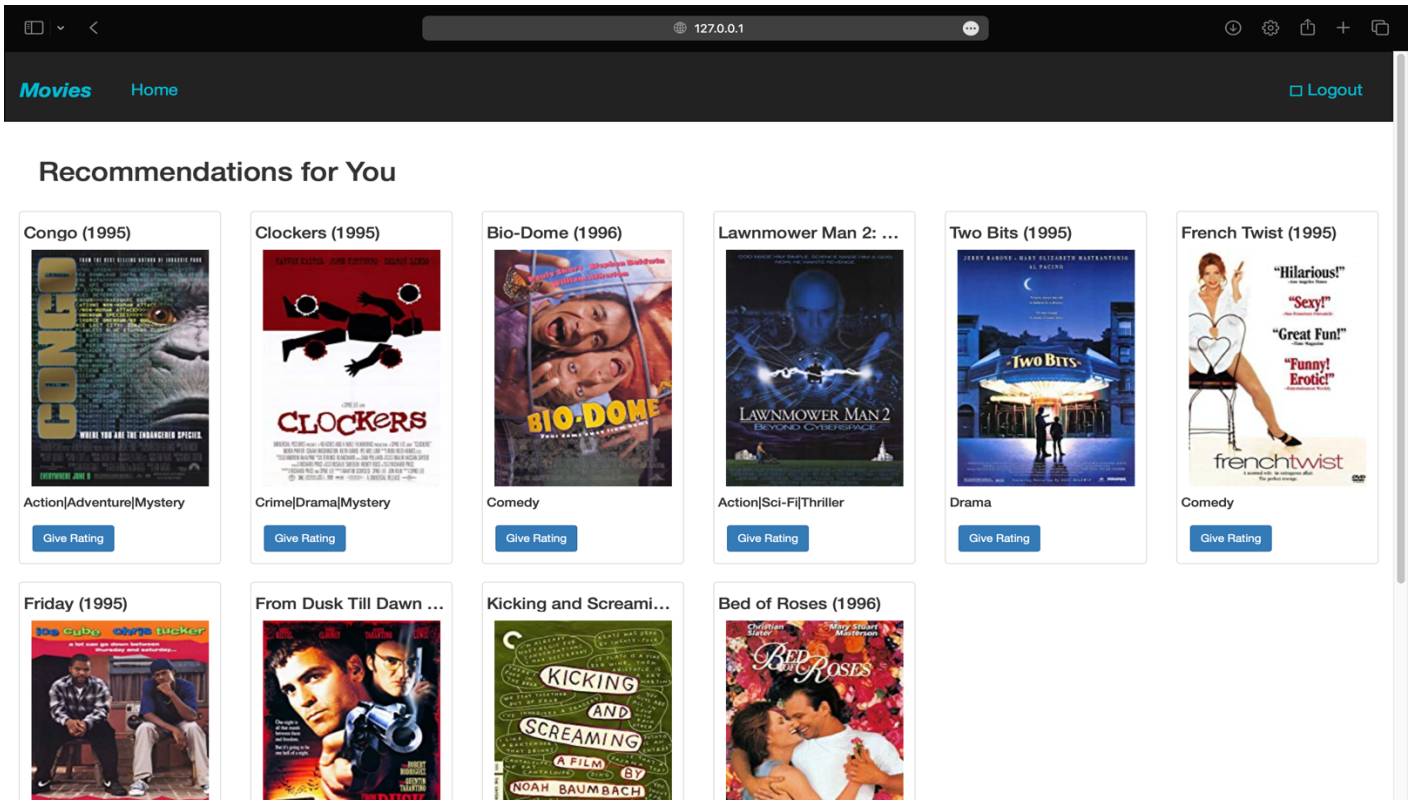

127.0.0.1
Movies
Home
Logout
Recommendations for You
Congo (1995)
Action|Adventure|Mystery
Give Rating
Clockers (1995)
Crime|Drama|Mystery
Give Rating
Bio-Dome (1996)
Comedy
Give Rating
Lawnmower Man 2: ...
Action|Sci-Fi|Thriller
Give Rating
Two Bits (1995)
Drama
Give Rating
French Twist (1995)
Comedy
Give Rating
Friday (1995)
From Dusk Till Dawn ...
Kicking and Screami...
Bed of Roses (1996)

Fig. 5.3

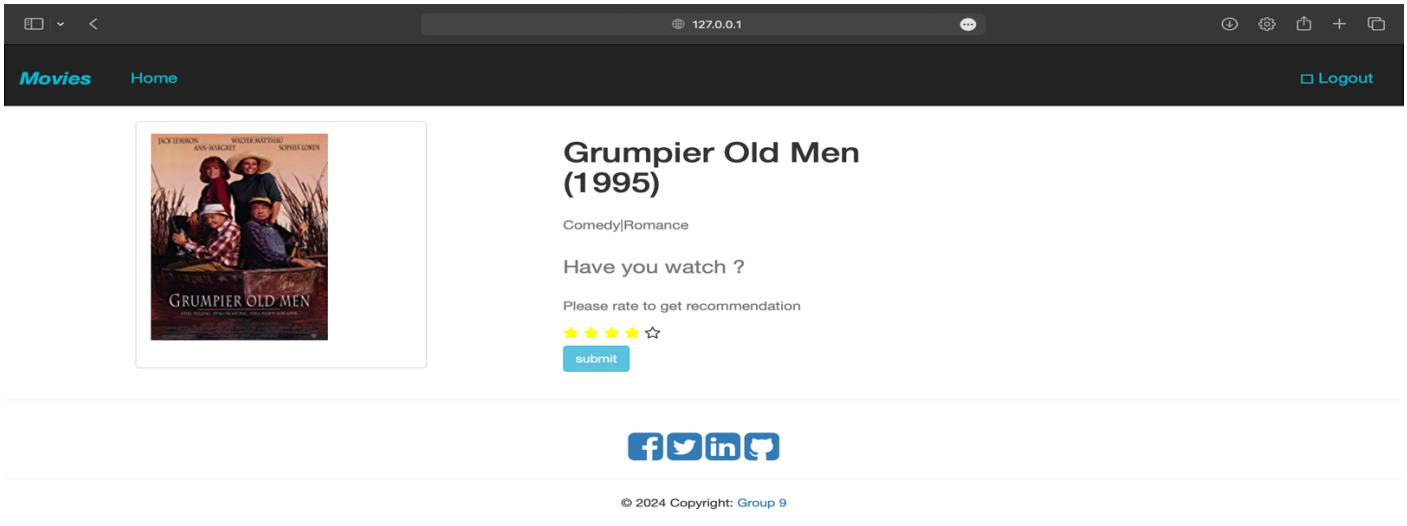

127.0.0.1
Movies
Home
Logout
Grumpier Old Men (1995)
Comedy|Romance
Have you watch ?
Please rate to get recommendation
submit
© 2024 Copyright: Group 9

