

Movie Recommendation System

*Project Report submitted in partial fulfillment for
the award of the degree of*

BACHELOR OF ENGINEERING IN COMPUTER SCIENCE & ENGINEERING

Submitted by

S. No	Enrolment Number	Admission Number	Student Name	Degree / Branch	Sem
1	19021011752	19SCSE1010594	RITIKA	B. TECH CSE	5 th
3	19021011744	19SCSE1010584	RAKHI SINGH	B. TECH CSE	5 th

COMPUTER SCIENCE & ENGINEERING

**Under the Supervision of
Shubham Kumar
Assistant Professor**



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)



SCHOOL OF COMPUTING SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled “**MOVIE RECOMMENDATION SYSTEM**” in partial fulfillment of the requirements for the award of the B.TECH of School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of September, 2021 to December,2021 under the supervision of Shubham Kumar(Assistant Professor), Department of Computer Science and Engineering, Galgotias University, Greater Noida

The matter presented in the project has not been submitted by us for the award of any other degree of this or any other places.-

Ritika(19SCSE1010594)

Rakhi Singh (19SCSE1010584)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Shubham Kumar
Assistant Professor

CERTIFICATE

The Final Project Viva-Voce examination of Ritika(19SCSE1010594) and Rakhi Singh (19SCSE1010584) has been held on 24 December,2021 and their work is recommended for the award of B.TECH

Signature of Examiner(s)



Signature of Supervisor(s)

Signature of Project Coordinator

Signature of Dean

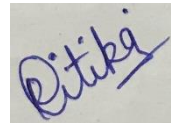
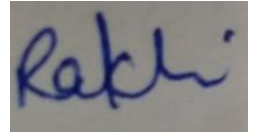
Date: 24 December, 2021

Place: Greater Noida

Statement of Project Report Preparation

Thesis title: **Movie Recommendation System**

Pursuing degree in B.tech computer science and technology with cyber security and networking. Project Supervisor was referred to for preparing the report. Specifications regarding thesis format have been closely followed. The contents of the thesis have been organized based on the guidelines. The report has been prepared without resorting to plagiarism. All sources used have been cited appropriately. The report has not been submitted elsewhere for a degree.

A handwritten signature in blue ink that reads "Ritika".A handwritten signature in blue ink that reads "Rakhi".

Ritika & Rakhi Singh

TABLE OF CONTENTS

ABSTRACT

ii

LIST OF FIGURES

iii

	1. Introduction about Project.	4
	1.1 Objective and Introduction... ..	5
	1.2 Recommender System General Model.	6
	2. Requirements, Feasibility and Scope/Objective	9
2.1	Required Tools.....	9
2.2	Feasibility Analysis.....	10
3.	Analysis, Activity Time Schedule (PERT)	11
	3.1 Problem Formulation	11
	3.2 Activity Time Scheduler	12
4.	DESIGN... ..	17
4.1	Flow Chart	17
4.2	Class Diagram	18
	4.3 DB Class Diagram.....	19
	4.4 Activity Diagram.....	20
	4.5 Context Diagram.....	21
	4.6 Use Case Diagram.....	21
	4.7 Search Diagram.....	22
	4.8 Common Links Diagram.....	22
	5. Implementation and Testing.....	23
	5.1 Project Implementation.....	24
	5.2 Implementing Sentiment Analysis.....	26
	5.3 Predicting the Review.....	27
	5.4 Website Implementation (End Product) and Testing.	30
	5.5 Risk Management... ..	34
	6. Limitations and Future Scope of the Project.....	36
	7. Conclusion	38
	8. References... ..	38

ABSTRACT

Recommender systems are information filtering tools that aspire to predict the rating for users and items, predominantly from big data to recommend their likes. Music recommendation systems provide a mechanism to assist users in classifying users with similar interests. This makes recommender systems essentially a central part of websites and e-commerce applications. This project focuses on the movie recommendation systems whose primary objective is to suggest a recommended movie through a content-based recommendation system. This recommendation system will collect information about the user's preferences of different music in two ways, either implicitly or explicitly. Implicit acquisition of user information typically involves observing the user's behavior, such as watched movies. On the other hand, explicit acquisition involves collecting the user's previous ratings or history.

Such recommendation systems are beneficial for organizations that collect data from large amounts of customers and wish to effectively provide the best suggestions possible. A lot of factors can be considered while designing a movie recommendation system like the genre of the movie, actors present in it, or even the director of the movie. The systems can recommend movies based on one or a combination of two or more attributes. In this project, the recommendation system has been built on the type of genres that the user might prefer to watch. The approach adopted to do so is content-based filtering using genre correlation. The dataset used for the system is collected from the Internet from various sources. The data analysis tool used is Python. Moreover, this project will be covering a full-stack website made on MERN stack, providing recommendations based on user tastes, rich data images, and trailers. The website will be well designed and functional, easy to use, optimized for mobile, fresh quality content. The user's data, like password or data, will be secured with token-based authentication and hashing techniques making this website reliable and secure.

