**KABARAK UNIVERSITY**

**SCHOOL: SCHOOL OF SCIENCE ENGENEERING AND TECHNOLOGY**

**DEPARTMENT: COMPUTER SCIENCE AND IT**

**RESEARCH PROJECT**

# PROJECT TITLE: MOVIE RECOMMENDER SYSTEM

**A RESEARCH PROJECT SUBMITTED IN THE DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY IN PARTIAL FULFILMENT OF DEGREE IN INFORMATION TECHNOLOGY**

**PRESENTED BY: CHRIS NGATIA**

**REGNO: INTE/MG/2341/09/19**

**COURSE NAME: PROJECT II**

**UNIT CODE: INTE 414**

**NOVEMBER 2022**

## DECLARATION

I solemnly declare that this project proposal is based on my own work carried out during the course of my study under the supervision of Mr. SIMON RUORO. The statements made is the outcome of my research work. The work contained in this report is original and has never been submitted to any other Institution for the award of any other degree/diploma/certificate in this university or any other University in Kenya. I have followed the guidelines provided by the university in writing the proposal.

**Name: CHRIS NGATIA**

Sign………………………………. Date…………………...

SUPERVISOR

This paper has been submitted with my approval.

NAME: MR SIMON RUORO SIGN …………………………

## ACKNOLEDGEMENT

I would like to express my appreciation to all those who provided me with the possibility to complete this project on the topic Speech Emotion Recognition. I would also like to thank my parents, siblings and friends who helped me a lot to complete this proposal within the required time.

I want to give a special gratitude to my project supervisor, Mr. SIMON RUORO whose contribution in suggestions and guidance helped me to coordinate my project. I came to know about many new things through him, and am grateful, especially in writing this project proposal.

## ABSTRACT

Traditional movie recommendation systems often rely on generalized algorithms that may not capture users' diverse preferences accurately. This document introduces the Movie Recommender Application, a hybrid recommendation system that addresses the limitations of conventional approaches. This system leverages both content-based and collaborative filtering techniques to provide users with personalized and relevant movie recommendations.

In the past, traditional recommendation systems struggled to offer precise suggestions due to their reliance on limited user data or simple algorithms. The Movie Recommender Application bridges this gap by combining the strengths of content-based filtering, which analyzes movie attributes, and collaborative filtering, which draws insights from user interactions. By doing so, the application offers an advanced solution that adapts to users' unique tastes and preferences.

The core of the Movie Recommender Application lies in its user-centric design. The front-end development emphasizes an intuitive user interface, featuring a user-friendly homepage with real-time search suggestions and personalized recommendations. Users can seamlessly navigate and explore movie details, including title, description, cast, and user reviews, enhancing their decision-making process. Additionally, the application's responsive design ensures optimal performance across various devices.

To further enhance its functionality, the application integrates a Flask-based backend service that facilitates communication between the content-based and collaborative filtering components. This integration enables the generation of hybrid recommendations, combining the best aspects of both methodologies. Additionally, the use of Firebase's Remote Config allows the application to dynamically adapt server URLs, ensuring smooth and uninterrupted user experience.

In conclusion, the Movie Recommender Application revolutionizes movie recommendations by offering a sophisticated hybrid system that overcomes the limitations of traditional approaches. By combining content-based and collaborative filtering techniques, providing an intuitive user interface, and offering dynamic backend integration, the application stands as a comprehensive solution that empowers users to discover movies aligned with their preferences.

## Table of Contents

# CHAPTER ONE

## 1.1 INTRODUCTION

Recommendation system helps users to find and select items (e.g. books, movies, restaurants) from a large Number available on the web or other electronic information sources. Given a large set of objects and describing user needs, they offer the user a small set of items that are well suited description. Similarly, a film recommendation system provides a level of comfort and personalization Helps users to interact better with the system and watch movies that meet their needs. Providing this level of User comfort was our primary motivation in opting for the film recommendation system as our BE project. The main objective of our system is to recommend movies to our users based on their viewing history and ratings they provide. The system will also recommend various e-commerce companies to promote their products depending on the genre of films of their choice to specific customers. Personalized recommendation engines help Collaborative filtering and Content-based filtering is the predominant approach for providing recommendations to users. They are both the best the specific scenarios are applied due to their respective fluctuations.

## 1.2 BACKGROUND STUDY

Recommender systems might be a new concept in research area,but it has been prevailing in the society since ages.Humans are known to have evolved and grow certain traits of complex thinking, using language, making tools etc. during the last 100,000 years. The concept of recommendations could be seen in case of cavemen, ants and other creatures too. We may have seen ants running around in our house walking in a line behind the ants that went before and found food.This is because ants have genetically evolved to leave markers for other ants which further serve as a recommender to other ants,showing them the way to food. Ancient civilizations flourished between the period 4000 to 1200 BCE.If we talk about recommendations at that time,these could possibly range from what crop to cultivate and during what time, what religion to follow etc. Later, during the colonization period between the 11th-18th centuries, the recommendations got transformed into what territory to absorb in accordance with the fruitful aspects like land fertility, manpower (for slavery) and numerous other resources.

Humans undoubtedly rely on recommendations from other sources for one purpose or the other, given the fact that the choices available to us are increasing manifold day by day. While buying a cell phone, we go through numerous sites to check the reviews for the various sets available that we might have interest in; we take suggestions from our peers and so on. So is the case with buying new vehicles, going for movies, reading suggestions and so on.Even matrimony sites have a lot to offer.Earlier what our aunts used to do is now replaced by the matrimony sites thereby saving the effort. So we can conclude recommender systems to be as software tools that narrow down our choices and provide us with the most suitable suggestions as per our requirements.Recommender systems evolved as an independent research area in the mid 1970s in Duke University. Since then, a lot of work has been done in this area till date with the introduction of various approaches. the first recommender  system that came into existence was Tapestry1 which was developed at the Xerox Palo Alto Research Centre. the motivation that led to its development was the growing number of incoming emails, mostly unnecessary ones that were too annoying sometimes and difficult to maintain. So what was done to overcome this problem was that users had their mailing lists made and only those people who were in the contact lists could send the mails to them or the ones the user might want to hear from while the others were sent to the spam list, exactly how it works in present in our email accounts. For this collaborative filtering was used. Soon recommender systems got more popular and now play an important role in internet sites like Amazon,Pandora, Netflix, Matrimonial sites,Social networking sites,YouTube, Yahoo,Tripadvisor etc. RSs did not travel this whole path alone, rather included Artificial intelligence, Information retrieval and Human computer interaction in their journey,and hence got more efficient and gained more popularity.

Till today, researchers have developed many recommender systems for almost every domain like entertainment, social-networking sites,content-based sites (e-learning, books or articles recommendation, efiltering etc.), e-commerce, tourism, match-making and a lot more, all dealing with real-world. Thus, this paper gives an overview of various recommender systems developed so far and their application areas. We have categorized recommender systems on the basis of their application areas: Entertainment, Socialnetworking based, Content-based, Services-based and E-commerce. Our focus is movie recommendation app under entertainment.

### 1.2.1 CURRENT SYSTEM

**Simple Recommender**

The Simple Recommender offers generalized recommendations to every user based on movie popularity and (sometimes) genre. The essential idea behind this recommendation is that movies that are more popular and have more ratings will have a better probability of being liked by the typical audience. This model doesn’t give personalized recommendations based on the user.
The implementation of this model is extremely trivial. The movies needs to be sorted based on parameters like ratings and popularity and display the highest rated movies of the list. Also, we can also use a genre parameter to fetch top movies of a specific genre.

Before getting started with this -

• We need a metric to score or rate movie.

• Calculate the score for every movie.

• Sort the scores and recommend the best rated movie to the users.

Recommendation would be done using IMDB’s weighted rating (wr) which is given as :-

Weighted Rating(WR) =

$(v.R + m.C)$

$(v + m)$

(3.1) where,

• v is the number of votes for the movie.

• R is the average rating of the movie.

• m is the minimum votes required to be listed in the chart.

• C is the mean vote across the whole report

**Limitations of Simple Recommendation System**

Simple Recommender which suffers from severe limitations.

1)It gives the same recommendations to everyone, without considering the user profile and one's personal taste. If an individual who enjoys action movies (and hates romantic movies) were to look at recommended Top 10 Chart, one probably wouldn't like most of the films . If one were to go one step further and look at our charts by genre, then still one wouldn't get the simplest recommendations.

For instance, consider a person who loves movies like Mission Impossible for example.One inference we can obtain is that the person loves action movies. Even if he was recommended a romantic movies list, he wouldn't find these as the top recommendations.

2)Personalization is a key factor while making recommendation engine, and the engine should take it into consideration. It should also measure the similarity between movies based on certain metrics and suggests movies that are most similar to a particular movie that a user liked which can be done through Content Based Filtering.

## 1.3 STATEMENT OF THE PROBLEM

Digital TV broadcasting brought a huge increase in number of tv channels and amount of content they provide.It is very hard for users to find and watch the content they actually like among these options. Users have to zap around the channels or follow program guidelines to find the contents they prefer.But today the programming guide is a long list,and zapping through it takes a lot of time if you are up to it.

Lately, the demand for recommendation services have severely increased thanks to the huge flow of latest content on to the web . In order for users to seek out the content they desire, competent recommendation services are extremely helpful. It's challenging for a user to seek out the acceptable movies suitable for his/her tastes. Different users like different movies or actors. It is important to seek out a way of filtering irrelevant movies and/or find a group of relevant movies. Therefore, automated recommendation services are implemented to ease this task.

## 1.4 PURPOSE OF THE STUDY

The purpose of a recommender system may be is the increase of the system efficiency by allowing the user to more directly get the content she is looking for. This in return can lower the amount of data to be exchanged, thus lowering the costs of running a system. In clear, the different objectives or purposes that a recommender may have are: be a major service or not, enable easy access to previously hard to find items, secure users' loyalty and increase the system's efficiency.

## 1.5 OBJECTIVE

### 1.5.1 MAIN OBJECTIVES

1)Filter and predict only those movies that a corresponding user is most likely to want to watch.

### 1.5.2 SPECIFIC OBJECTIVES

1) Minimize time used by user t the best rated movies out of the whole bunch.

2)Find movies similar to what you had watched
before 3) Get the best rated movies out of the
whole bunch

## 1.6 PROPOSED SYSTEM

Hybrid of collaborative and content based filtering algorithm.

In Content based recommendation System, there was no user interference as any user would be recommended the same set of movies having watched similar movies while in Collaborative filtering there was no relations between movies watched by the user like if the user loves watching

Action movies then the recommendation system would not capture his genre interest. Thus the idea of Hybrid Movie Recommendation System came up as it tries to use both Content and Collaborative Filtering methods combining advantages and eliminating disadvantages of both to get the best recommendation outcome for users.

Steps

- Get the index of the movie given its title.
- Get the list of cosine similarity scores for that particular movie with all movies. Convert it into a list of tuples where the first element is its position and the second is the similarity score.
- Sort the list of tuples based on the similarity scores.
- Get the top movies of this list. Ignore the first movie as it refers to self (the movie most similar to a particular movie is the movie itself).
- Apply Svd on to the list of movies for that particular user and sort the recommendation list.
- Return the titles corresponding to the indices of the top elements.

### 1.6.1SYSTEM MODULE

**User module-**this is the module used by users to access the application and get their recommendation.

**Admin module-**this module is where by the admin is able to update and delete data

**Ranking module/rating-**where the movie ratings data set is stored s

**Feedback module –** this is where customers will be able to ask queries in case of any issues.

## 1.7 JUSTIFICATION OF THE STUDY

With the growth of the movie making industry, the number of available movies on the internet has become so vast and diverse that people are finding it hard to get what they want at the appropriate time.This study is relevant that it will allow a user to search and get the content they want from a huge set of movies that is satisfactory from his previous viewing history and his ratings. This will bring a lot of comfort to the user since the movies will be personalised to him/her

## 1.8 SCOPE AND LIMITATION OF THE STUDY

### 1.8.1 scope

The scope of the project is to develop an online recommender application that will be a ble to assist the user too locate the movies they want with ease.

### 1.8.2 limitation of the study

**Cold Start Problem.**The cold start problem is a typical problem in recommendation systems. The “cold start" problem happens in recommendation systems due to the lack of information, on users or items. The Cold-Start problem is a well-known issue in recommendation systems: there is relatively little information about each user, which results in an inability to draw inferences to recommend items to users. The Cold start problem refers to the situation when a new user or item just enters the system. Three kinds of cold start problems are: new user problem, new item problem and new system problem. In such cases, it is really very difficult to provide recommendation as in case of new user, there is very less information about user that is available and also for a new item, no ratings are usually available and thus collaborative filtering cannot make useful recommendations in case of new item as well as new user.

# CHAPTER TWO

## LITERATURE REVIEW

## 2.1 INTRODUCTION

Recommender system [RS] keeps growing in popularity in the world of machine learning and predictive modelling. It is ubiquitous and is used in virtually every industry. Due to its direct impact on the bottom line, RS has been embraced by most of the major corporations in the world. Various movie recommendation techniques have developed by researchers to recommend movies for the user according to their interest or preferences. Since recommendation system are such a hot topic in recent data science research, many scientific articles have been published in the field of recommendation system.

## 2.2 FILTERING AND PREDICTING ONLY THOSE MOVIES THAT A CORRESPONDING USER IS MOST LIKELY TO WANT TO WATCH

In this section we are going to see various techniques and approaches to the implementation of recommender systems. Note however that this section's objective is not to fully define each one of the different approaches, but rather to give a broad overview. For a deep understanding of these concepts, we recommend the recommender systems.

There are several taxonomies for recommender systems but there are five most popular approaches: collaborative, content-based, community-based, demographic, and knowledge based.

### 2.2.1Collaborative filtering

It is the most popular and most implemented approach. In its original and simplest implementation this approach recommends to the active user, items that other users with similar tastes liked in the past. The similarity in taste of two users is calculated based on the similarity in the rating history of the users. In fact, collaborative recommenders exploit the nearest neighbor methods (user-based neighborhood method and item-based neighborhood method). User-based methods rely on the opinion of like-minded users to predict a rating, and generate recommendations. Item-based approaches look at ratings given to similar items and generate recommendations.

So, the first step of Movie-based collaborative-filtering is to find out movies that are similar with what the user liked before. The core point of movie-based collaborative-filtering is to calculate the similarity of two movies. Movie CF considers that movies that are liked by more same users, the more similar they are. Assume N(i) and N(j) are user sets who like i and j respectively. So, the similarity of i and j can be defined as:

$$Sji = \frac{|\mathrm{N(i)} \cap \mathrm{N(j)}|}{|\mathrm{N(i)} \cup \mathrm{N(j)}|}$$

User u’s likeability of item i can be calculated by:

$$pui = \sum_{\mathrm{j} \in \mathrm{S(i,k)} \cap \mathrm{N(u)}}^{\infty} (sijpuj)$$

Table 1 is an example of movie-based CF recommendation. According to the interest history of all the users for movie A, people who like movie A like movie C as well, so we can conclude that movie A is similar with movie C. While User C likes movie A, so we can deduce that perhaps User C likes movie C as well.

| **USER/MOVIE** | **MOVIE A** | **MOVIE B** | **MOVIE C** |
|---|---|---|---|
| USER A | √ | | √ |
| USER B | √ | √ | √ |
| USER C | √ | | recommended |

### 2.2.2 Content-based filtering

The system learns to recommend items that are similar to the ones that the user liked in the past.