Fig. 5.4

CHAPTER 6

CONCLUSION AND FUTURE WORK:

Most of the collaborative filtering, content-based filtering, and hybrid recommendation strategies that have been expected so far have been successful in resolving issues while also giving better suggestions. Nevertheless, with the explosion of information, this study topic must be worked on to discover and create new ways for providing recommendations across a wide variety of applications while taking quality and privacy into consideration. It's now much simpler to track out a good movie thanks to server-based recommendation engines. Assists us identify films that we need to watch instead of searching extensively online and helps cinephiles and movie enthusiasts by recommending top-tier films to watch without digging into vast databases, which is time intensive. For this problem, we propose a collaborative and content-based strategy that uses a range of Machine Learning algorithms from a large database to provide a movie recommendation based on the user's taste and previous viewing history or genre.

The findings demonstrate that the hybrid movie recommendation system operates effectively with a modest margin of error when suggesting movies to users. Users can rate specific movies to receive personalized recommendations tailored to their preferences, moving beyond trending selections. The proposed algorithm successfully integrates similarity scores with genre weighting to refine recommendations. Most predictions align with user preferences, indicating the system's accuracy. Increasing data density and expanding genre categories can further enhance recommendation accuracy and specialization. Overall, the results validate the viability of hybrid recommenders as effective tools for movie recommendations.

Movie recommendations inherently reflect subjective opinions, as users often respond differently to suggestions compared to their past activities, posing challenges for mathematical evaluation with significant margins of error. Real-world deployment is essential to gather manual feedback and accurately assess the system's performance.

The development of a Movie Recommender System utilizing Collaborative Filtering Recommendation techniques addresses common challenges in information retrieval systems, particularly the overwhelming volume of available content online. By leveraging Collaborative Filtering, which analyzes user

interactions and similarities to make personalized recommendations, the system provides customers with quick and convenient access to movies aligned with their preferences and interests. Through techniques like Analytic Hierarchy Process (AHP), the recommendation model becomes more accurate and responsive, enhancing data collection procedures and improving overall system performance. However, challenges such as data sparsity and runtime issues inherent in Collaborative Filtering can be mitigated by implementing a Hybrid movie recommendation system, which combines multiple recommendation techniques to leverage their respective strengths and overcome limitations.

In this project, the Movie Recommender System was developed using Python for website development, leveraging its high level of abstraction, readability, and clean code structure. The system utilizes a Relational Database, specifically SQLite, to store user information, relationships, and movie properties efficiently. Additionally, the project integrates Scipy, a Python library that simplifies the implementation of mathematical formulas and equations, enhancing the system's analytical capabilities. The user experience is prioritized, with users required to register on the website to access personalized movie recommendations and interact with other users. Moving forward, the project aims to address system weaknesses and further improve the user interface to enhance usability and engagement. By leveraging these technical capabilities and methodologies, the Movie Recommender System offers a valuable solution to information overload, providing users with tailored movie recommendations while continuously striving for system optimization and enhancement.

Research, Development, and Deployment of Needed Technical Capabilities:

The development of a Movie Recommendation System utilizing a combination of contextual and collaborative filtering techniques requires a comprehensive approach encompassing research, development, and deployment of key technical capabilities. Initially, extensive research is essential to understand the underlying principles of recommendation systems, particularly contextual and collaborative filtering methodologies. Research efforts will involve studying existing literature, exploring state-of-the-art algorithms and models in the field of recommender systems, and analyzing real-world datasets to identify relevant features and patterns that can enhance recommendation accuracy.

In the development phase, technical capabilities will be built upon the foundation laid out in the research phase. This involves implementing and fine-tuning contextual and collaborative filtering algorithms tailored to the specific requirements of the Movie Recommendation System. Contextual filtering will enable the system to consider additional factors such as user preferences, viewing history, and movie metadata (e.g., genre, cast, release year) to generate personalized recommendations. Collaborative filtering techniques will leverage user-item interactions and similarities to predict user preferences and improve recommendation accuracy. Development efforts will also focus on optimizing algorithms for scalability and efficiency to handle large datasets and ensure real-time responsiveness.

Finally, the deployment of the Movie Recommendation System will involve integrating the developed technical capabilities into a user-friendly and robust platform. Deployment efforts will include building a scalable architecture, implementing APIs for seamless interaction with front-end interfaces, and incorporating user feedback mechanisms to continuously refine recommendation quality. User experience will be a central consideration throughout the deployment phase, ensuring that the system delivers intuitive and engaging movie recommendations based on both contextual relevance and collaborative filtering insights. Ongoing monitoring and evaluation post-deployment will enable iterative improvements and refinement of the technical capabilities to meet evolving user needs and preferences.

A VIEW OF THE FUTURE WORK

Looking ahead to future developments in movie recommendation systems, several key areas of focus have been identified for further enhancement and refinement. Firstly, there is a strong interest in improving the search functionality of the system to better align with user requirements. This involves developing more advanced algorithms and techniques that allow users to search for movies based on specific criteria such as genre, actors, directors, or even mood or theme, enabling a more tailored and intuitive movie discovery experience.

Secondly, enhancing the accuracy of movie recommendations remains a priority. Future efforts will involve delving deeper into collaborative filtering techniques and exploring advanced machine learning models to better understand user preferences and behaviors. This will enable the system to recommend movies more accurately based on individual tastes and interests, ultimately improving user satisfaction and engagement.

Another important area of future development is adapting recommendation systems to dynamic environments. This involves creating systems that can continuously learn and adapt to changing user preferences and trends in the movie industry, ensuring that recommendations remain relevant and up-to-date over time.Furthermore, future iterations of the recommendation system will take into account additional factors such as user demographics, specifically age. Understanding how movie preferences evolve with age will enable more targeted and personalized recommendations. For example, considering that younger audiences often prefer animated movies, incorporating age as a factor in the recommendation algorithm will lead to more relevant suggestions for different age groups.