Keywords: Recommender Systems, Collaborative Filtering, Content-based Filtering, Recommendation, Evaluation Metrics.

List of Figures

Figure No.	Title	Page No.
1.	Recommender System General Model	4
2.	Activity Time Schedule	9
3	Flow Chart	14
4	Class diagram	15
5	DB-Class-Diagram and Activity Diagram	16
6	Project Implementation	20
7	Implementing Sentimental Analysis	23
8	Movie Recommendation System (Demo)	25

CHAPTER 1: INTRODUCTION ABOUT PROJECT

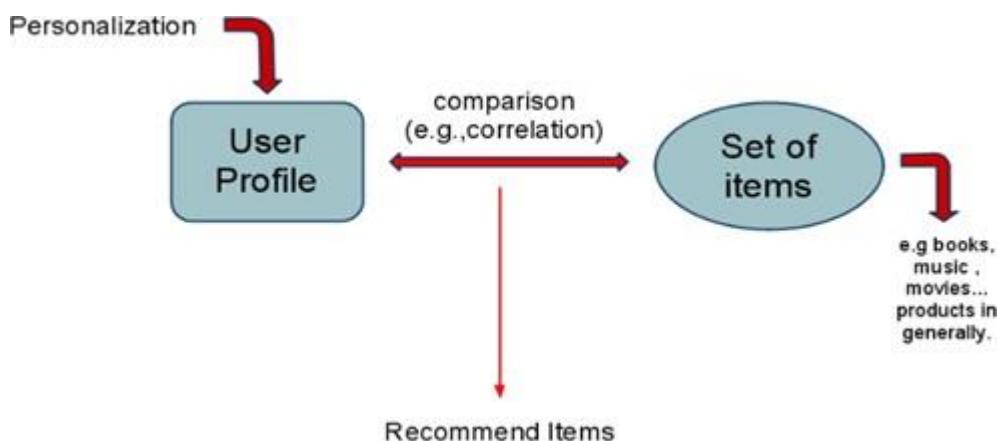
Objective:

The chief purpose of our system is to recommend movies to its users based on their viewing history and ratings that they provide. The system will also recommend various E-commerce companies to publicize their products to specific customers based on the genre of movies they like.

Introduction

Recommendation systems help users find and select items (e.g., books, movies, restaurants) from the huge number available on the web or in other electronic information sources. Given a large set of items and a description of the user's needs, they present to the user a small set of the items that are well suited to the description. Similarly, a movie recommendation system provides a level of comfort and personalization that helps the user interact better with the system and watch movies that cater to his needs. Providing this level of comfort to the user was our primary motivation in opting for movie recommendation system as our BE Project. The chief purpose of our system is to recommend movies to its users based on their viewing history and ratings that they provide. The system will also recommend various E-commerce companies to publicize their products to specific customers based on the genre of movies they like. Personalized recommendation engines help millions of people narrow the universe of potential films to fit their unique tastes. Collaborative filtering and content-based filtering are the prime approaches to provide recommendation to users. Both of them are best applicable in specific scenarios because of their respective ups and downs. In this paper we have proposed a mixed approach such that both the algorithms complement each other thereby improving performance and accuracy of the of our system.

Recommender Systems general model



We may say that recommendation systems resemble search engines, however the user is not searching explicitly for an item but the engine recommends items to the user based on his previous interactions with the system, which are used to model his preferences

Existing System

- It does not work for a new user who has not rated any item yet as enough ratings are required content-based recommender evaluates the user preferences and provides accurate recommendations.
- No recommendation of serendipitous items.
- Limited Content Analysis- The recommender does not work if the system fails to distinguish the items that a user likes from the items that he does not like.

Proposed System

- We have proposed a hybrid recommender system which is known as content-boosted collaborative filtering system.
- This hybrid system takes advantage from both the representation of the content as well as the similarities among users.
- The intuition behind this technique is to use a content-based predictor to fill the user-rating matrix that is sparsely distributed.
- A web crawler is used to download necessary movie content for our dataset. After the preprocessing the movie content database is stored. The dataset consists of a user-rating matrix.

Applications

- What to buy:
 - E-commerce, Books, Movies, Beer, Shoes
- Where to eat?
- Which job to apply to?
- Who you should be friends with?
 - LinkedIn, Facebook.....
- Personalize your experience on the web