It exploits the content of data items to predict its relevance based on the user's profile. For instance, if the active user has rated positively a news movie or series, the system can learn to recommend other news movies and series in the same section. Content-based filtering uses two main methods to generate recommendations:

i. Heuristic-based methods: use common techniques of information retrieval like TF-IDF (term frequency–inverse document frequency), clustering.

ii. Model-based methods: use probabilistic model to learn prediction of the users.

### 2.2.3 Community-based recommender systems

This type of system recommends items based on the preferences of the users friends. This approach follows the principle according to which people tend to rely more on recommendations from their friends than on recommendations from similar but anonymous individuals.

### 2.2.4 Demographic recommender systems

This type of system recommends items based on the demographic profile of the user. The assumption is that different recommendations should be generated for different demographic niches. An example of demographic recommender at work could be the display of ads to users depending on the country they are accessing the system or the language they are speaking.

### 2.2.5 Knowledge based recommender system

Knowledge-based systems recommend items based on specific domain knowledge about how certain item features meet users' needs and preferences and, ultimately, how the item is useful for the user. In such systems a similarity function estimates how much the user needs (problem description) match the recommendations (solutions of the problem).

## 2.3 MINIMIZE TIME USED BY THE USER

Many people have problem selecting the alternative item of movie due to lack of time and due to search issues. Also, movie recommendations from friends can be time consuming. The system helps in saving lots of time.

## 2.4 FINDING SIMILAR MOVIES

Calculate the similarity between movies is the objective of content-based recommender systems. The content can be anything such as text, video and image. In our project, each movie is represented by a feature vector. I will introduce the cosine similarity algorithm then. Cosine Similarity is the most popular measurement for document similarity. In order to calculate the similarity between two features, we can calculate the cosine of the angle between the feature vector using Equation.

$$\text{Similarity=} \cos(\theta) = \frac{A.B}{[A|||B]} = \frac{\sum_{n}^{i=1} Ai \times Bi}{\sqrt{\sum_{n}^{i=1}(Ai)2}\sqrt{\sum_{n}^{i=1}(Bi)2}}$$

### 2.4.1 Recommender engines

As we have seen in the definition above, recommender engines try to infer tastes and preferences and identify unknown movies that are of interest. They are the most immediately recognizable machine learning technique in use today. You probably have already seen services or sites that attempt to recommend books or movies or articles based on your past actions.

### 2.4.2 Clustering

Clustering is less apparent, but it turns up in equally well-known contexts. As its name implies, clustering techniques attempt to group a large number of things together into clusters that share some similarity. It’s a way to discover hierarchy and order in a large or hard-to-understand data set and in that way reveal interesting patterns or make the data set easier to comprehend. Clustering helps identify structure, and even hierarchy, among a large collection of things that may be otherwise difficult to make sense of. Enterprises might use this technique to discover hidden groupings among users, or to organize a large collection of documents sensibly, or to discover common usage patterns for a site based on logs.

### 2.4.3 Classification

Classification techniques decide how much a thing is or isn't part of some type or category, or how much it does or doesn't have some attribute. Classification, like clustering, is ubiquitous, but it's even more behind the scenes. Often these systems learn by reviewing many instances of items in the categories in order to deduce classification rules. Yahoo! Mail has a concrete application of classification. Yahoo! Mail decides whether or not incoming messages are spam based on prior emails and spam reports from users, as well as on characteristics of the email itself. Classification helps decide whether a new input or thing matches a previously observed pattern or not, and it's often used to classify behavior or patterns as unusual. It could be used to detect suspicious network activity or fraud. It might be used to figure out when a user's message indicates frustration or satisfaction.

## 2.5 GETTING BEST RATED MOVIES

Personalized recommender system is a very important application for movie and video website, which can help users to find what they really like among the vast of videos. It can be illustrated that the recommendation algorithm is similar with Amazon according to the recommendation.

### 2.5.1 Choosing a neighborhood

In this technique, the neighbors that we are going to use as a part of prediction also makes an influence on the recommendations that are going to be generated. Consequently, this determination of neighbors must be accomplished more cautiously to not influence the nature of suggestions created. Hence, we will be choosing the most related neighbors who have the highest match compared to others. So, this value must be chosen more delicately.

### 2.5.2 Predicting undetermined ratings

In this for the user, to whom we want to predict those movies for which he hasn't rated, should be predicted using similar weights that are calculated in the previous steps

# CHAPTER THREE

# 3.0 METHODOLOGY

## 3.1 INTRODUCTION

In this chapter ,I will give an explanation, description and procedure used in the study.It also outlines the research methodology employed to meet the specific objectives which in turn help achieve the main objective.

## 3.2 WATERFALL MODEL

The waterfall method has been used in the development of the system. This model has been used because the requirements for this proposed system are clear, well-understood thought and does not require returning to previous step. This method of software development involves completing of one development step followed by next step in sequential manner. The next step is done if the previous one was completed successfully. The output of one phase is input to the next phase. It starts with requirements gathering where requirements are put into consideration and then documented. The second step is analysis; in this phase, the gathered requirements are either validated or invalidated. The third step is design where the design of the system along with hardware and system configuration is done. After the design phase implementation is done where development work is done, it involves actual programming before the developed system is handed to next fifth phase of testing. In testing, each unit of the system is checked if it works as expected and that it is not malfunctioning, here the system is tested against the requirements to ensure it is working as expected. The next phase is deployment where the users can now start using the developed product. The seventh and the last phase of waterfall model in software development is maintenance. In maintenance, there is provision of all necessary solutions to issues arising as reported by end users; it occurs also when the product is updated depending on the needs that shall arise.

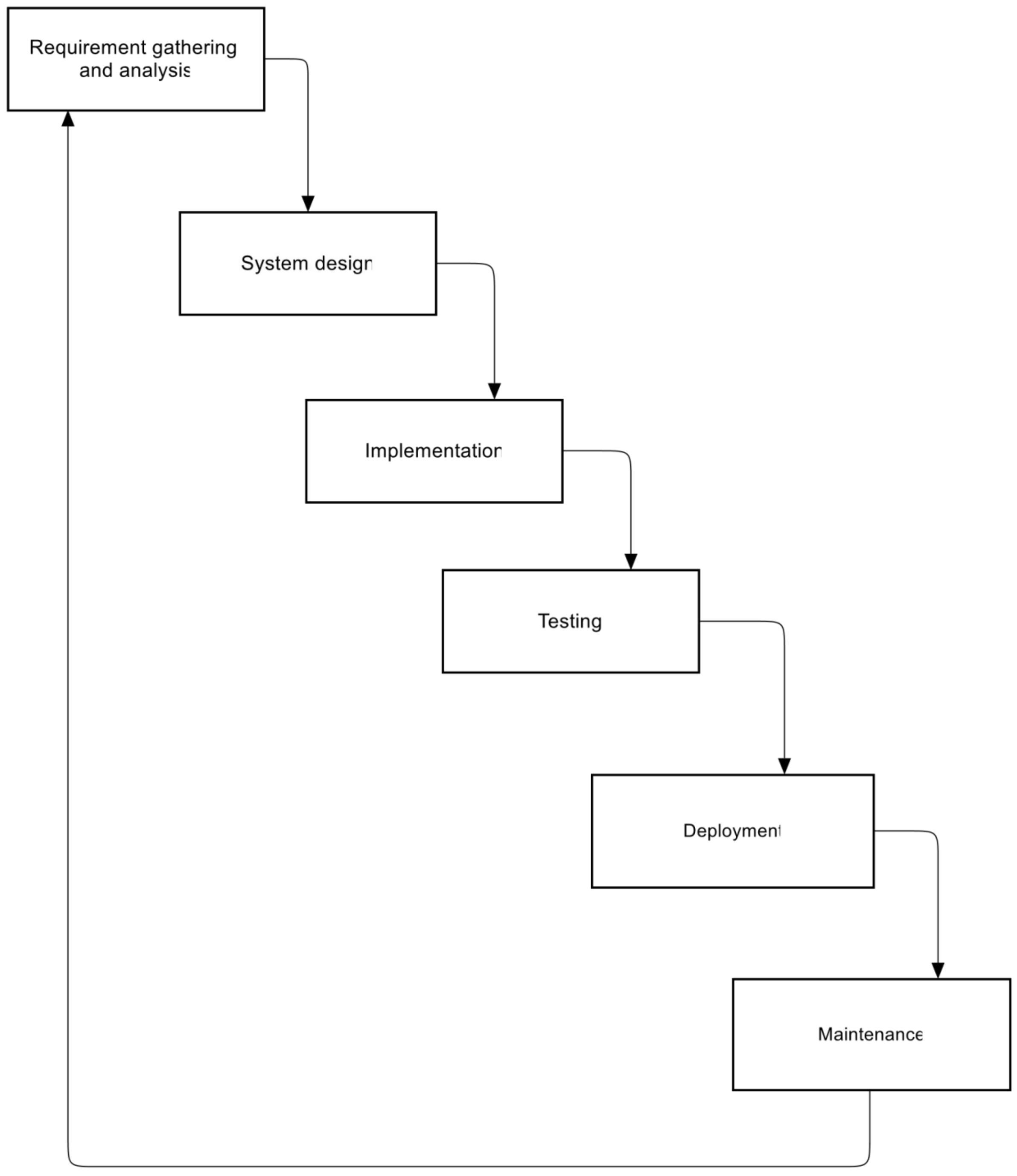


Figure 1: Waterfall model

## *3.3* DATA COLLECTION METHODS

This section outlines the various data collection instruments that will be used in research and the analysis of the data collected. This will involve finding of facts through collection of information on how business transactions are carried out, how customers are made aware of products being sold, how information is handled in the current system. The two types of data collection tools that will be used are:

- Primary data collection.

- Secondary data collection.

### 3.3.1 PRIMARY DATA COLLECTION

It looks into collection of first-hand information from users of the current system and the following tools will be used in conducting these research.

- ✓ Questionnaires.
- ✓ Interview..

1) Interviews In this research, interviews will be conducted by asking some customers a series of questions. From their responses analysis of the data will be made.The following is a sample of interview questions that will be addressed to the interviewees.

2) Questionnaire A series of printed questions will be handed out to the entrepreneurs that will give more information on the current system and also specification of intended system. The questions will be open and closed ended in order to provide detailed responses; this will provide a clear understanding about the current system and also get a detailed response on the proposed system.

### 3.3.2 SECONDARY DATA COLLECTION

The Internet The use of the internet will also enable comprehensive study of the flexibility of the emarketing systems.

Books and journals will also contribute to the collection of the required information