In terms of technical improvements, there is a need to optimize memory requirements for the recommendation system. This will involve refining algorithms to be more efficient in terms of memory usage, enabling the system to handle larger datasets and scale effectively. Additionally, future work will involve testing and evaluating the proposed approach on diverse and larger datasets such as Film Affinity and Netflix datasets to assess performance and ensure robustness across different platforms and scenarios. By addressing these future directions, the goal is to create a movie recommendation system that is more sophisticated, adaptive, and capable of delivering highly personalized and relevant movie recommendations to users.

REFERENCES

1. Ahuja R, Solanki A, Nayyar A (2019) Movie recommender system using k-means clustering and k-nearest neighbor. Proc 9th IntConf Cloud Comput Data SciEngConflu 2019 263–268.
2. Indira K, Kavithadevi MK (2019) Efficient machine learning model for movie recommender.pdf
3. Yang D, Zhou Z (2013) Personalized mining of preferred paths based on web log. Proc 2013 IEEE 11th IntConf Electron Meas Instruments, ICEMI 2013 2:993–997.
4. Wu CSM, Garg D, Bhandary U (2019) Movie recommendation system using collaborative filtering. Proc IEEE IntConfSoftwEngServSci ICSESS 2018-November 11–15. https://doi.org/10.1109/ICSESS.2018.8663822
5. Xu X, Zhang Y (2018) Collaborative filtering recommendation algorithm based on hybrid similarity. 2017 IntConfComputSyst Electron Control ICCSEC 2017 2018:1372–1375. https://doi.org/10.1109/ICCSEC.2017.8447000
6. Yang D, Zhou Z (2013) Personalized mining of preferred paths based on web log. Proc 2013 IEEE 11th IntConf Electron Meas Instruments, ICEMI 2013 2:993–997.
7. M. Shah, D. Parikh and B. Deshpande, "Movie Recommendation System Employing Latent Graph Features
8. in Extremely Randomized Trees," in Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies (ICTCS '16), New York, 2016.
9. Y. Deldjoo, M. Elahi, M. Quadrana and P. Cremonesi, "Using visual features based on MPEG-7 and deep learning for movie recommendation," Int J Multimed Info Retr, vol. 7, p. 207–219, 2018.
10. J. K. Leung, I. Griva and W. G. Kennedy, "Making Use of Affective Features from Media Content Metadata for Better Movie Recommendation Making," arXiv, 2020.
11. S. Reddy, S. Nalluri, S. Kunisetti, S. Ashok and B. Venkatesh, "Content-Based Movie Recommendation System Using Genre Correlation," in Smart Intelligent Computing and Applications, Singapore, 2019.
12. H. Li, J. Cui, B. Shen and J. Ma, "An intelligent movie recommendation system through group-level sentiment analysis in microblogs," Neurocomputing, vol. 210, pp. 164-173, 2016.

13. P. P. Adikara et al. ISSN 2502-3357 (online) | ISSN 2503-0477 (print) regist. j. ilm. teknol. sist. inf. 7 (1) January 2021 31-42 Movie recommender systems using hybrid model based on graphs with co-rated ... http://doi.org/10.26594/register.v7i1.2081 \
14. O.-J. Lee and J. J. Jung, "Explainable Movie Recommendation Systems by using Story-based Similarity," in Explainable Smart Systems 2018 (ExSS '18), Tokyo, 2018.
15. J. Li, W. Xu, W. Wan and J. Sun, "Movie recommendation based on bridging movie feature and user interest," Journal of Computational Science, vol. 26, pp. 128-134, 2018.
16. W. E. Winkler, "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage," The Educational Resources Information Center (ERIC), Washington, DC, 1990.
17. A. Vukotic, N. Watt, T. Abedrabbo, D. Fox and J. Partner, Neo4j in Action, Greenwich, CT, United States: Manning Publications Co, 2014.
18. Anjum, I., & Shaikh, A. A. A REVIEW STUDY ON MOVIE RECOMMENDATION SYSTEM.

APPENDICES

Appendix A: Test Plan Summary

Objectives	Ensure accuracy of algorithms. Verify integration of Django, Python, and SQLite. Validate performance and scalability. Confirm UI functionality and usability
Scope	Algorithm Verification. Integration Testing. Performance Testing. User Interface Testing
Algorithm Verification	Content-Based Filtering**:** Test predefined preferences, verify edge cases. Collaborative Filtering: Simulate user interactions, validate user clusters
Integration Testing	Django Integration: Test views, templates. Database Operations: Verify CRUD operations, data consistency
Performance Testing	Response Time: Measure recommendation generation time. Load Testing: Simulate concurrent users, identify bottlenecks
User Interface Testing	Navigation and Usability: Test functionality and ease of use. Form Submissions: Validate input forms, test data integrity
Test Environment	Development Tools: Django, Python, SQLite. Testing Tools: Postman, Selenium, JMeter. Browsers: Chrome, Firefox, Safar

Appendix B: Project Scope

The project will begin with requirements analysis, defining both functional and non-functional requirements. Functional requirements will cover features such as user authentication, movie search, preference input, and recommendation display. Non-functional requirements will establish performance benchmarks, security protocols, and data privacy measures. In the system design phase, a scalable and modular system architecture will be developed, incorporating Django for the web framework, Python for core logic, and SQLite for database management. A relational database schema will be created to store user profiles, movie data, and user interactions. Algorithm development will involve creating content-based filtering algorithms that recommend movies based on user preferences and movie attributes, and collaborative filtering algorithms that suggest movies based on user interactions and similarities with other users Features will be implemented for users to input their preferences, view recommendations, and provide feedback. Integration and testing will ensure seamless integration between Django views, templates, and the SQLite database. The accuracy and relevance of recommendation algorithms will be tested using predefined test cases, and performance testing will evaluate the system's response time and load handling capacity. UI testing will validate the functionality, navigation, and usability of the web interface. Performance optimization will improve the efficiency of recommendation algorithms to minimize response time, optimize database queries, and implement indexing to enhance performance. Caching strategies will be implemented to reduce data retrieval times and improve recommendation speed. Deployment and maintenance will involve setting up the deployment environment and deploying the application to a suitable hosting platform. Detailed documentation covering system architecture, API usage, and user guides will be provided. A maintenance plan will be established for regular updates, bug fixes, and performance monitoring. The objective of this project is to deliver highly accurate movie recommendations to users. The primary goal is to enhance the quality of the movie recommendation system in terms of accuracy, quality, and scalability, surpassing the limitations of pure approaches. This will be achieved through a hybrid approach that combines content-based filtering and collaborative filtering. By integrating these methods, we aim to alleviate data overload and improve the information filtering process used in social networking sites. Consequently, there is significant potential for advancing scalability, accuracy, and quality in movie recommendation systems. Movie recommendation systems are

immensely powerful and essential. However, they often face challenges related to recommendation quality and scalability, particularly when relying solely on collaborative filtering approaches.

Appendix C: Software and Hardware Requirements

Software Requirements

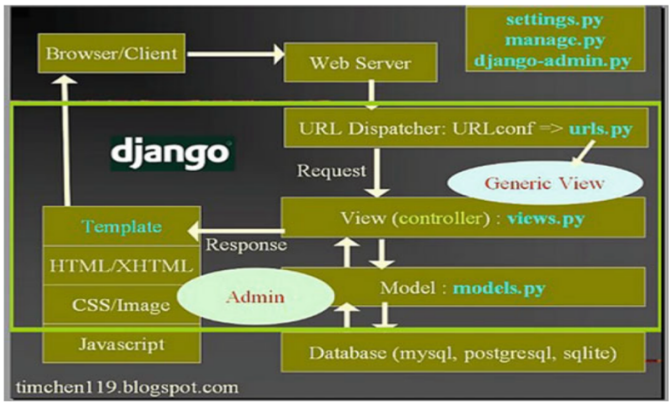
- Python
- Numpy
- Django

Hardware Requirements

- A PC with Windows/Linux OS
- Processor with 1.7-2.4gHz speed
- Minimum of 8gb RAM

Appendix D: System Architecture

For each different individual use different list of movies are recommended ,as user login or enters the user id based on two different approaches used in the work each will recommend the set of movies to the particular user by combining the both the set of movie based on the user the hybrid model will recommend the single list of movie to the user.



The project is a web application designed with a client-server architecture. Users interact with the application via a web browser on their system, acting as clients, while the system resides on the server, which responds to user requests with appropriate outputs.The front end of the project uses HTML5 and CSS3, with dynamic and interactive elements driven by JavaScript and JQuery. Bootstrap is utilized for providing CSS and templates for the UI design. The application is developed using the Django web framework, with Jinja handling dynamic HTML components.The server is responsible for managing client requests and authenticating clients using the CSRF token provided by Django's authentication services. The application follows the Model-View-Template (MVT) architecture, where the model represents database schemas, the view encompasses the user interface, and the templates are a combination of HTML and Jinja. Django's URL dispatcher manages the mapping of templates to the appropriate URLs.The recommendation engine is built using Python, leveraging its extensive libraries and data structures to ensure the system's functionality and efficiency.

PUBLICATIONS

MOVIE RECOMMENDATION SYSTEM

[1]Archit Vijay Singh,[2]Raghav Swaroop,[3]Shreyansh Pandey,[4]Prasant Srivastav

architvij22@gmail.com,raghvasrivastav468@gmail.com,pandeyshreyansh786@gmail.com,psprashant472@gmail.com

[1,2,3,4]*Noida Institute of Engineering and Technology, Greater Noida, Uttar Pradesh, India*

***Abstract*—In our social lives, movie recommendations play a vital role in enhancing entertainment experiences. Movies have a unique ability to provide enjoyment and captivate audiences based on their interests and the overall popularity of the films. A well-designed recommendation system can offer users a curated selection of movies tailored to their preferences.**

Given the vast array of movie options available, consumers often face challenges in discovering new releases or hidden gems due to time constraints. This creates a need for efficient recommendation systems that not only assist users in choosing movies but also contribute to boosting movie sales and overall viewer satisfaction. By suggesting relevant movies, these systems bridge the gap between consumer preferences and available content.

Our study is centered on exploring a hybrid approach to recommendation systems, specifically blending content-based and collaborative filtering techniques. This innovative perspective aims to optimize movie recommendations by combining the strengths of both methods. By examining this hybrid technique, we seek to enhance the effectiveness and accuracy of movie recommendations, ultimately improving the movie-watching experience for users and supporting the film industry.

Keywords—Movie recommendation, Filtering method, Hybrid Method

I. Introduction

The recommendation system is part of daily life, helping people decide on activities based on shared information[14].The recommendation system plays a significant role in everyday decision-making, helping individuals navigate choices based on their preferences and interests. Collaborative filtering models are particularly important in this context, leveraging users' past interactions such as purchases or ratings to predict their future preferences or ratings. Despite the existence of various established approaches, search-based systems are still prevalent in many applications due to their widespread use, although they may lack accuracy.

Researchers have explored several techniques to enhance recommendation systems, including Alternating Least Squares, Singular Value Decomposition, K-Nearest Neighbor, and Normal Predictor algorithms. These methods fall into two main categories: memory-based and model-based collaborative filtering. Memory-based methods adapt easily to incorporate all available ratings, ensuring up-to-date recommendations. In contrast, model-based systems like neural networks learn from user-item interactions to generate recommendations.

Despite advancements, there remains room for improvement in recommendation systems to deliver stronger and more accurate suggestions to users. Movies have broad appeal across genres and age groups, making the film industry highly profitable[11]. Effective recommendations not only aid decision-making but also introduce users to products that align with their interests and preferences.

In this study, we explore various approaches to enhance recommendation systems, recognizing the perpetual human need for enjoyment and satisfaction in daily life. Movies, being universally popular, serve as a prime source of entertainment for people of all ages and preferences, contributing to the thriving movie industry. Understanding user preferences and optimizing movie recommendations can significantly enhance user experience and support the growth of the film sector. To refresh their memory on what they want to watch, many consumers use the Internet, like online stores that sell or rent movies[10].

In the field of machine learning, categorization techniques that use different ways to organize and classify data are prevalent. It's feasible to use data to train these classifiers[8]. Numerous movies are released simultaneously to cater to diverse audiences and generate revenue. However, due to time or financial constraints, many individuals cannot watch all the new releases promptly. Some prefer to watch movies at a later time, risking forgetting their desired selections. To recall what they want to watch, consumers often turn to online platforms like movie retailers or streaming services accessible through web browsers and smartphones. The prevalence of smart TVs and set-top boxes with streaming capabilities is also increasing.

In the realm of machine learning, categorization methods utilizing various data organization and classification techniques are prevalent. These methods are used to train classifiers with data to enhance their efficiency in organizing and categorizing information.

A. Background

Initially, recommendation systems were content-based, relying solely on a user's browsing history to suggest items. This approach could only provide recommendations tailored to individual users, making it unsuitable for recommending a single item to multiple users. To address this limitation, collaborative filtering systems were developed.