## 3.4 SYSTEM REQUIREMENTS

### 3.4.1 HARDWARE

Processor: 2.5GHZ `

Ram :4GB minimum

Hard disk :500GB minimum

Keyboard: normal

### 3.4.2 SOFTWARE

Operating system: Windows 11/10

Firebase for storages

## 3.5 DESIGN DIAGRAMS

### 3.5.1 CONTEXT DIAGRAMS

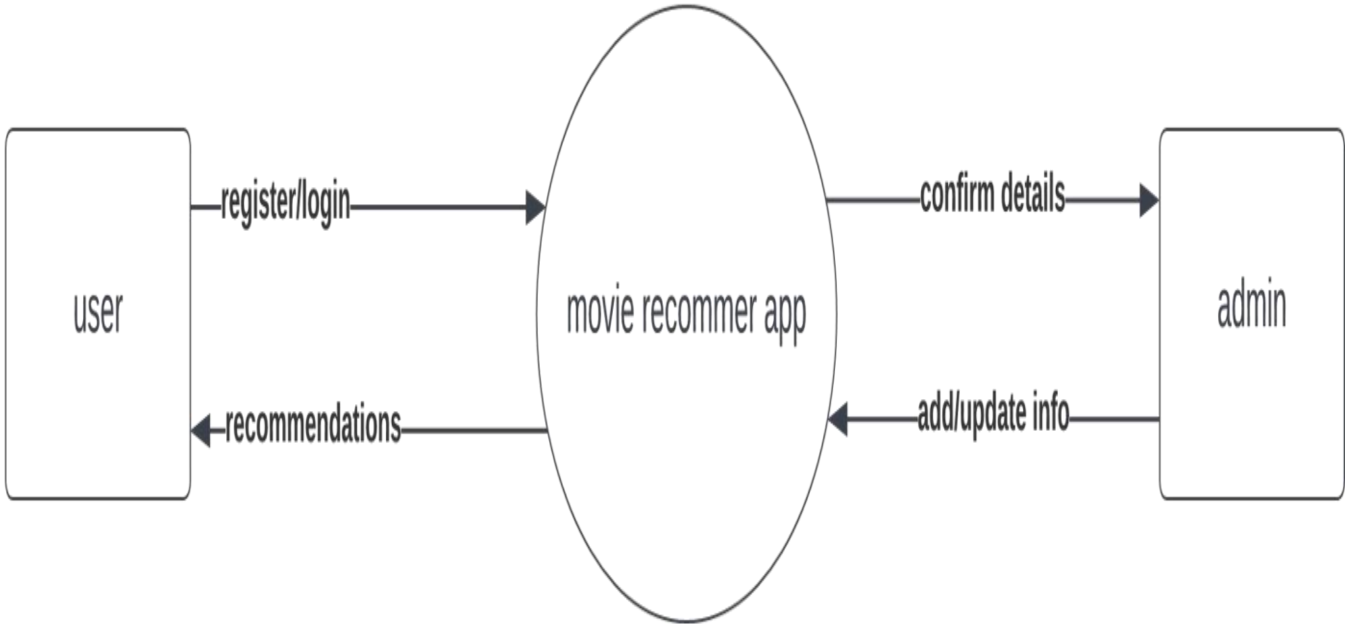


*Figure 2*

### 3.5.2 USE CASE DIAGRAM

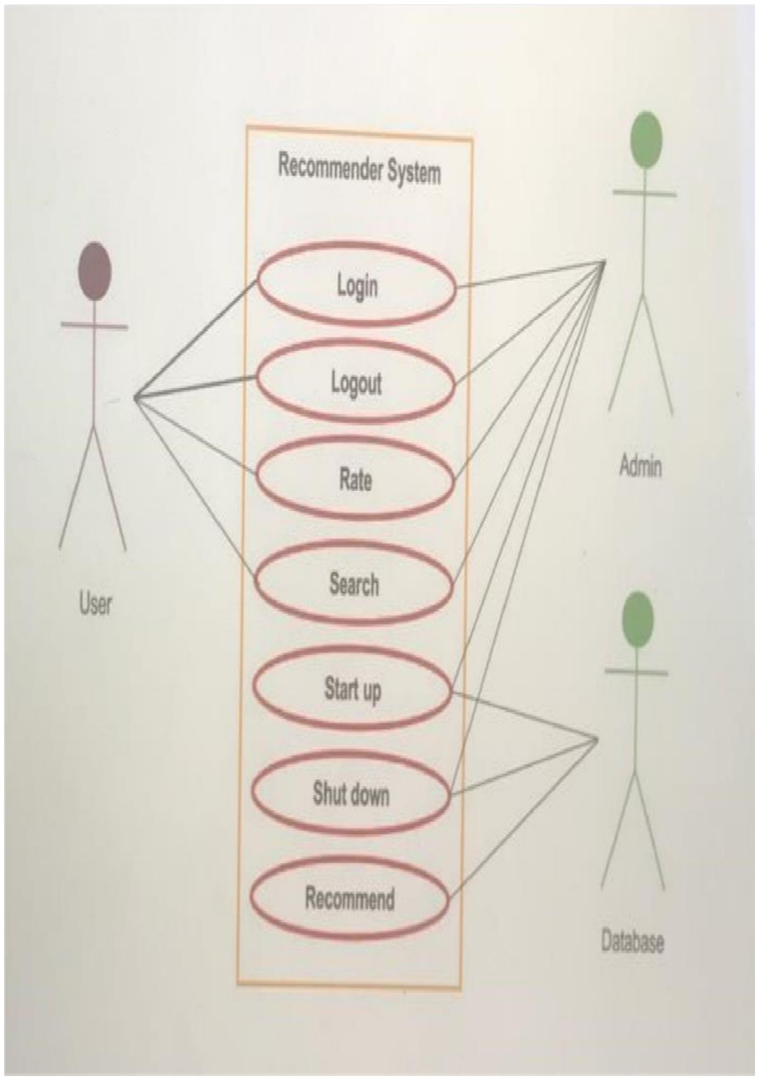


*Figure 3*

## 3.5.3 DATA FLOW DIAGRAM

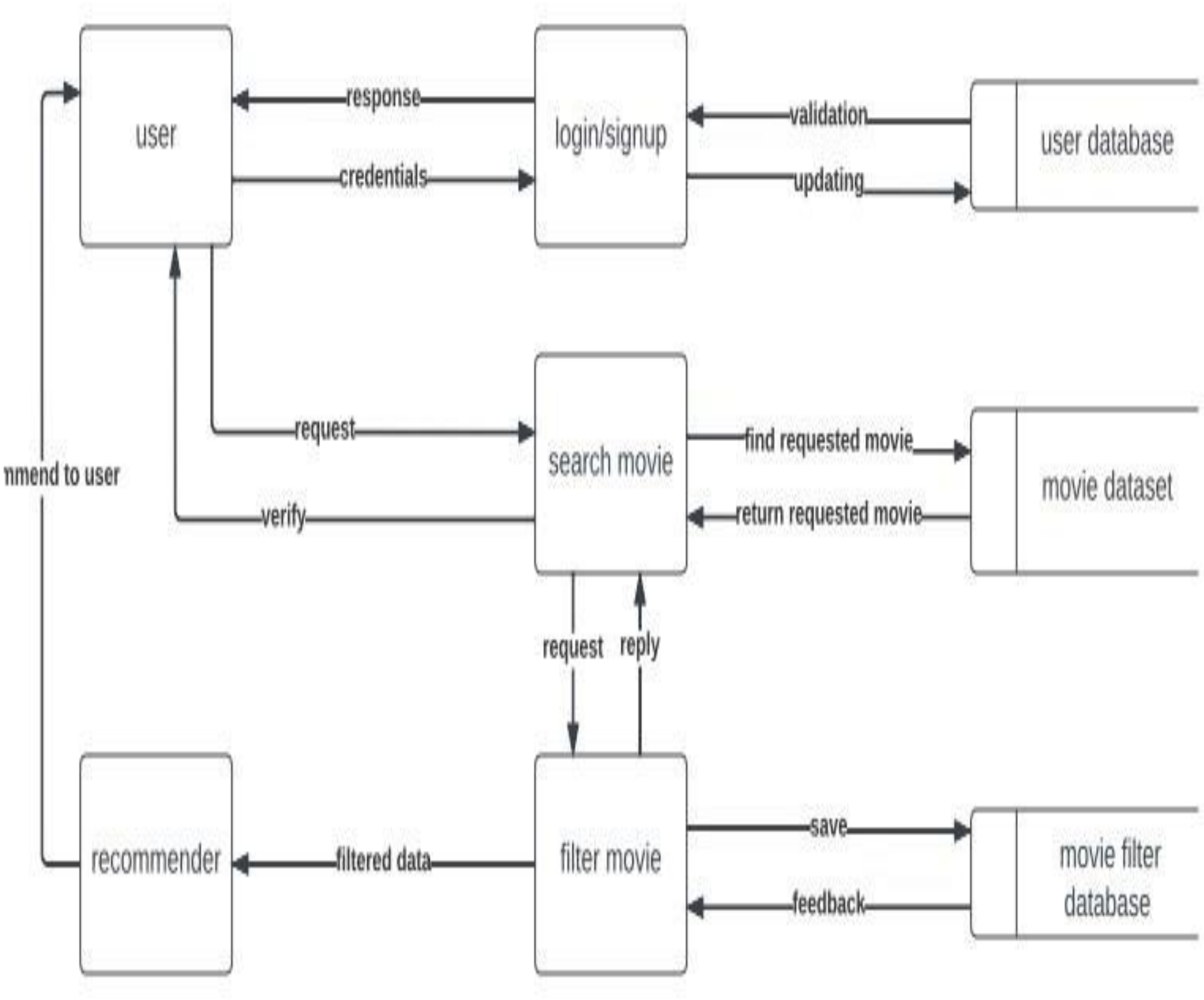


*Figure 4*

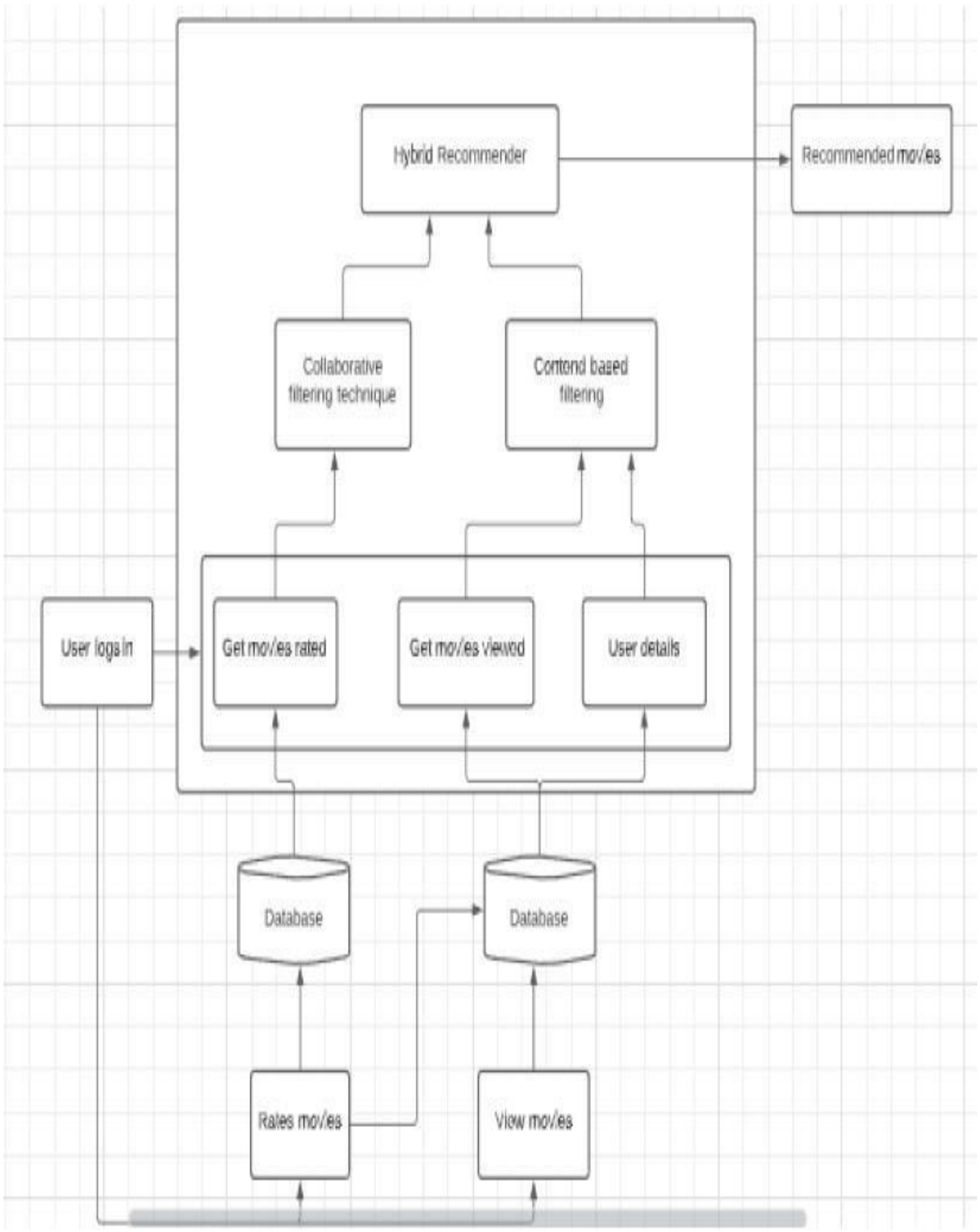


*Figure 5 system overview*

## 3.6 RESEARCH ETHICS

When carrying out the research, I will observe the various research ethics such as conserving anonymity for people who do not wish to be mentioned and also mentioning and citing publishers and other researchers where I have gotten some information from there works

# CHAPTER 4:SYSTEM IMPLEMENTSATION AND DEVELOMENT

### 4.1.1 System Description

The movie recommendation application is designed to provide users with personalized movie suggestions, enhance their movie discovery experience, and reduce the time they spend searching for information within a vast collection of movies. The application caters to both general users and movie enthusiasts, allowing anyone with a smartphone to enjoy its benefits.

The application's key objectives include providing personalized movie suggestions through a hybrid recommendation algorithm that combines content-based and collaborative filtering techniques, offering an exciting way for users to discover new movies that align with their tastes, and enabling efficient information retrieval by integrating data from the Movie Database and the Movielens dataset.

The application also offers several key features, including user registration and sign-up, a powerful movie search function, the ability for users to rate movies they have watched, and the display of ratings from the TMDB database. The application is developed using Flutter, a versatile cross-platform framework that primarily focuses on the Android platform.

In terms of user experience goals, the application aims to be simple to use, visually appealing and visible, and responsive. This ensures that users can easily explore movie recommendations and features without encountering unnecessary complexities, while also enjoying a visually engaging and well-designed user interface and swift response times.

Overall, this overview provides a high-level understanding of the movie recommendation application, highlighting its purpose, target audience, recommendation algorithm, data sources, key features.

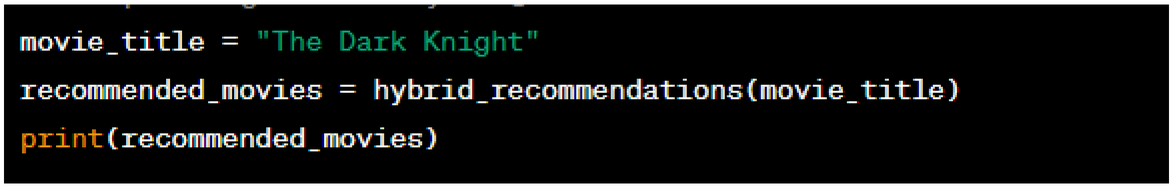
The movie recommendation application uses a hybrid recommendation algorithm that combines the strengths of both content-based and collaborative filtering techniques to provide users with personalized movie suggestions.

Content-based filtering involves preprocessing the movie genres from the ‘genres’ column, converting the genre information into a numerical representation using a Term Frequency-Inverse Document Frequency (TF-IDF) vectorizer, and calculating cosine similarity to measure the similarity between movies based on their genre representations. When a user searches for or views a movie, the content-based filtering component recommends movies with similar genres.

Collaborative filtering involves constructing a user-item matrix using ratings data from the ‘ratings.csv’ file, calculating user similarity using Pearson correlation, and recommending movies based on other users’ preferences and ratings. When a user rates or interacts with a movie, the collaborative filtering component suggests movies that similar users have enjoyed.

The hybrid recommendation algorithm combines the results of both content-based and collaborative filtering techniques to produce more accurate and personalized movie suggestions. Given a movie title as input, the algorithm identifies similar movies using content-based filtering, ranks them based on their content similarity, applies collaborative filtering to gather additional movie recommendations based on user preferences, and combines both scores using a weighted average approach to produce the final movie suggestions.

An example usage of the hybrid_recommendations function is provided, which returns a DataFrame with the top movie recommendations ranked by their hybrid scores



```
movie_title = "The Dark Knight"
recommended_movies = hybrid_recommendations(movie_title)
print(recommended_movies)
```

*figure 6 example query*

## 4.2 FRONT END DEVELOPMENT

The Front End Development and User Interface (UI) Design of the movie recommendation application involved designing the UI, including creating wireframes and mockups, and implementing the Front End using Flutter.

During the UI Design phase, design specifications such as color schemes, typography, iconography, and user experience goals were considered to create a simple, intuitive, and visually appealing interface. The Front End Development utilized Flutter to build the application for Android, using Dart to implement various widgets and functionalities for the user interface.

The application's key components include the homepage, which features a clean layout and a search bar for finding movies easily; the movie search functionality, which enables users to quickly find movies by entering titles, genres, or keywords; the MediaDetailsPage widget, which displays comprehensive movie details; the ability for registered users to rate movies they have watched; and personalized movie recommendations based on user interactions and preferences,I also intergrated fetching movies according to genres if one wants a specific genre to see what's there.

In terms of user experience goals, the Front End emphasizes intuitive navigation, a visually appealing interface, and efficient interaction. This ensures that users can easily explore movie recommendations and features without encountering any confusion while also enjoying visually appealing graphics, well-organized layouts, quick response times, and seamless interactions as you will see below.

## 4.3 INTERFACE MODULES

The movie recommendation application includes several key interface modules that contribute to the overall user experience. These include the homepage with movie search, the movie details display, and the recommendations section,the watchlist page and the movies page.

The homepage features a user-friendly interface with a clean layout and a prominently displayed search bar that allows users to quickly find movies. Real-time search suggestions are provided to enhance the search experience and aid users in discovering relevant movies. A code snippet is provided to illustrate the implementation of the homepage with movie search using Flutter.