Collaborative filtering is a method for making automatic predictions about a user's interests by gathering explicit ratings or taste preferences from many users. The core assumption of this approach is that if Person X shares the same opinion as Person Y on one issue, X is more likely to share Y's opinion on another issue compared to a randomly chosen person. For example, a collaborative filtering system for TV show recommendations could predict which shows a user might like based on a partial list of the user's preferences (likes or dislikes). These predictions are personalized, utilizing information from multiple users, unlike simpler methods that provide an average score for each item based on its overall number of votes. Similarly, item-based collaborative filtering identifies items that are similar to the

item in question and estimates the desired rating based on the average ratings of these similar items[3].

Collaborative filtering systems recommend items by measuring similarities between users and/or items. They suggest items to a user based on the preferences of similar users. These systems are generally classified into three categories: User-Based, Item-Based, and Model-Based collaborative filtering. However, the primary focus is often on User-Based and Item-Based methods. User-Based collaborative filtering identifies users who are similar to the target user and estimates the desired rating as the average rating of these similar users. Item-Based collaborative filtering, on the other hand, recommends items that are similar to items the user has liked in the past.

II. MOTIVATION

The rapid growth of web technologies has inundated internet users with an overwhelming amount of information, making it challenging for them to efficiently and satisfactorily find content that matches their interests. As people's reliance on the internet grows, finding relevant information that meets their needs has become increasingly difficult. To address this issue, recommendation systems have been developed, providing users with an automated way to discover both relevant and new content. These systems are crucial for the success of the e-commerce and IT industries today and are gaining widespread popularity across various applications, such as YouTube, Google, and Amazon. For instance, when users search for videos on YouTube, they encounter a section of recommended videos, sparking curiosity about the underlying mechanisms of these recommendations.

A. Scope

The objective of this project is to deliver precise movie recommendations to users, aiming to enhance the recommendation system's accuracy, quality, and scalability beyond what is achievable with pure approaches. This improvement is achieved through a hybrid approach that combines content-based filtering and collaborative filtering. By addressing data overload, recommendation systems serve as effective information filtering tools on social networking sites. Therefore, there is significant potential for advancing the scalability, accuracy, and quality of movie recommendation systems in this field.

Movie recommendation systems are powerful and essential, yet they often encounter challenges such as poor recommendation quality and scalability issues associated with pure collaborative approaches. By integrating content-based and collaborative filtering methods, the hybrid approach aims to mitigate these problems, offering a more robust and efficient recommendation system.

III. LITERATURE REVIEW

In a study conducted by Ahuja et al. (2019) [1], a recommendation approach combining K-nearest neighbors (KNN) algorithms with the K-means technique is proposed. The strategy involves collecting detailed user information such as user ID, gender, and age to tailor recommendations. Data processing involves using the pandas module to segment customer and movie data into separate dataframes. The Kmeans module utilizes movie genres as key data points, with within-cluster sum of squares (WCSS) aiding in determining the optimal number of clusters. Pearson's correlation similarity and regularization models calculate relationships using a matrix format.For movie rating predictions, the algorithm leverages KNN predictions and a utility matrix (UC grid) to compare outcomes. Pre-processing steps, as seen in Indira and Kavithadevi's work (2019) [2] and the present study (NPCA-HAC), involve outlier removal followed by feature selection and principal component analysis (PCA). Selected characteristics are then grouped using K-means and hierarchical agglomerative clustering (HAC). A trust rating algorithm is applied to assess clustered groups, albeit resulting in some data loss due to dimensionality reduction.PCS[2] is employed to uncover relationships between user ratings. Instead of emphasizing what users like about an item, it highlights what the item itself prefers. Recommendations are then based on how similar items are to the target item[6]. Collaborative filtering addresses challenges like data sparsity, computational complexity, and overspecialization. Combining models are recommended to generate real-time personalized item recommendations.Wu et al. (2019) [4] describe two collaborative models for recommender system usage. This work utilizes user and item collaborative model strategies to develop a system leveraging commonalities across entities. Explicit ratings denote how customers rate items on a specific scale, aiding in the calculation of nearest neighbors (NN) for each user. Principal Component Analysis (PCA) is utilized to explore correlations between user ratings.

Unlike user-focused approaches, item-based recommendations prioritize item preferences when suggesting similar items based on their likeness to a target item.The final recommendation list categorization is based on the MP neuron model. However, scalability remains an unaddressed challenge in this proposed paradigm.An innovative itemcentered strategy integrating collaborative filtering (CF) and content-based filtering (CBF) techniques suggests items based on emotional connections derived from product reviews and comments[18]. Emotions are harnessed to establish item-toitem similarities. While promising, this approach falls short in addressing scalability and computing time constraints.The process of developing and tailoring films considers the cinema preferences of potential audiences. Users are categorized based on their shared tastes and movie ratings. Recurrent Neural Networks (RNNs) are applied to analyze and identify patterns in viewing habits among similar user groups. This paper employs three methodologies: a fundamental recommendation system, a content-based approach, and a collaborative filtering (CF) approach.Machine learning techniques are applied in this study. The basic recommender system utilizes IMDB's weighted rating method to generate a

chart. Two additional techniques are implemented to address challenges such as data sparsity, new user issues, and diminishing computational efficiency. Item-based collaborative filtering (ICF) is favored over user-based CF due to its superior performance in analysis and data processing complexity.To enhance system performance, personalized user information is collected through a registration system. Experimental results are used to assess the level of affinity between participants, resulting in the creation of an adjacency matrix representing user proximity.Xu X's methodology (2018) [5] incorporates feedback from both users and items to enhance recommendation quality using machine learning tools and deep learning models. Mapping users and items generates representations used for item retrieval and ranking, streamlining the recommendation process. Backpropagation is employed to refine the framework.

IV. PROBLEM STATEMENT

The aim of this project is to enhance user engagement and satisfaction by recommending relevant content from a diverse collection of items. Recommendation systems can be either customer-driven or business-driven, focusing on user satisfaction or engagement metrics, respectively. Our research highlights the inefficiencies of content-based recommendation systems and the effectiveness of collaborative filtering in identifying user preferences based on similarity. By studying existing architectures like Mosaic, Netscape, and YouTube, we advocate for a collaborative filtering approach, which leverages user similarities to recommend items based on their past interactions. Our project focuses on building a recommendation system for a blogging site that suggests relevant papers based on user-posted content, tags, and labels, enhancing user experience and content discovery.

V. HYBRID APPROACH

Collaborative Filtering (CF), content-based filtering, and knowledge-based filtering each offer unique benefits and drawbacks. While CF may face challenges such as sparsity and cold start problems, content-based techniques often struggle with limited scope and the necessity of detailed item descriptions. Combining these diverse approaches has the potential to enhance the reliability and effectiveness of recommender systems.

A. Types of Hybrid

The weighted hybrid method involves calculating a score for each recommended item by aggregating recommendation ratings from various sources with assigned weights. Users have the flexibility to adjust these weights using sliders to customize their recommendations. Automatically optimizing these weights is a desired but complex task that requires historical data for empirical bootstrapping to derive an optimal weighting system. This approach aims to refine the weighting scheme based on past experiences and data analysis to enhance recommendation accuracy.

The mixed hybrid method involves generating recommendations from multiple sources and then sorting these recommendations individually. The top recommendations from each source are selected sequentially, rotating through the sources with each selection. This method focuses solely on the ranked position of recommendations rather than considering individual recommendation ratings. If a recommendation has been previously selected from one source, the algorithm moves to the next recommendation in the ranked list from a different source for diversity in recommendations[18].

The cross-source hybrid method emphasizes recommendations that are supported by multiple sources or algorithms. Recommendations that are endorsed by more than one context source or algorithm, such as collaborative filtering from Facebook and content-based recommendations from Wikipedia, are considered more significant in this study. This approach prioritizes consensus among diverse sources to enhance the reliability and relevance of recommendations.

B. Issue with Hybrid Approach

- Effective Integration: The initial challenge involves generating recommendations by leveraging collaborative and content-based data. Both collaborative and content-based techniques can be employed independently or in combination. However, this approach has its limitations. Previous studies have suggested selecting a recommendation system based on predefined quality indicators, but the shortcomings of the chosen system persist over time. The heuristicbased integration lacks a foundational rationale in earlier research[15].
- Streamlined Computation: As the volume of ratings and users expands, recommender systems face increasing challenges in performance. Memory-based approaches offer a straightforward solution by utilizing the entire dataset for generating recommendations efficiently. Alternatively, delayed responses leverage probabilistic techniques within a collaborative filtering framework [16] to mitigate this issue. Additionally, a model-based collaborative filtering approach has been developed, gradually training a latent factor model to address scalability concerns. Notably, there is a lack of research on incremental adaptation of hybrid recommender systems, thus precluding commentary on this topic. When designing a hybrid architecture, it is crucial to consider whether past methodologies can be applied effectively to enhance performance and scalability.

VI. PROPOSED METHODOLOGY

Fig.1 [18]

In this chapter, we will explore different approaches to building movie recommendation systems, progressively advancing to enhance the quality of recommendations. The first method discussed is the Content-Based Recommendation System, which doesn't rely on other users' preferences. Instead, it recommends movies based on the similarity of content to what the user has expressed liking. For instance, if a user indicates a preference for action movies, this system will recommend similar action films until the user explores other genres independently and indicates interest in them. Similarity can be calculated based on various categories like genre, keywords, cast, or director using algorithms such as cosine similarity.

Next, we delve into Collaborative Filtering as a solution to the limitations of the content-based approach. Collaborative Filtering recommends movies based on the preferences of similar users. This User-to-User Collaborative Filtering method identifies users with similar movie-watching behaviors and recommends unseen movies liked by these similar users. Techniques like K Nearest Neighbors (KNN) and Matrix Factorization are employed for this purpose. KNN selects the most similar users and predicts movie ratings based on their ratings, while Matrix Factorization addresses sparsity issues by decomposing the ratings matrix into lowdimensional matrices with latent features.

	Item 1	Item 2	Item 3	Item 4
User A	4	?	3	5
User B	?	5	4	?
User C	5	4	2	?
User D	2	4	?	3
User E	3	4	5	?

Fig.2

Cold start problems, where recommendations are challenging for new users or movies, are addressed using meta-information such as user demographics (location, age, gender) and movie attributes (genre, cast, crew). This supplementary data helps generate recommendations tailored to new users' or movies' characteristics. For instance, for new users lacking viewing history, their profile information can be leveraged to make initial recommendations, while new movies can be recommended based on genre and associated cast and crew details. These strategies aim to improve the effectiveness and relevance of movie recommendations in various scenarios.

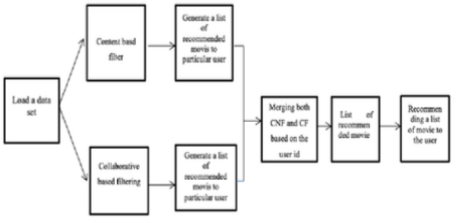


Fig.3 [18]

Each user is recommended a unique list of movies depending on their login or user ID, utilizing two distinct approaches within the project. Each approach recommends a set of movies specific to the individual user. The hybrid model combines both sets of movie recommendations tailored to the user, ultimately providing a single consolidated list of recommended movies.To start building the model, we need to load the necessary datasets: movies.csv, ratings.csv, and users.csv from the MovieLens dataset. This project involves developing two models: content-based and collaborative filtering. Each model generates a list of recommended movies for a specific user. By combining the outputs of both models based on the user ID, a final unified list of recommended movies is produced for each individual user.

	userId	movieId	rating
0	1	1	4.0
1	1	3	4.0
2	1	6	4.0
3	1	47	5.0
4	1	50	5.0
...	...	...	...
100831	610	166534	4.0
100832	610	168248	5.0
100833	610	168250	5.0
100834	610	168252	5.0
100835	610	170875	3.0

Fig.4

VII. Conclusion

The implementation of recommended solutions has opened up new avenues for accessing personal information online. Information retrieval systems often face the challenge of information overload, which our solution mitigates by providing customers with efficient access to products and services that may otherwise be difficult to find. By leveraging past customer feedback and tailoring recommendations to individual preferences using the Analytic Hierarchy Process (AHP), our recommendation model has become more accurate and responsive. This approach has not only validated the model but also enhanced system performance and data collection accuracy.

In our project, we developed a Movie Recommender System based on Collaborative Filtering. We utilized Python for website development due to its high level of abstraction, readability, and clean code, which enhances the overall quality of our project. To manage user data, relationships, labels, and properties effectively, we employed a Relational Database, specifically SQLite. Additionally, we leveraged the Scipy Python library to facilitate the implementation of mathematical formulas and equations within our application.

The movie recommendation website operates by first requiring users to register before utilizing the recommendation features. Once registered, users can interact and share important information with other users connected to the website. Moving forward, we aim to address system weaknesses and enhance the user interface to further improve user experience and system efficiency.