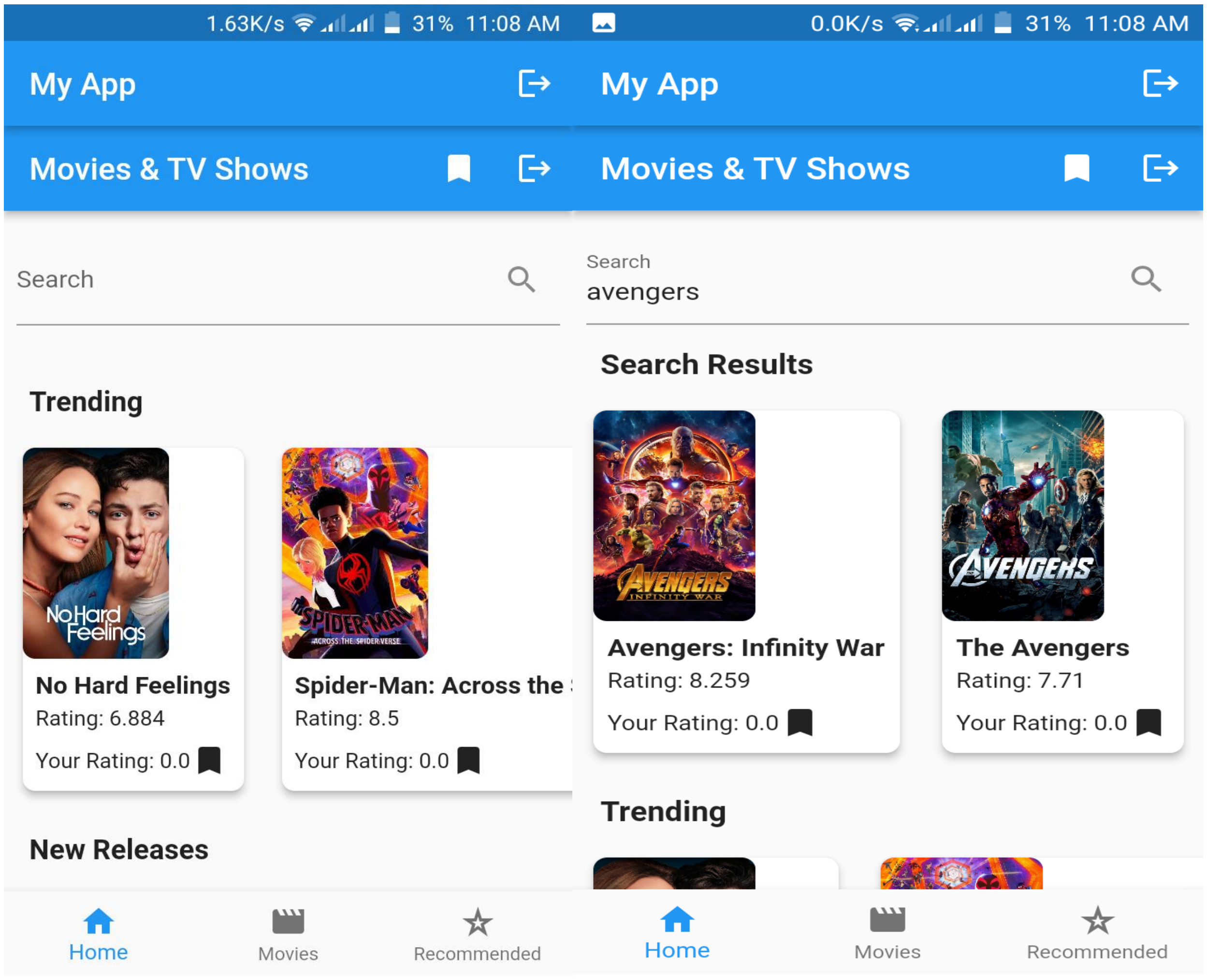
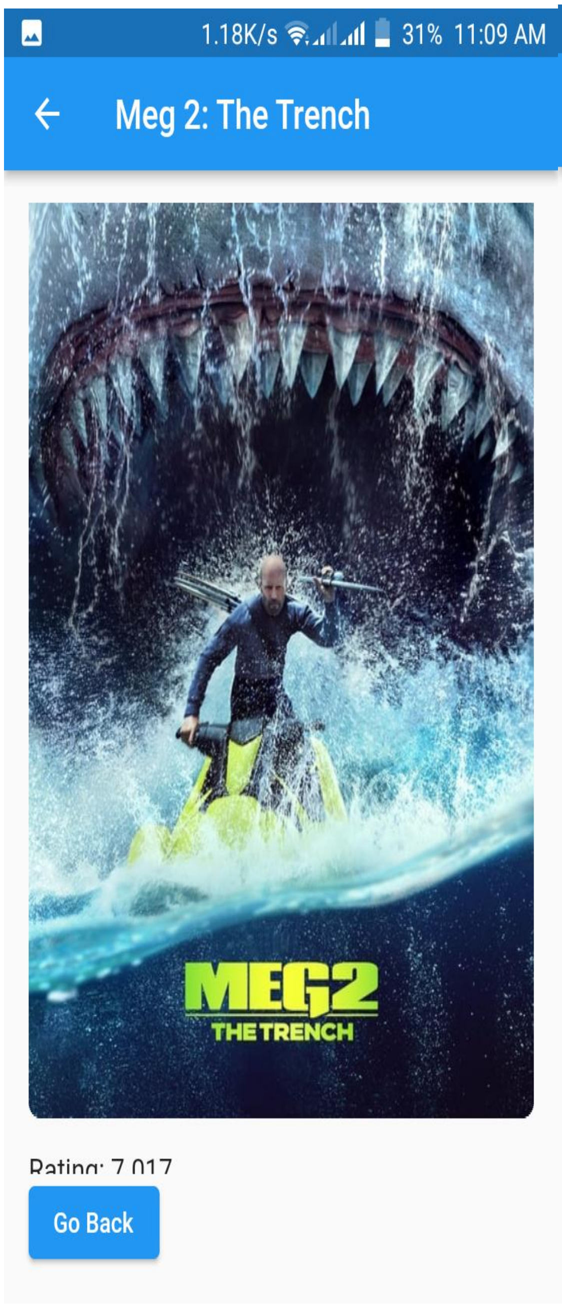
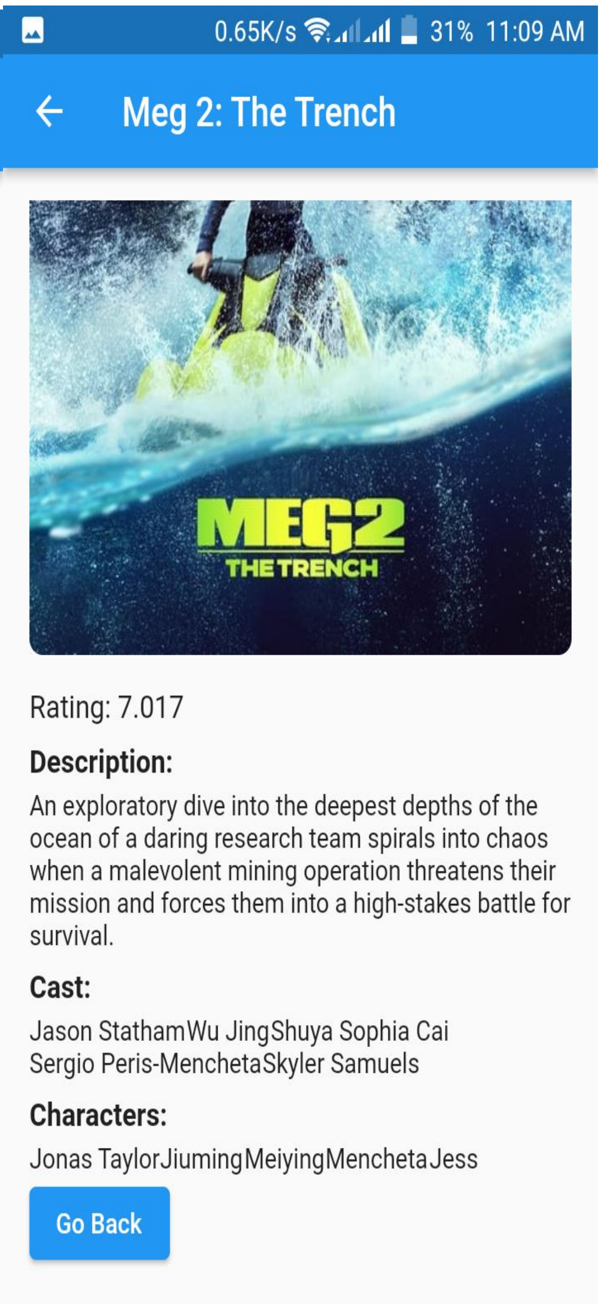


*figure 7 and 8:home page and search 1*

The MediaDetailsPage widget provides detailed information about movies, including their title, poster, rating, description, cast, and characters. The movie's poster is displayed using a ClipRRect with rounded corners for an appealing visual presentation, and users can read reviews and ratings from other users to make informed decisions. A code snippet is provided to illustrate the implementation of the MediaDetailsPage widget using Flutter.



Rating: 7.017



Rating: 7.017

**Description:**

An exploratory dive into the deepest depths of the ocean of a daring research team spirals into chaos when a malevolent mining operation threatens their mission and forces them into a high-stakes battle for survival.

**Cast:**

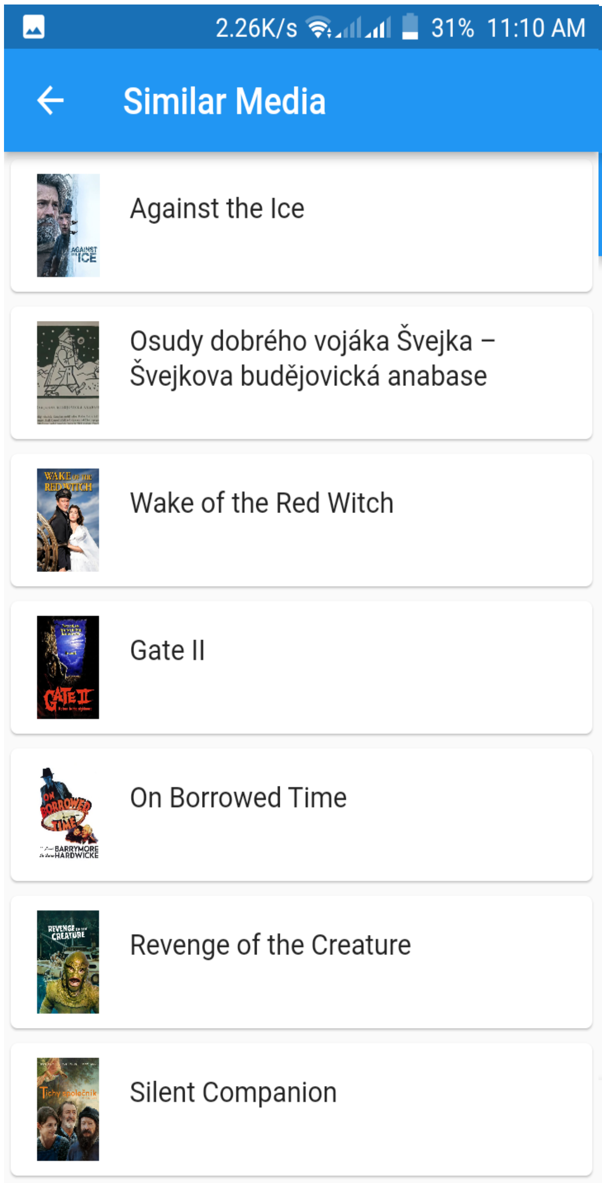
Jason StathamWu JingShuya Sophia Cai
Sergio Peris-MenchetaSkyler Samuels

**Characters:**

Jonas TaylorJiumingMeiyingMenchetaJess

*figure 8 and 9.Mediadetails page 1*

The recommendations section of the application provides personalized movie suggestions based on user interactions and preferences. The section showcases a curated list of movies tailored to individual user interests. A code snippet is provided to illustrate the implementation of the recommendations section using Flutter.(remember to add screenshots and codes if necessary)


2.26K/s 31% 11:10 AM

Similar Media

Against the Ice

Osudy dobrého vojáka Švejka – Švejkova budějovická anabase

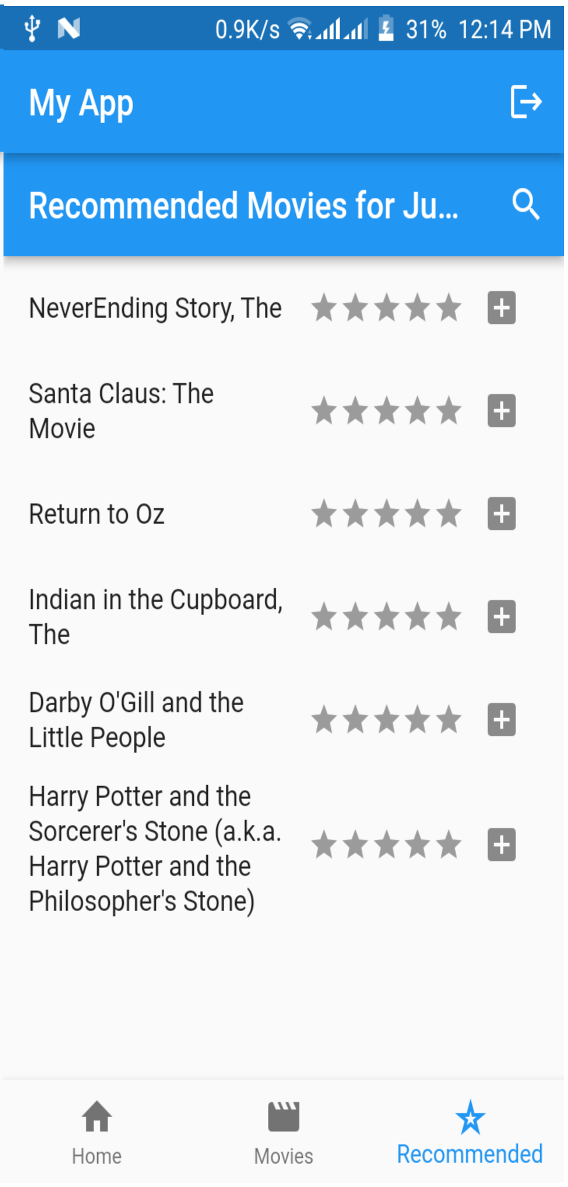
Wake of the Red Witch

Gate II

On Borrowed Time

Revenge of the Creature

Silent Companion


0.9K/s 31% 12:14 PM

My App

Recommended Movies for Ju...

NeverEnding Story, The ★★★★★

Santa Claus: The Movie ★★★★★

Return to Oz ★★★★★

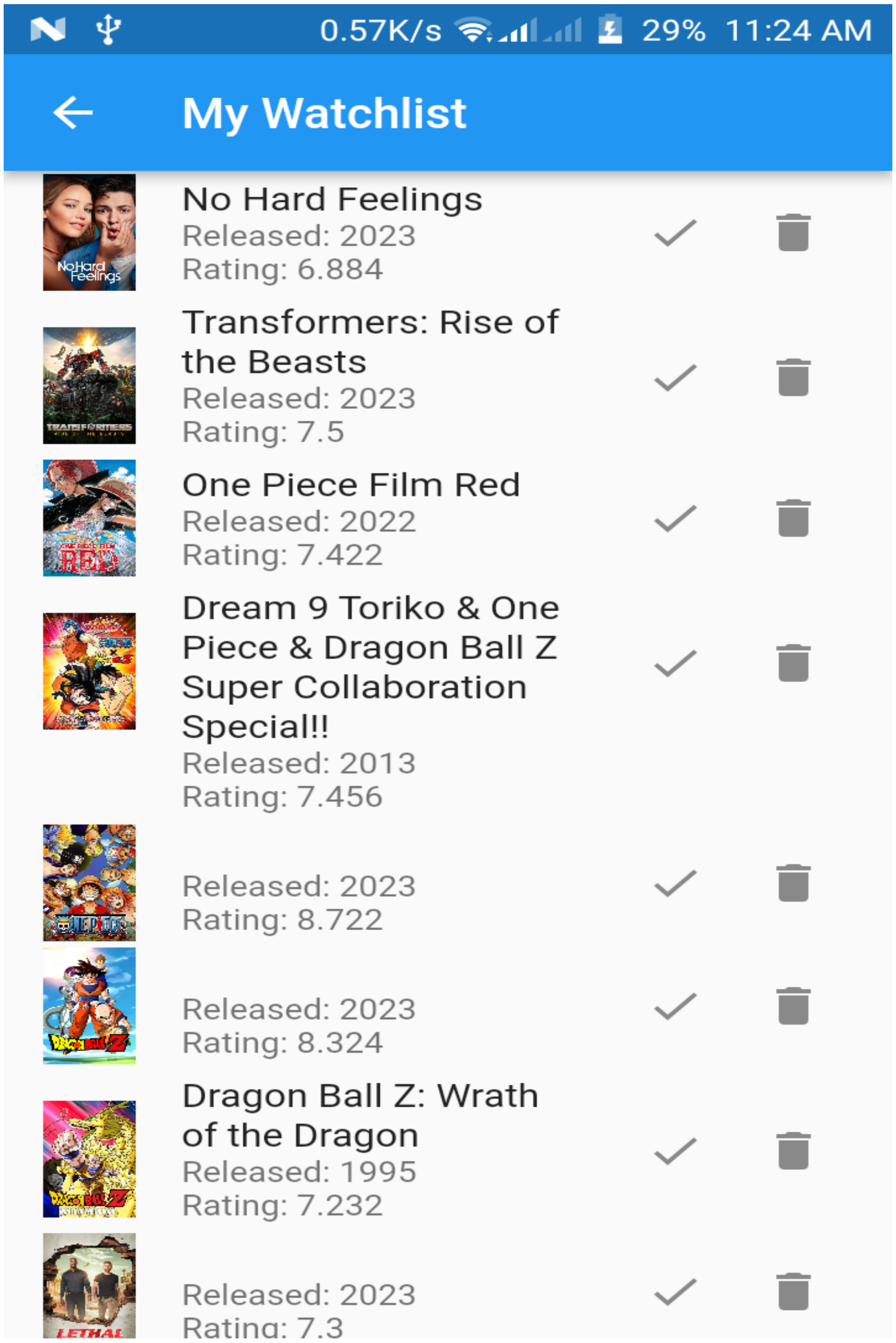
Indian in the Cupboard, The ★★★★★

Darby O'Gill and the Little People ★★★★★

Harry Potter and the Sorcerer's Stone (a.k.a. Harry Potter and the Philosopher's Stone) ★★★★★

*figure 10 and 11:Recommendations 1*

The watchlist performs the action of saving the movie that the user likes on the phone and the user is able to remove and add as they please and alsso do ratings.this figure shows the watchlist page.



No Hard Feelings
Released: 2023
Rating: 6.884

Transformers: Rise of the Beasts
Released: 2023
Rating: 7.5

One Piece Film Red
Released: 2022
Rating: 7.422

Dream 9 Toriko & One Piece & Dragon Ball Z Super Collaboration Special!!
Released: 2013
Rating: 7.456

Released: 2023
Rating: 8.722

Released: 2023
Rating: 8.324

Dragon Ball Z: Wrath of the Dragon
Released: 1995
Rating: 7.232

Released: 2023
Rating: 7.3

*figure 12:Watchlist screen 1*

The movies page is used to show the movies present ie the trending,newly released,recommended media and upcoming media.the movies page looks like this ie

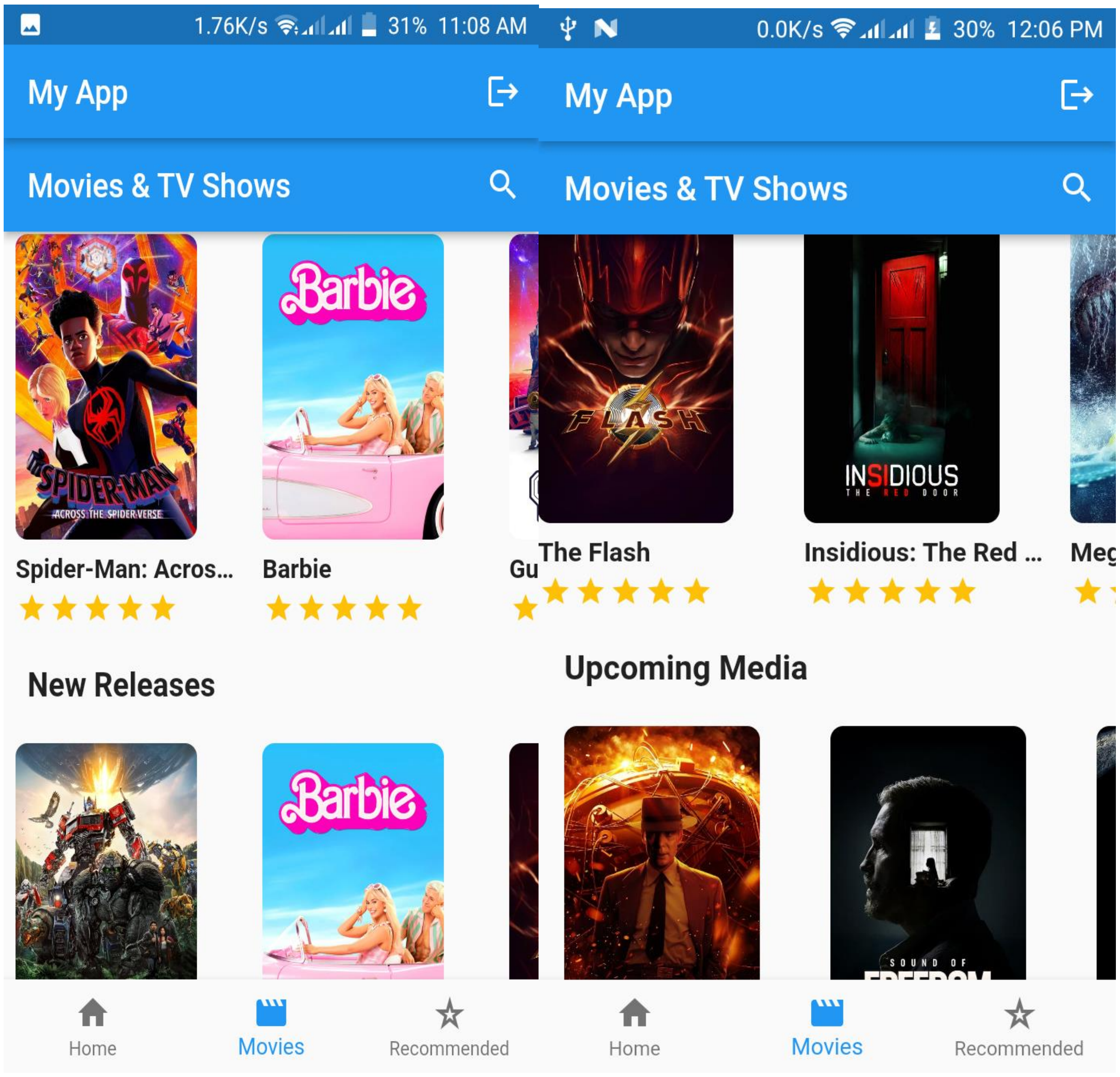


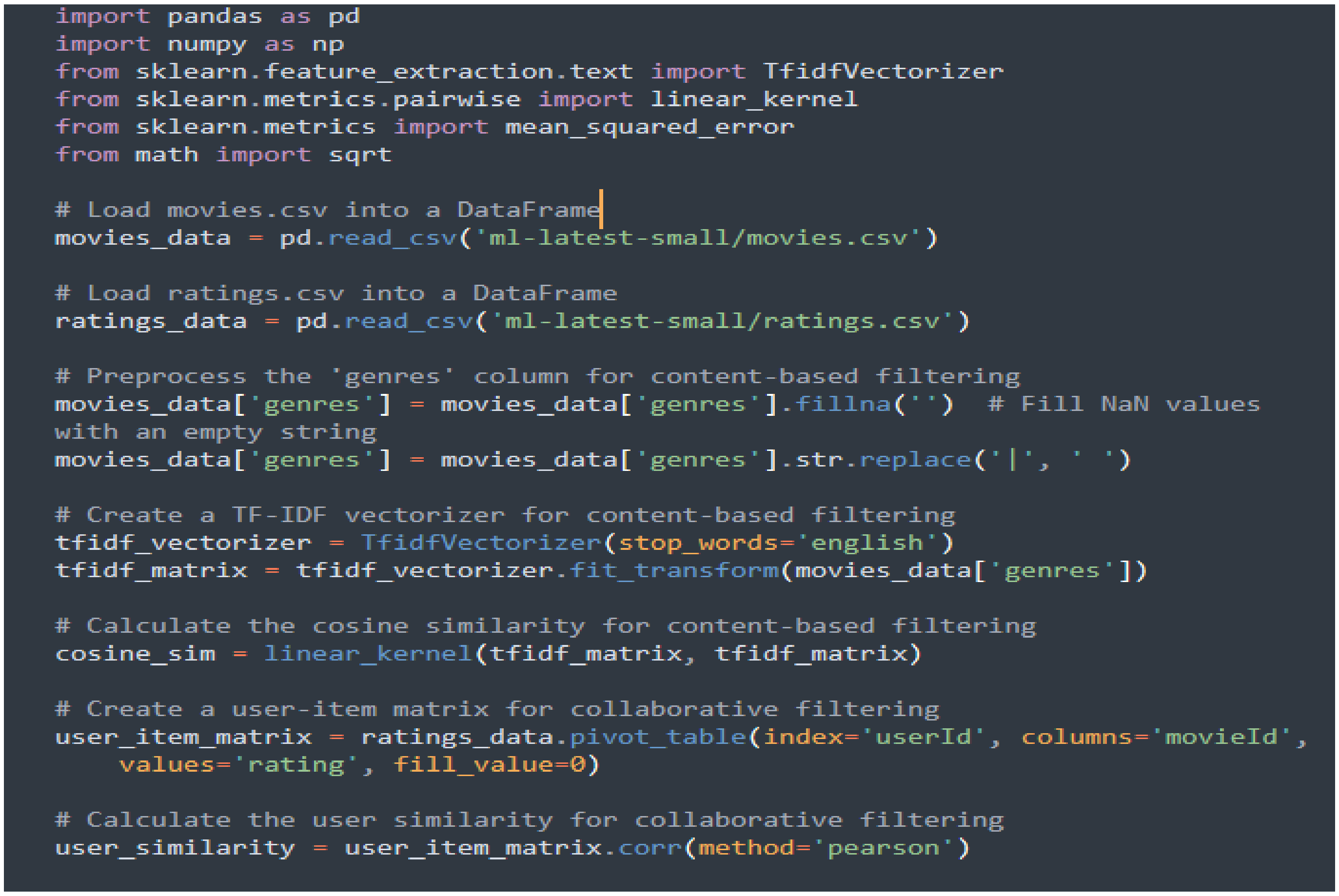
*figure 13:movies page 1*

Overall, these interface modules play an important role in enhancing the user experience of the movie recommendation application

## 4.4 Backend Development

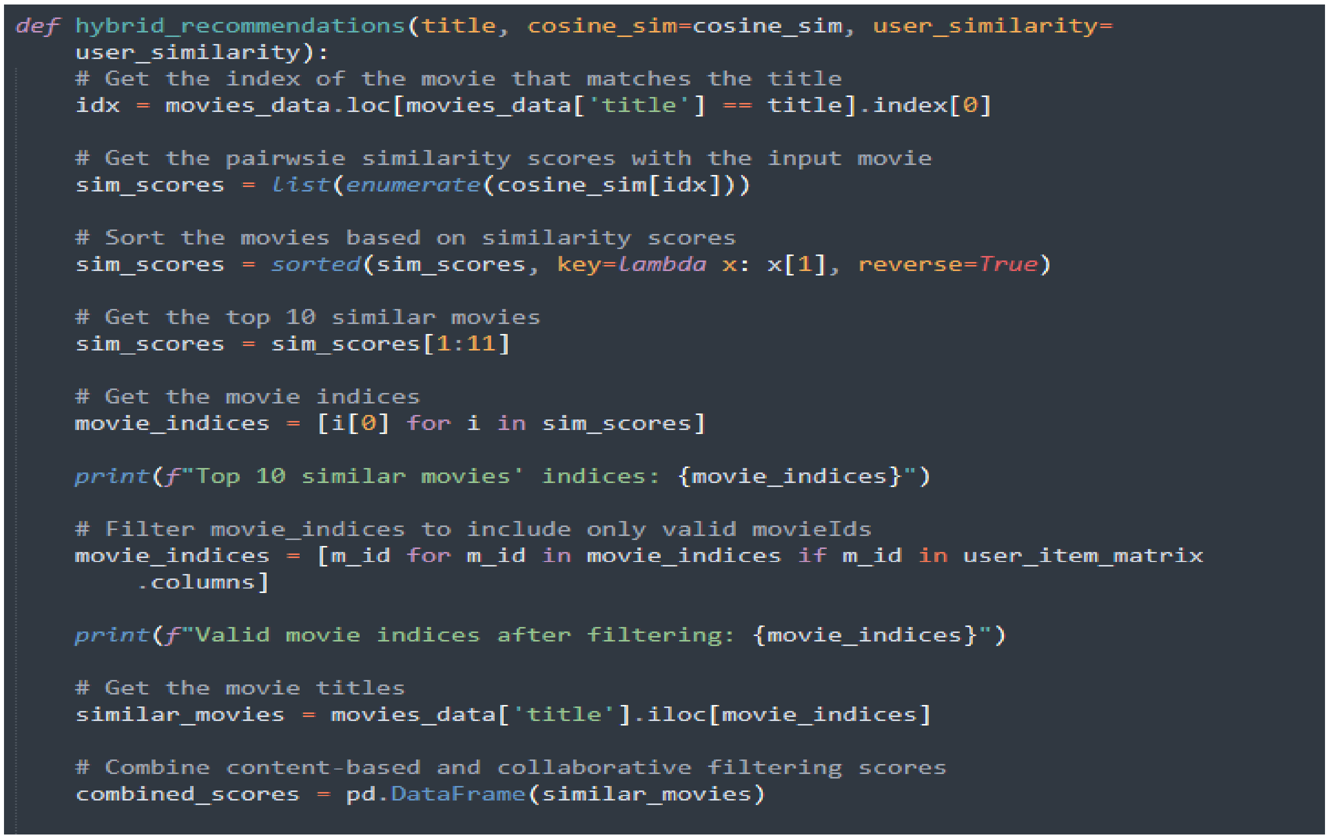
The Backend Development of the movie recommendation application involves using Python, Flask, and other libraries to create an API that provides movie recommendations based on the hybrid recommendation algorithm. Additionally, Firebase is used to store user information and authentication.