A. Future Scope

In future work will focus on addressing the following key areas:

1.Enhancing movie search functionality to align more closely with user preferences and requirements.

2.Improving the accuracy of movie recommendations by refining our recommendation system algorithms.

3.Developing the recommendation system to operate effectively in dynamic environments where user preferences and trends evolve over time.

4.Expanding the scope of our approach to consider additional factors such as user age alongside movie genres. Understanding age-related preferences can significantly enhance recommendation accuracy, for instance, recognizing that animated movies may be favored during childhood.

5.Optimizing memory usage within our approach to ensure efficient performance, particularly when scaling to larger datasets beyond the ones used in our current implementation. This includes evaluating the system's performance on datasets from platforms like Film Affinity and Netflix to assess its broader applicability and effectiveness.

These future directions aim to refine and expand our recommendation system's capabilities, ultimately enhancing user satisfaction and system performance across diverse movie datasets and evolving user preferences.

Acknowledgment

We would like to express our sincere appreciation to Ms. Pooja Sharma for her valuable feedback and suggestions, which have significantly improved the quality of our paper. We are also grateful for her diligent review of our work during its development. Furthermore, we extend

our thanks to the Department of Computer Science at Noida Institute of Engineering and Technology for their support and encouragement, which inspired us to undertake this research paper.

REFERENCES

[1] Ahuja R, Solanki A, Nayyar A (2019) Movie recommender system using k-means clustering and k-nearest neighbor. Proc 9th IntConf Cloud Comput Data SciEngConflu 2019 263–268.

[2] Indira K, Kavithadevi MK (2019) Efficient machine learning model for movie recommender.pdf

[3] Yang D, Zhou Z (2013) Personalized mining of preferred paths based on web log. Proc 2013 IEEE 11th IntConf Electron Meas Instruments, ICEMI 2013 2:993–997.

[4] Wu CSM, Garg D, Bhandary U (2019) Movie recommendation system using collaborative filtering. Proc IEEE IntConfSoftwEngServSci ICSESS 2018 November 11-15. https://doi.org/10.1109/ICSESS.2018.8663822

[5] Xu X, Zhang Y (2018) Collaborative filtering recommendation algorithm based on hybrid similarity. 2017 IntConfComputSyst Electron Control ICCSEC 2017 2018:1372–1375. https://doi.org/10.1109/ICCSEC.2017.8447000

[6] Yang D, Zhou Z (2013) Personalized mining of preferred paths based on web log. Proc 2013 IEEE 11th IntConf Electron Meas Instruments, ICEMI 2013 2:993–997.a

[7] M. Shah, D. Parikh and B. Deshpande, "Movie Recommendation System Employing Latent Graph Features in Extremely Randomized Trees," in Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies (ICTCS '16), New York, 2016.

[8] Y. Deldjoo, M. Elahi, M. Quadrana and P. Cremonesi, "Using visual features based on MPEG-7 and deep learning for movie recommendation," Int J Multimed Info Retr, vol. 7, p. 207–219, 2018.

[9] J. K. Leung, I. Griva and W. G. Kennedy, "Making Use of Affective Features from Media Content Metadata for Better Movie Recommendation Making," arXiv, 2020.

[10] S. Reddy, S. Nalluri, S. Kunisetti, S. Ashok and B. Venkatesh, "Content-Based Movie Recommendation System Using Genre Correlation," in Smart Intelligent Computing and Applications, Singapore, 2019.

[11] H. Li, J. Cui, B. Shen and J. Ma, "An intelligent movie recommendation system through group-level sentiment analysis in microblogs," Neurocomputing, vol. 210, pp. 164-173, 2016.

[12] P. P. Adikara et al. ISSN 2502-3357 (online) | ISSN 2503-0477 (print) regist. j. ilm. teknol. sist. inf. 7 (1) January 2021 31-42 Movie recommender systems using hybrid model based on graphs with co rated … http://doi.org/10.26594/register.v7i1.2081 \

[13] O.-J. Lee and J. J. Jung, "Explainable Movie Recommendation Systems by using Story-based Similarity," in Explainable Smart Systems 2018 (ExSS '18), Tokyo, 2018.

[14] J. Li, W. Xu, W. Wan and J. Sun, "Movie recommendation based on bridging movie feature and user interest," Journal of Computational Science, vol. 26, pp. 128-134, 2018.

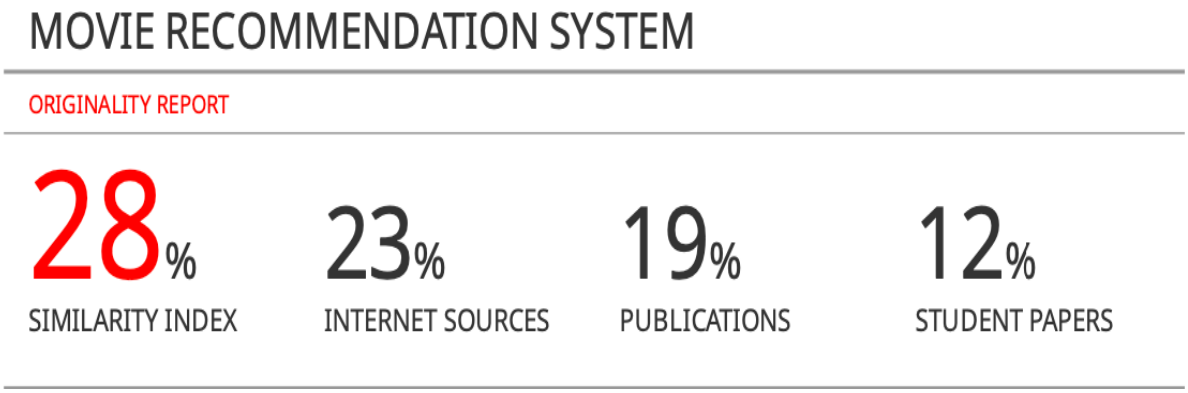
[15] W. E. Winkler, "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage," The Educational Resources Information Center (ERIC), Washington, DC, 1990.

[16] C. D. Manning, P. Raghavan and H. Schütze, An Introduction to Information Retrieval, Cambridge: Cambridge University Press, 2008.

[17] A. Vukotic, N. Watt, T. Abedrabbo, D. Fox and J. Partner, Neo4j in Action, Greenwich, CT, United States: Manning Publications Co, 2014.

[18] Anjum, I., & Shaikh, A. A. A REVIEW STUDY ON MOVIE RECOMMENDATION SYSTEM.