The data_preparation.py script is used to prepare the movie and rating data for the recommendation algorithm by processing the 'movies.csv' and 'ratings.csv' files from the MovieLens dataset. The movierecommender.py module contains the implementation of the hybrid recommendation algorithm, which combines content-based filtering and collaborative filtering to generate movie recommendations.the code is like this ie;



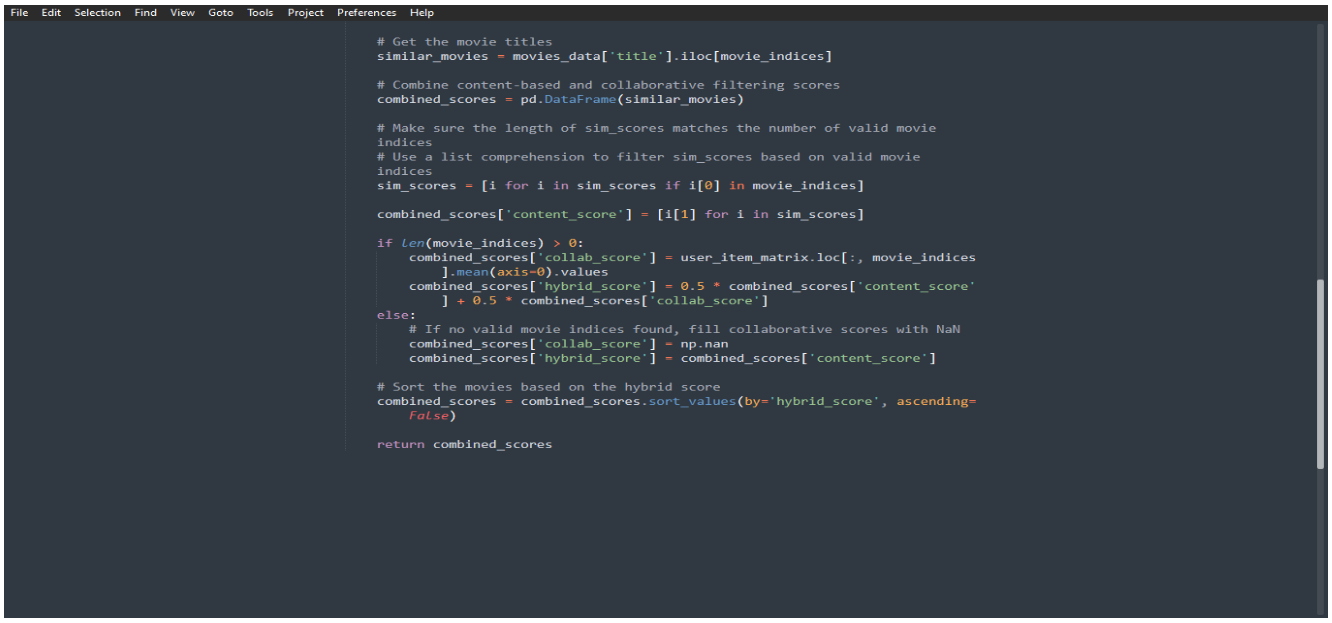
```
import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from sklearn.metrics import mean_squared_error
from math import sqrt

# Load movies.csv into a DataFrame
movies_data = pd.read_csv('ml-latest-small/movies.csv')

# Load ratings.csv into a DataFrame
ratings_data = pd.read_csv('ml-latest-small/ratings.csv')

# Preprocess the 'genres' column for content-based filtering
movies_data['genres'] = movies_data['genres'].fillna('')  # Fill NaN values
with an empty string
movies_data['genres'] = movies_data['genres'].str.replace('|', ' ')

# Create a TF-IDF vectorizer for content-based filtering
tfidf_vectorizer = TfidfVectorizer(stop_words='english')
tfidf_matrix = tfidf_vectorizer.fit_transform(movies_data['genres'])

# Calculate the cosine similarity for content-based filtering
cosine_sim = linear_kernel(tfidf_matrix, tfidf_matrix)

# Create a user-item matrix for collaborative filtering
user_item_matrix = ratings_data.pivot_table(index='userId', columns='movieId',
    values='rating', fill_value=0)

# Calculate the user similarity for collaborative filtering
user_similarity = user_item_matrix.corr(method='pearson')
```

*code 1 for movierecommendation.py 1*



```python
def hybrid_recommendations(title, cosine_sim=cosine_sim, user_similarity=
    user_similarity):
    # Get the index of the movie that matches the title
    idx = movies_data.loc[movies_data['title'] == title].index[0]

    # Get the pairwsie similarity scores with the input movie
    sim_scores = list(enumerate(cosine_sim[idx]))

    # Sort the movies based on similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)

    # Get the top 10 similar movies
    sim_scores = sim_scores[1:11]

    # Get the movie indices
    movie_indices = [i[0] for i in sim_scores]

    print(f"Top 10 similar movies' indices: {movie_indices}")

    # Filter movie_indices to include only valid movieIds
    movie_indices = [m_id for m_id in movie_indices if m_id in user_item_matrix
        .columns]

    print(f"Valid movie indices after filtering: {movie_indices}")

    # Get the movie titles
    similar_movies = movies_data['title'].iloc[movie_indices]

    # Combine content-based and collaborative filtering scores
    combined_scores = pd.DataFrame(similar_movies)
```

*code 2 for movierecommender.py 1*

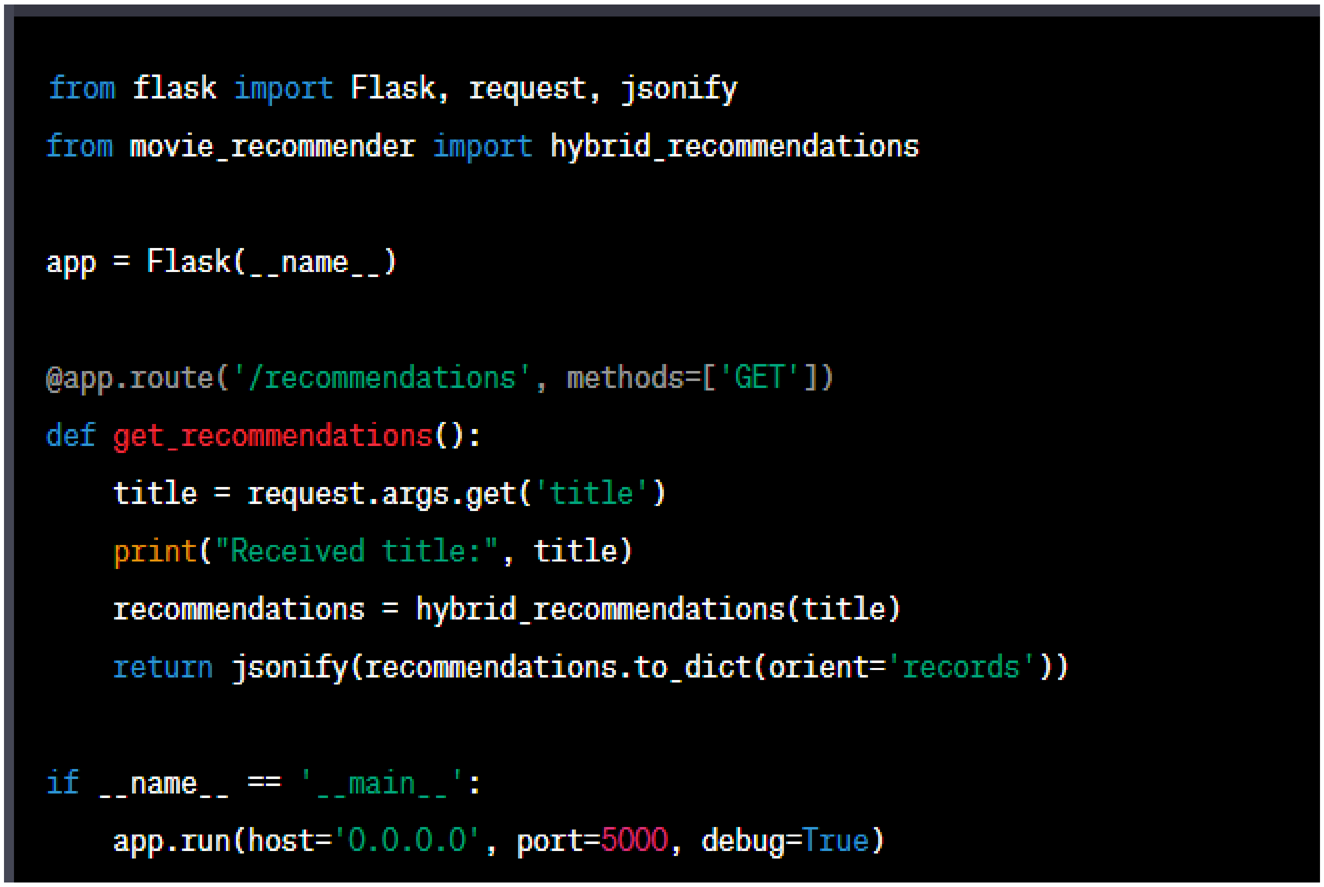


File Edit Selection Find View Goto Tools Project Preferences Help

```python
    # Get the movie titles
    similar_movies = movies_data['title'].iloc[movie_indices]

    # Combine content-based and collaborative filtering scores
    combined_scores = pd.DataFrame(similar_movies)

    # Make sure the length of sim_scores matches the number of valid movie
    indices
    # Use a list comprehension to filter sim_scores based on valid movie
    indices
    sim_scores = [i for i in sim_scores if i[0] in movie_indices]

    combined_scores['content_score'] = [i[1] for i in sim_scores]

    if len(movie_indices) > 0:
        combined_scores['collab_score'] = user_item_matrix.loc[:, movie_indices
            ].mean(axis=0).values
        combined_scores['hybrid_score'] = 0.5 * combined_scores['content_score'
            ] + 0.5 * combined_scores['collab_score']
    else:
        # If no valid movie indices found, fill collaborative scores with NaN
        combined_scores['collab_score'] = np.nan
        combined_scores['hybrid_score'] = combined_scores['content_score']

    # Sort the movies based on the hybrid score
    combined_scores = combined_scores.sort_values(by='hybrid_score', ascending=
        False)

    return combined_scores
```