PLAGIARISM REPORT



CURRICULUM VITAE

Archit Vijay Singh

+91 9565280709 architvij22@gmail.com Greater Noida Archit Vijay Singh

Summary

Experienced and dedicated web developer proficient in Python, Java, Django, and MEAN stack technologies. Skilled in developing robust web applications and platforms. Committed to delivering high-quality work and continuously enhancing skills. Seeking an opportunity in a reputable company to contribute expertise and drive success.

Education

Noida Institute of Engineering and Technology, B.tech (Computer Science) 2021-2025

Holy Cross School, 10th and 12th (PCM with Java) 2017-2020

- Percentage 12th : 82.8
- Percentage 10th : 92.8

Experience

Internshala Web Development Intern.(Project:-Login Page)

Internshala Machine Learning Intern.(Project:-Course Recommender using StreamLit)

Skills

- Java
- Python
- Web Development
- MEAN Stack

Additional Experience And Awards

1st Position: Cloud Solution Awareness Competition.

Top Performer: Machine learning at Internshala.

Technologies

Languages: Python, Java, SQL, JavaScript, HTML, CSS, PHP, MEAN

Software: Visual Studio, Eclipse, Net Beans, MySql, Anaconda

Hobbies

- Coding
- Reading Mangas
- Drawing

Raghav Swaroop

Greater Noida (U.P)| +917521949761 | raghavsrivastav468@gmail.com

EDUCATION

- B. Tech in Computer Science(2021-2025) from Noida Institute Of Engineering and Technology ,(Greater Noida) CGPA:7.3
- Intermediate from Shiv Jyoti Convent School ,Kota (Rajasthan) with 77.4 percent.
- Highschool from Shiv Jyoti Convent School ,Kota (Rajasthan) with 94.2 percent.

CAREER SUMMARY

"Dynamic and motivated B.Tech student with a specialization in Computer Science. Possessing a solid foundation in Cloud computing and java, along with hands-on experience gained through internships and academic project. . Committed to continuous learning and professional growth, seeking to embark on a rewarding career journey where I can make meaningful contributions and achieve personal and organizational objectives."

WORK EXPERIENCE

- Internship Trainee at Eduonix (May 2023)

SKILLS AND STRENGTHS

- Programming Languages: Java ,Python ,HTML,CSS.
- Database : MySQL
- Cloud Computing
- Natural language: English and Hindi
- Soft Skills: Good Verbal Communication Skills, Strong analytical and Problem Solving Skill.

PROJECTS

- Developed a Web Tech Based Movie Recommendation System.
- Developed Emotion Recognition System using Django.

CERTIFICATIONS

Exploratory Data Analysis(https://coursera.org/share/c275ad26d45359b112c719ada153637f)

Introduction to Big Data(https://coursera.org/share/98eccfa2ca9caae12e7316859edaa0aa)

Areas Of Interest

- Research and development
- Content Writing
- Coding

Shreyansh Pandey

(+91)8595183415,pandeyshreyansh786@gmail.com •Noida (U.P)

Education

• Noida Institute of Engineering and Technology , Greater Noida
Bachelor of Technology (Computer Science) -CGPA: **7/10** •Nov (2021)-present

• St. Basil School ,Basti• ICSE Board–**75percent**

• **WorkExperience**

Internship Trainee at Eduonix • May 2023

- Designed and developed responsive web pages using HTML, CSS, and JavaScript to enhance user experience and engagement.
- Collaborated closely with team members to identify, troubleshoot, and resolve technical issues, ensuring smooth project execution.
- Actively participated in brainstorming sessions and contributed ideas for new features and improvements to the web application.
- Assisted in the implementation of frontend features and backend functionalities, working in team with developers and stakeholders.

Skills

• Soft skills: Excellent verbal and communication skills, strong analytical and problem-solving skills
• Programming Languages: JAVA,HTML ,Python
• Database: MySQL ,
• Natural Language: English, Hindi

Projects/Reports

- Developed Web Tech Based Movie Recommendation System
- Developed a Full stack E-commerce website with good user Experience

Certifications

• Introduction to Web Development with Html, Css ,Javascript |IBM|Coursera
• Data Structure|University of Calinfornia San Diego|Coursera

Achievements

- Completed Data Structures and Algorithms course by Coding Blocks, covering essential concepts such as arrays, linked lists ,stacks, queues , trees , graphs , sorting algorithms , and searching algorithms . Developed proficiency in problem solving techniques and algorithmic thinking through hands –on coding exercises and projects.
- Attended a week workshop on Adavance java Programming, presented by experts from the Department of Computer Science and Engineering, NIET gaining knowledge of various optimization techniques and their applications in the industry.

Prasant Srivastav

Psprashant472@gamil.com
8429924143
LINKEDIN PROFILE: **Prashant**

Skills

Languages: Java, Python, JavaScript, OOPS, SQL, HTML, CSS **Technologies &**

Tools: AWS, EC2, DynamoDB, S3, ReactJS

CERTIFICATION

Python basics: https://coursera.org/share/b1b7ed8b843734a475ce337bf46ddd97

Python for data Science , AI & development : https://coursera.org/share/aba87f4fff01a8866a8bf832c8bb1f08

The Bits and bytes of Computer Networking : https://coursera.org/share/6a25b3bdbed12165d600ec6a547a55c3

Education

Degree- Noida Institute of Engineering and Technology

- Bachelor of Technology (B. TECH)
- Pursuing from Computer Science Branch

School- St. Xavier's High School

- High school(10^{th})- **2018 CBSE**
- Secondary school(12^{th})- 2020 **CBSE**

EXPERIENCE:

FRESHER

Project Work

1. **Temperature Converter:** A temperature converter is a tool or program that allows you to convert temperatures between different temperature scales. The most common temperature scales are Celsius (°C), Fahrenheit (°F), and Kelvin (K).
2. **Movie Recommendation System:** A movie recommendation system is a type of technology or software that suggests movies to users based on their preferences, viewing history, and other relevant factors. These systems utilize various algorithms and data analysis techniques to predict which movies a user might
3. **Restaurant Website:** A restaurant website is a digital platform created by a restaurant to showcase its offerings, provide essential information to potential customers, and facilitate online reservations or orders. It serves as an online representation of the restaurant's brand and allows customers to interact with the restaurant digitally
4. **Alarm Clock:** Building a simple yet effective alarm clock app using Android Studio

Address:

74, Kurmi Tola, Chowk
Azamgarh