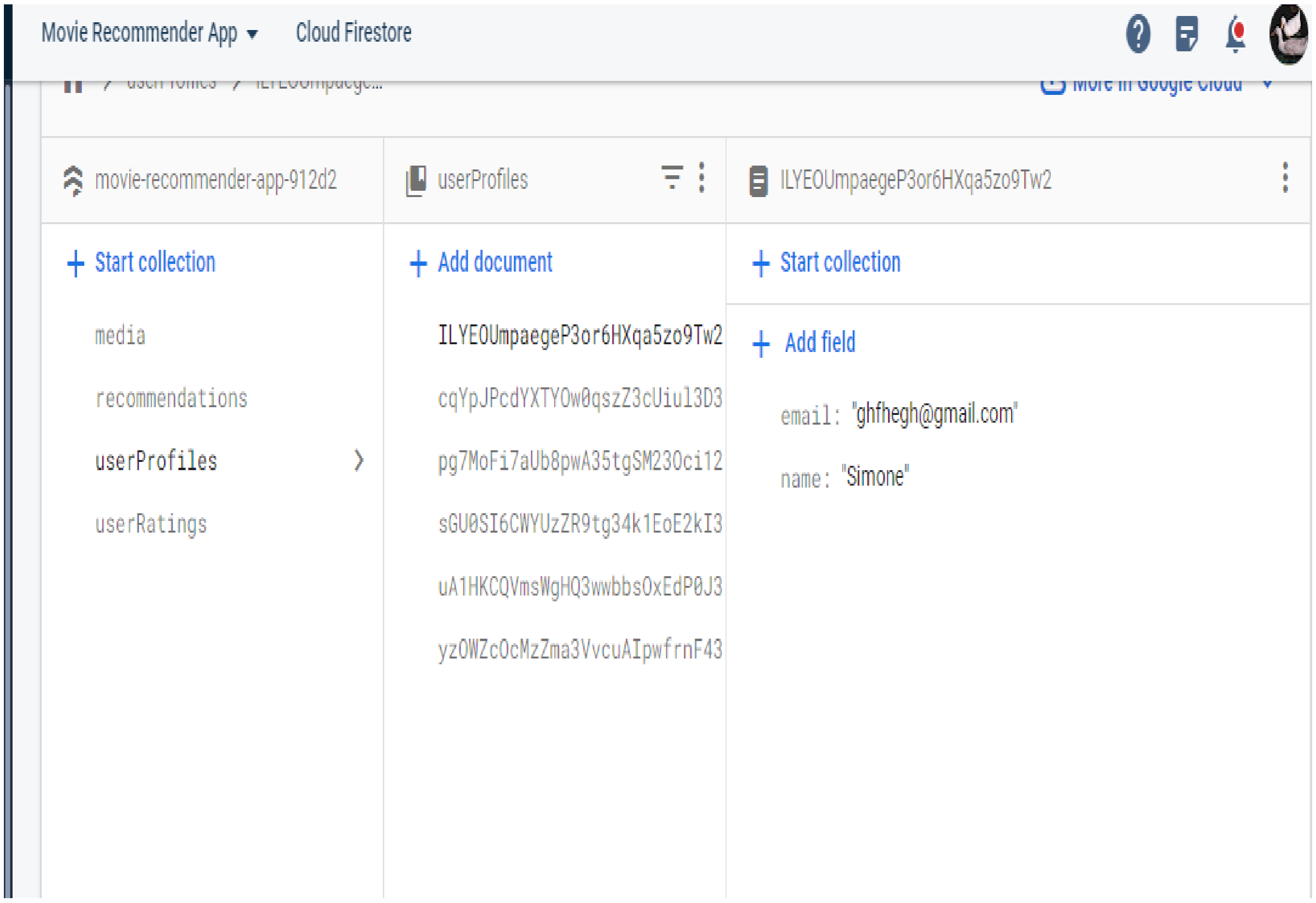
*code 3 for movierecommender.py 1*

The app.py script implements a Flask application that creates an API endpoint to provide movie recommendations to the Flutter mobile application. The ‘/recommendations’ route handles movie recommendations and accepts a movie title as a query parameter. When a GET request with a movie title is received, the hybrid_recommendations function is called to generate movie recommendations based on the provided title, and the recommendations are returned as a JSON response to the Flutter app,as shown below ie ;



```python
from flask import Flask, request, jsonify
from movie_recommender import hybrid_recommendations

app = Flask(__name__)

@app.route('/recommendations', methods=['GET'])
def get_recommendations():
    title = request.args.get('title')
    print("Received title:", title)
    recommendations = hybrid_recommendations(title)
    return jsonify(recommendations.to_dict(orient='records'))

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

Firebase is integrated into the application to store user information, including user IDs,ratingsUser profiles,and authentication. This allows users to create accounts and access enhanced features such as movie recommendations.



movie-recommender-app-912d2

media
recommendations
userProfiles
userRatings

userProfiles

ILYEOUmpaegeP3or6HXqa5zo9Tw2
cqYpJPcdYXTYOw0qszZ3cUiul3D3
pg7MoFi7aUb8pwA35tgSM230ci12
sGU0SI6CWYUzZR9tg34k1EoE2kI3
uA1HKCQVmsWgHQ3wwbbsOxEdP0J3
yzOWZcOcMzZma3VvcuAIpwfrnF43

ILYEOUmpaegeP3or6HXqa5zo9Tw2

email: "ghfhegh@gmail.com"
name: "Simone"

*Collections in Cloud Firestore 1*

### 4.4.1Data Preparation(Flask)

Before implementing the hybrid recommendation algorithm in movierecommender.py, the data needs to be prepared and preprocessed to ensure its suitability for recommendation tasks. The datapreparation.py script performs the necessary steps to explore and process the datasets from the MovieLens dataset.

#### 4.4.1.1Data Exploration and Loading:

The script loads multiple CSV files, including 'genome_scores.csv', 'movies.csv', 'ratings.csv', 'genome_tags.csv', 'links.csv', and 'tags.csv' into separate pandas DataFrames.

For each DataFrame, the script displays its first few rows, shape, data types, and basic statistics to gain insights into the dataset's structure and characteristics.

#### 4.4.1.2 Data Preprocessing for Content-Based Filtering:

In movierecommender.py, content-based filtering relies on movie genres for similarity calculations.

The 'movies.csv' file is preprocessed to ensure the genres are in a suitable format for content-based filtering.

The 'genres' column is modified to replace '|' separators with spaces, enabling proper genre representation.

#### 4.4.1.3 Data Preparation for Collaborative Filtering:

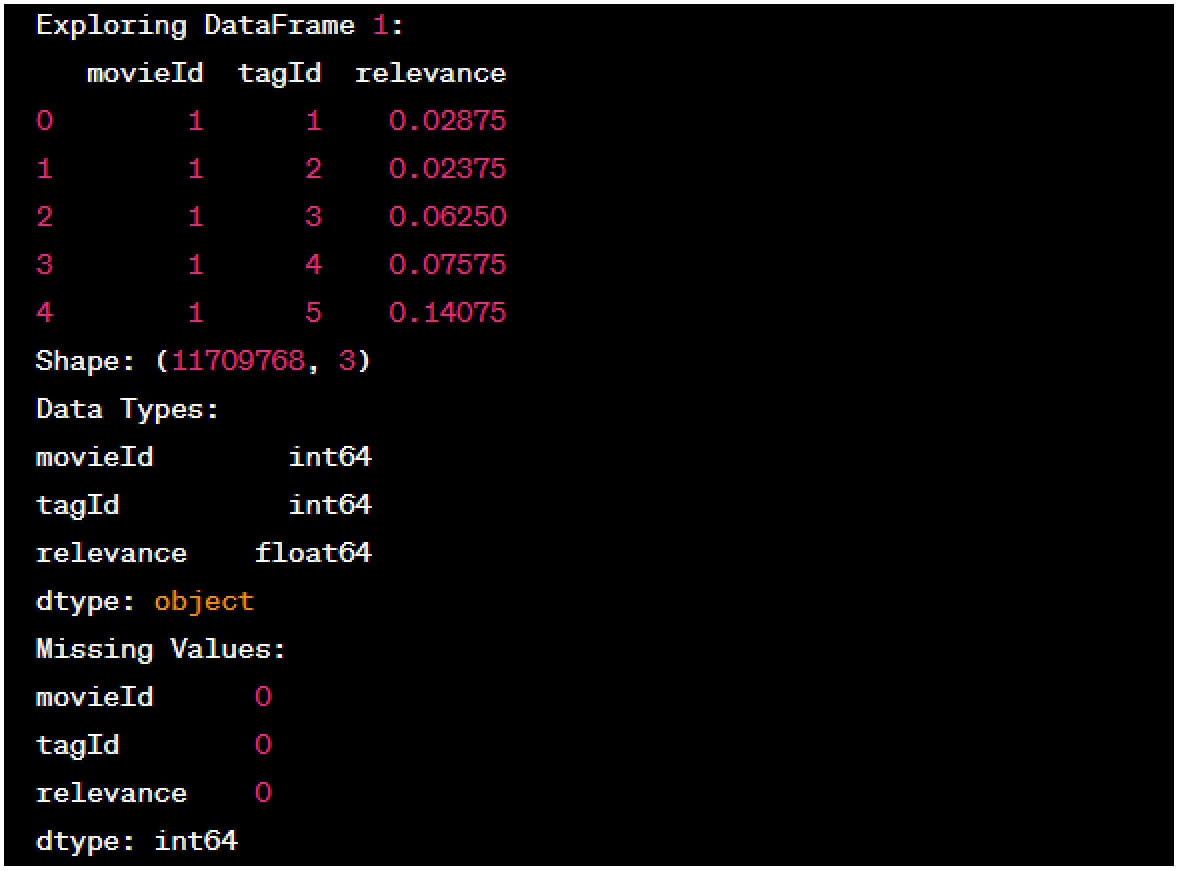
Collaborative filtering depends on user-movie interactions for recommendation.

The 'ratings.csv' file is loaded and processed to create a user-item matrix, representing users' ratings for movies.

Any missing values in the matrix are filled with 0 as the default value.

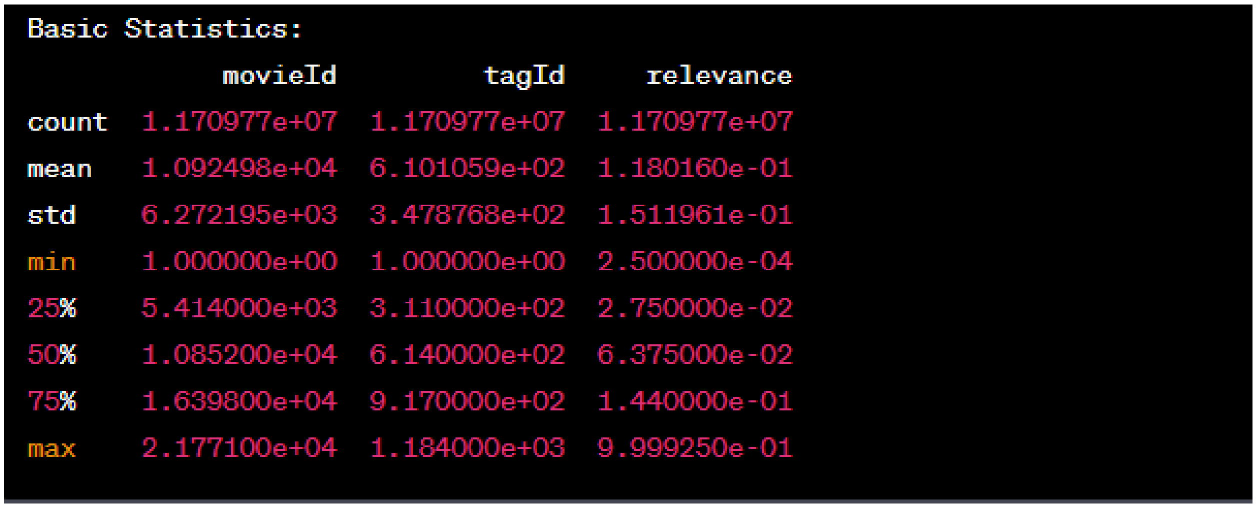
#### 4.4.1.4Hybrid Recommendation Algorithm:

After data preparation, the hybrid recommendation algorithm in movierecommender.py combines content-based and collaborative filtering for personalized movie recommendations.



```
Exploring DataFrame 1:
   movieId  tagId  relevance
0        1      1    0.02875
1        1      2    0.02375
2        1      3    0.06250
3        1      4    0.07575
4        1      5    0.14075
Shape: (11709768, 3)
Data Types:
movieId        int64
tagId          int64
relevance    float64
dtype: object
Missing Values:
movieId      0
tagId        0
relevance    0
dtype: int64
```

*calculations in flask 1*



```
Basic Statistics:
            movieId         tagId     relevance
count  1.170977e+07  1.170977e+07  1.170977e+07
mean   1.092498e+04  6.101059e+02  1.180160e-01
std    6.272195e+03  3.478768e+02  1.511961e-01
min    1.000000e+00  1.000000e+00  2.500000e-04
25%    5.414000e+03  3.110000e+02  2.750000e-02
50%    1.085200e+04  6.140000e+02  6.375000e-02
75%    1.639800e+04  9.170000e+02  1.440000e-01
max    2.177100e+04  1.184000e+03  9.999250e-01
```

*statistics in flask*

This sections documents the Data Preparation process, including the data exploration and loading steps in datapreparation.py. It emphasizes the importance of data preparation in preparing the datasets for the hybrid recommendation algorithm's accurate functioning.

### 4.4.2 Database Design Models

In this section, we describe the database design models used in the movie recommendation application, focusing on storing user information and authentication details. While the specific database management system (DBMS) is not provided in the code snippets, it's essential to ensure that the chosen DBMS supports the required functionalities efficiently.

Users Table:

The 'Users' table stores user information, including unique user IDs, usernames, email addresses, and hashed passwords for secure authentication.

User-specific data, such as user preferences or previous movie ratings, can also be linked to their corresponding user IDs.

#### 4.4.2.1 Data Models and CSV Files

In this section, we explore the data models used for content-based filtering and collaborative filtering, and their representation through CSV files.The data models were from Movielens.

#### 4.4.2.2 Content-Based Filtering Data Model:

The 'movies.csv' file contains movie information, including movie IDs, titles, genres, and other relevant attributes.

Preprocessing in datapreparation.py ensures that the genres are represented in a suitable format for content-based filtering.

**Collaborative Filtering Data Model:**

The 'ratings.csv' file holds user-movie interaction data, including user IDs, movie IDs, and corresponding movie ratings.

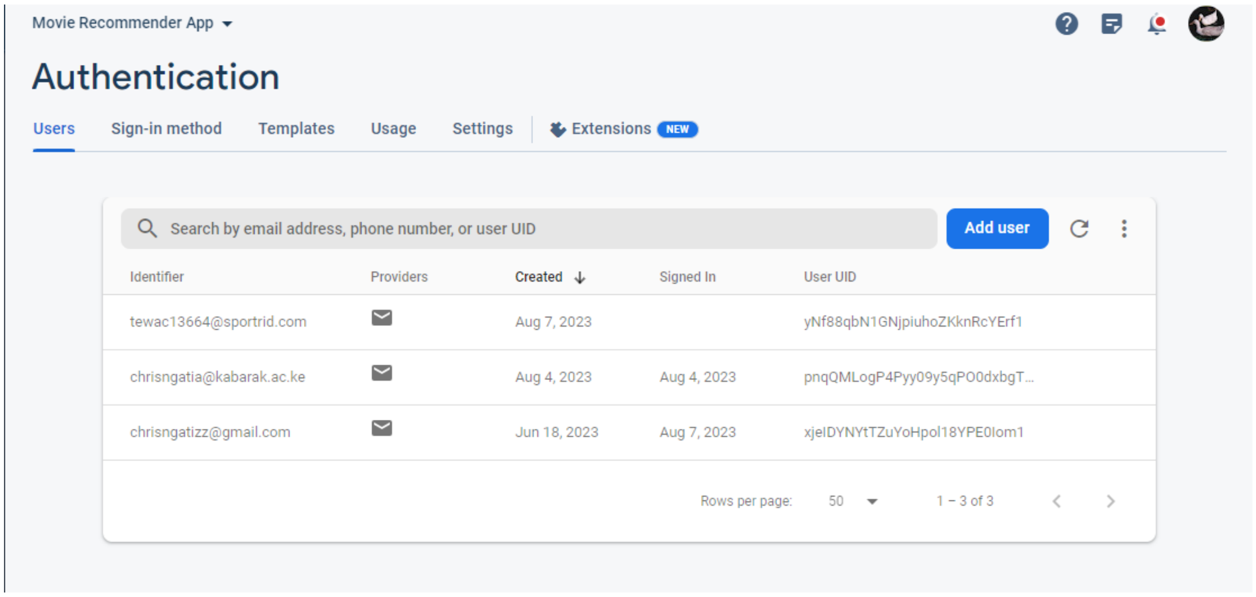
The user_item_matrix in movierecommender.py represents users' ratings for movies as a matrix.

**Integration with Firebase for User Authentication**

While the specific Firebase integration code is not provided in the code snippets, it's essential to ensure the secure management of user accounts and authentication.

**User Registration and Authentication:**

Firebase provides robust user registration and authentication mechanisms, ensuring secure access to personalized movie recommendations.



| Identifier | Providers | Created | Signed In | User UID |
|---|---|---|---|---|
| tewac13664@sportrid.com | ✉ | Aug 7, 2023 | | yNf88qbN1GNjpiuhoZKknRcYErf1 |
| chrisngatia@kabarak.ac.ke | ✉ | Aug 4, 2023 | Aug 4, 2023 | pnqQMLogP4Pyy09y5qPO0dxbgT... |
| chrisngatizz@gmail.com | ✉ | Jun 18, 2023 | Aug 7, 2023 | xjeIDYNYtTZuYoHpol18YPE0lom1 |

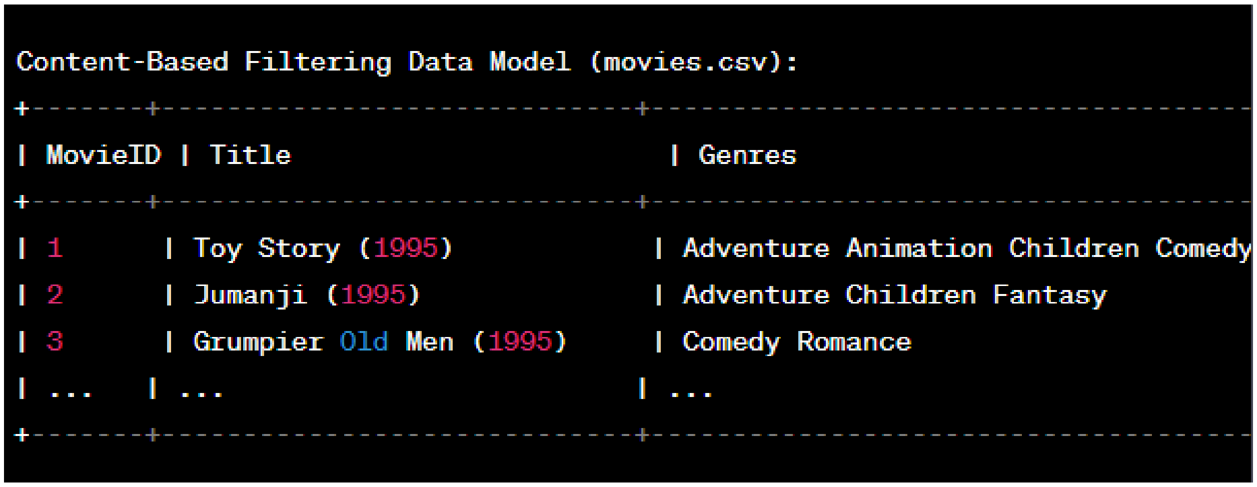
Rows per page: 50 1 – 3 of 3

*Firebase Authentication 1*

User Authentication Token:

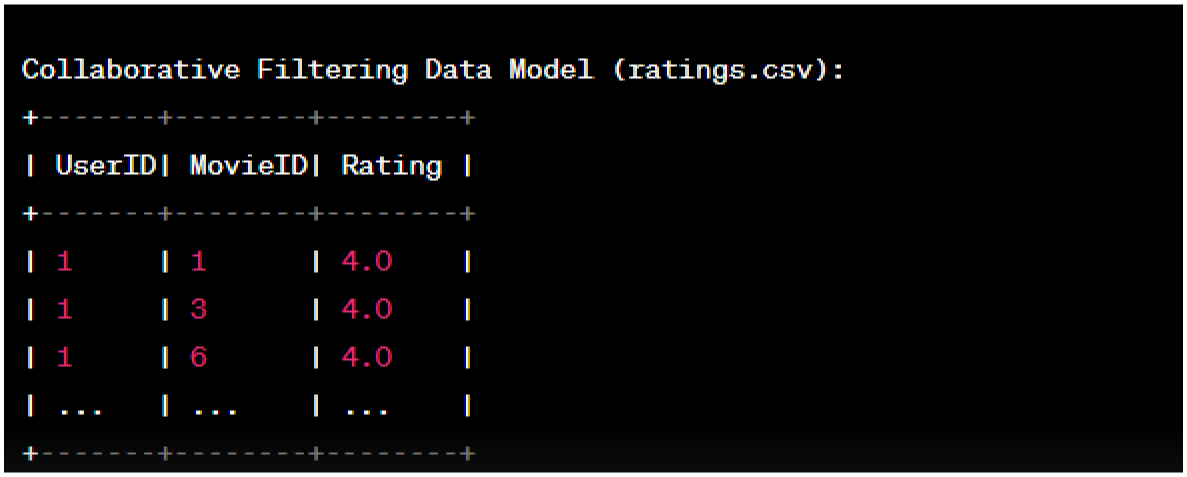
When users sign in or register, Firebase issues an authentication token, which can be used to verify user identity when making requests to the Flask API.

Example Output - Database Design Models:



```
Content-Based Filtering Data Model (movies.csv):
+-------+----------------------------+-----------------------------------
| MovieID | Title                      | Genres
+-------+----------------------------+-----------------------------------
| 1      | Toy Story (1995)           | Adventure Animation Children Comedy
| 2      | Jumanji (1995)             | Adventure Children Fantasy
| 3      | Grumpier Old Men (1995)    | Comedy Romance
| ...    | ...                        | ...
+-------+----------------------------+-----------------------------------
```

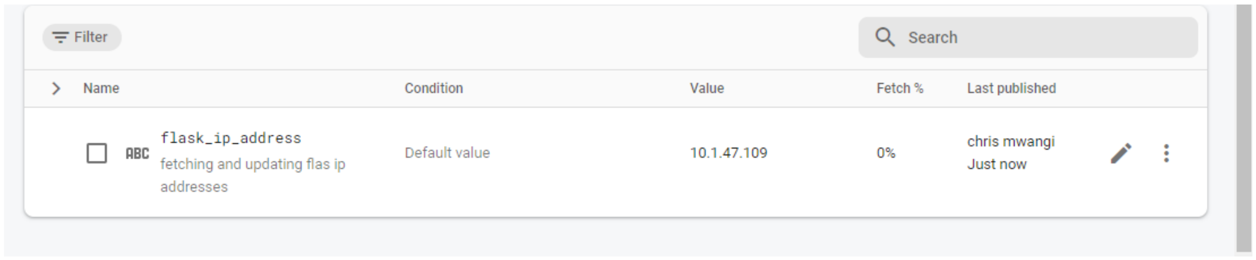
content model from movielens



```
Collaborative Filtering Data Model (ratings.csv):
+-------+--------+--------+
| UserID| MovieID| Rating |
+-------+--------+--------+
| 1     | 1      | 4.0    |
| 1     | 3      | 4.0    |
| 1     | 6      | 4.0    |
| ...   | ...    | ...    |
+-------+--------+--------+
```

collaborative model

Additionally we have intergrated the firebase remote configuration function in the application so that the user does not need to keep on doing the full process of configuring again the ip address for the flask api call in the code leading one to code,test,analyse and deploy again.Insteadit can be done on the firebase whereby when you switch to a different network and the ip address changes,all you need to do is update it in firebase and the flask server will then communicate to the application hence no need to do update or anything to the application.



| Name | Condition | Value | Fetch % | Last published |
|---|---|---|---|---|
| flask_ip_address fetching and updating flas ip addresses | Default value | 10.1.47.109 | 0% | chris mwangi Just now |

*remote configuration in firestore 1*

This section documents the Backend Development related to Database Design Models and Data Models used in content-based and collaborative filtering. Additionally, it highlights the integration with Firebase for user authentication.

## 4.5 System Testing

During the development process of the movie recommendation application, system testing was conducted to ensure that the application's functionalities were working correctly and met the specified requirements. This involved testing both individual components and the system as a whole.

Functional testing was carried out to verify that each component of the application worked as intended. This included testing the content-based and collaborative filtering algorithms, as well as the '/recommendations' API endpoint.

Unit testing was also performed to test individual units or functions of the application in isolation. This allowed for early detection of bugs and issues and ensured that each unit performed as expected.

Integration testing was conducted to evaluate how different components of the application interacted and functioned together.

Performance testing was carried out to assess the application’s response time, scalability, and resource usage under different load conditions. This helped to ensure that the application was efficient and could handle varying levels of demand.

Overall, system testing played a crucial role in ensuring that the movie recommendation application was functioning correctly and met the specified requirements.

# CHAPTER 5. CONCLUSION AND FUTURE WORK

In this final section, I conclude our journey through the development of the hybrid movie recommendation application. I have successfully explored and realized the intricacies of front-end development, user interface design, and the integration of interface modules, resulting in an engaging and intuitive platform for movie enthusiasts. Our application seamlessly integrates content-based and collaborative filtering techniques to provide personalized movie recommendations, enhancing the user experience.

## 5.1 Conclusion

The realization of the application was a fulfilling endeavor, where I focused on creating an interface that not only facilitated efficient movie discovery but also delivered tailored recommendations. By implementing content-based filtering, I harnessed the power of natural language processing to analyze genres and generate relevant movie suggestions. Collaborative filtering further enhanced our recommendations by considering user interactions and preferences, resulting in a holistic approach that catered to diverse tastes.

My user interface design emphasized user-friendly layouts, interactive components, and visually appealing displays. The homepage's search bar and real-time suggestions made movie searches effortless, while the detailed movie display provided users with comprehensive information to make informed decisions. Personalized recommendations, a key highlight of our design, presented users with a curated list of movies that resonated with their preferences, enhancing their movie-watching experience.

The final version of this application has some areas of improvement that were not Implemented because of time constraints and the capabilities of the device that I was using ie the size of the dataset to be used,It is true that a bigger dataset is better for doing and giving more accurate recommendations but I have tried to solve the issue by the use of similar media from the Movie Database for the movies not found in the dataset.Also I made the application to be of use for android users only but in future we can open the spectrum for Ios and also the we.As Flutter provides the option of easily creating cross platform applications, the testing in Ios and also the web can also be done.

All in all the application works as expected and it provides recommendations to users.I hope that my work is up to the highest of standards.

## 5.2 Future Work

While our application has reached a commendable stage, there are avenues for further enhancement and exploration in the future:

1. Social Integration: Enabling users to share their movie selections and recommendations on social media platforms could enhance user engagement and promote the application's reach.

2. Advanced UI/UX: Continual improvement of the user interface design by incorporating the latest design trends and principles could enhance the application's visual appeal and usability.

3. Integration with Streaming Platforms: Integrating the application with popular streaming platforms through APIs could provide users with a seamless transition from discovering movies to watching them.

4.Advanced Content Analysis: Exploring more advanced content analysis techniques, such as sentiment analysis or plot analysis, could further refine content-based recommendations.

As I conclude this documentation, I recognize that the world of movie recommendations is ever-evolving, and our application lays the foundation for continuous innovation and enhancement. I am excited to see this project evolve and adapt to the changing landscape of technology and user preferences.

I extend our gratitude to everyone involved in the development of this application and look forward to the future possibilities it holds.

With that, I mark the end of our journey through the development of the hybrid movie recommendation application, acknowledging its achievements and anticipating the exciting paths that lie ahead.

## 5.3 REFERENCES

1)Urszula Kużelewska; “Clustering Algorithms in Hybrid Recommender System on MovieLens Data”, Studies in Logic, Grammar and Rhetoric, 2014.

2)Dietmar Jannach, Gerhard Friedrich; “Tutorial: Recommender Systems”, International Joint Conference on Artificial Intelligence, Beijing, August 4, 2013.

3) Gaurangi, Eyrun, Nan; “MovieGEN: A Movie Recommendation System”, UCSB.

4) Harpreet Kaur Virk, Er. Maninder Singh,” Analysis and Design of Hybrid Online Movie Recommender System ”International Journal of Innovations in Engineering and Technology (IJIET)Volume 5 Issue 2,April 2015

5) L. M. de Campos, J. M. Fernández-Luna, J. F. Huete, and M. A. Rueda-Morales, “Combining contentbased and collaborative recommendations: A hybrid approach based on bayesian networks,” International Journal of Approximate Reasoning, vol. 51, no. 7, pp. 785 – 799, 2010.

6) E. A. S. Harpreet Kaur Virk, Er.Maninder Singh, “Analysis and design of hybrid online movie recommender system,” International Journal of Innovations in Engineering and technology, 2015

7) James Bennett, Stan Lanning ; “The Netflix Prize”, In KDD Cup and Workshop in conjunction with KDD,2007

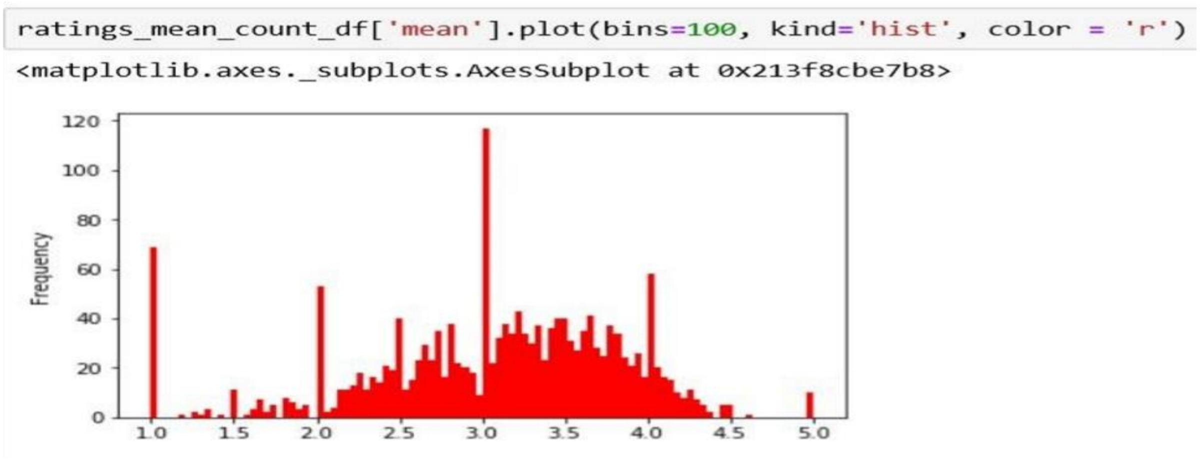
8) Mohammad Yahya H. Al-Shamri , Kamal K. Bharadwaj; “A Compact User Model for Hybrid Movie Recommender System ” in International Conference on Computational Intelligence and Multimedia Applications 2007

9) Costin-Gabriel Chiru, Vladimir-Nicolae Dinu , Ctlina Preda, Matei Macri ; “Movie Recommender System Using the User's Psychological Profile” in IEEE International Conference on ICCP, 2015.

10) Christina Christakou, Leonidas Lefakis, Spyros Vrettos and Andreas Stafylopatis; “A Movie Recommender System Based on Semi-supervised Clustering ”, IEEE Computer Society Washington, DC, USA 2015.

11) Luis M. de Campos, Juan M. Fernández-Luna *, Juan F. Huete, Miguel A. Rueda-Morales; “Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks”

## 5.4 APPENDIX



```
ratings_mean_count_df['mean'].plot(bins=100, kind='hist', color = 'r')
<matplotlib.axes._subplots.AxesSubplot at 0x213f8cbe7b8>
```

EXAMPLE OF RATINGS DATA.

## 5.5 Work plan

| ACTIVITY | ACTIVITY STAFF | MAY | JUNE | JULY | AUG | SEPT | OCT | NOV |
|---|---|---|---|---|---|---|---|---|
| Activity 1 | Project proposal | ■ | | | | | | |
| Activity 2 | Chapter One | ■ | | | | | | |
| Activity 3 | Chapter Two: literature review | | ■ | | | | | |
| Activity 4 | Chapter Three: Methodology And Design | | | ■ | ■ | | | |
| Activity 5 | Chapter Four: System implementation | | | | | ■ | | |
| Activity 6 | Conclusion and recommendation | | | | | ■ | | |
| Activity 7 | References | | | | | | ■ | |
| Activity 8 | Appendix | | | | | | ■ | |
| Activity 9 | Final Documentation | | | | | | | ■ |
| Activity 10 | Presentation | | | | | | | ■ |