News platform, News personalization

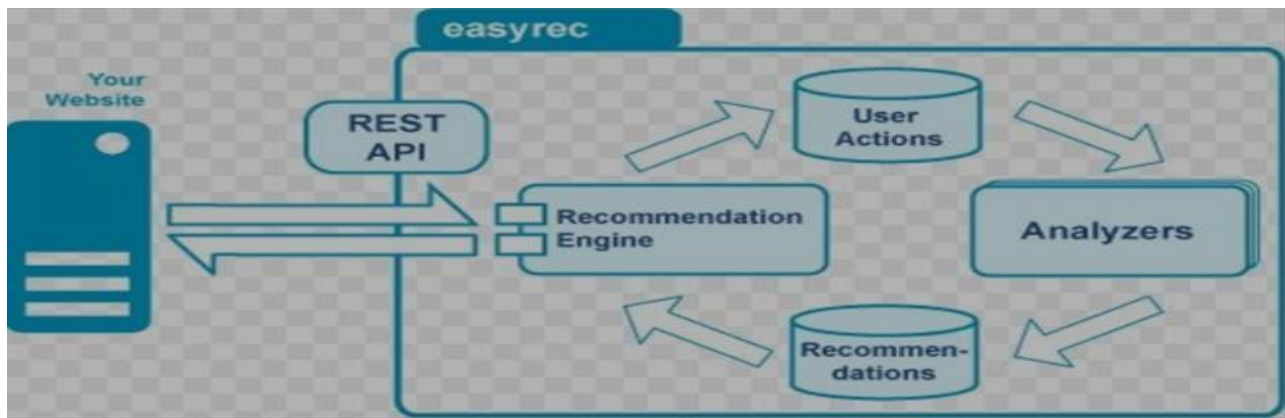
Content based Recommendation System



Recommender systems are active information filtering systems which personalize the information coming to a user based on his interests, relevance of the information etc.

Recommender systems are used widely for recommending movies, articles, restaurants, places to visit, items to buy etc.

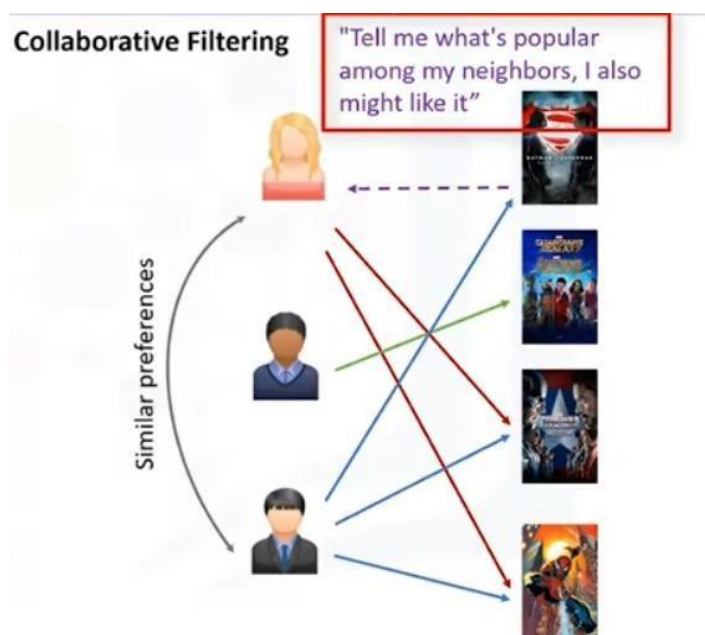
How do Content Based Recommender Systems work?



A content based recommender works with data that the user provides, either explicitly (rating) or implicitly (clicking on a link) .

Based on that data, a user profile is generated, which is then used to make suggestions to the user. As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.

Collaborative Filtering



Collaborative filtering(CF) is a popular recommendation algorithm that bases its predictions and recommendations on the ratings or behavior of other users in the system. The fundamental assumption behind this method is that other users' opinions can be selected and aggregated in such a way as to provide a reasonable prediction of the active user's preference. Intuitively, they assume that, if users agree about the quality or relevance of some items, then they will likely agree about other items.

CHAPTER 2: REQUIREMENTS, FEASIBILITY AND SCOPE/OBJECTIVE

Required Tools:

Hardware Requirements

- Processor : Single Core 1.0 Ghz
- Graphic Card: 64 MB minimum
- Storage : 100mb ~ 1GB
- Ram: 1GB
- Device : Laptop, Phone
- Mobile : Device capable of running a browser

Software Requirements

For Developer

- Python (3 or newer)
- NodeJS
- Jupyter Notebook
- MongoDB
- ReactJS
- ExpressJS
- Flask
- Visual Studio Code

For Users

- For Mobile Users: Android Version > 6.0
- For PC Users : Any browser supporting JavaScript.

Feasibility Analysis:

Financial: The proposed project is totally financial independent there is no financial requirement.

Technology: Movie recommendation systems available in the market are dependent on the dataset to contain large clusters of similar users and items. They also do not provide services such as effective remote access via cloud, customer interaction modules, etc. to be solved with the proposed system.

Operational Feasibility: The project will be implemented in a way that it will allow the functioning of recommendations smoothly. It will provide a user-friendly user interface in a modular fashion.

Product/Service Marketplace

The Movie recommendation system will impact client institutions in several ways. The following provides a high-level explanation of how the organization, tools, processes, and roles and responsibilities will be affected as a result of the movie recommendation system implementation:-

Tools: The existing requirement for on site management systems will be eliminated completely with the availability of a cloud-based system.

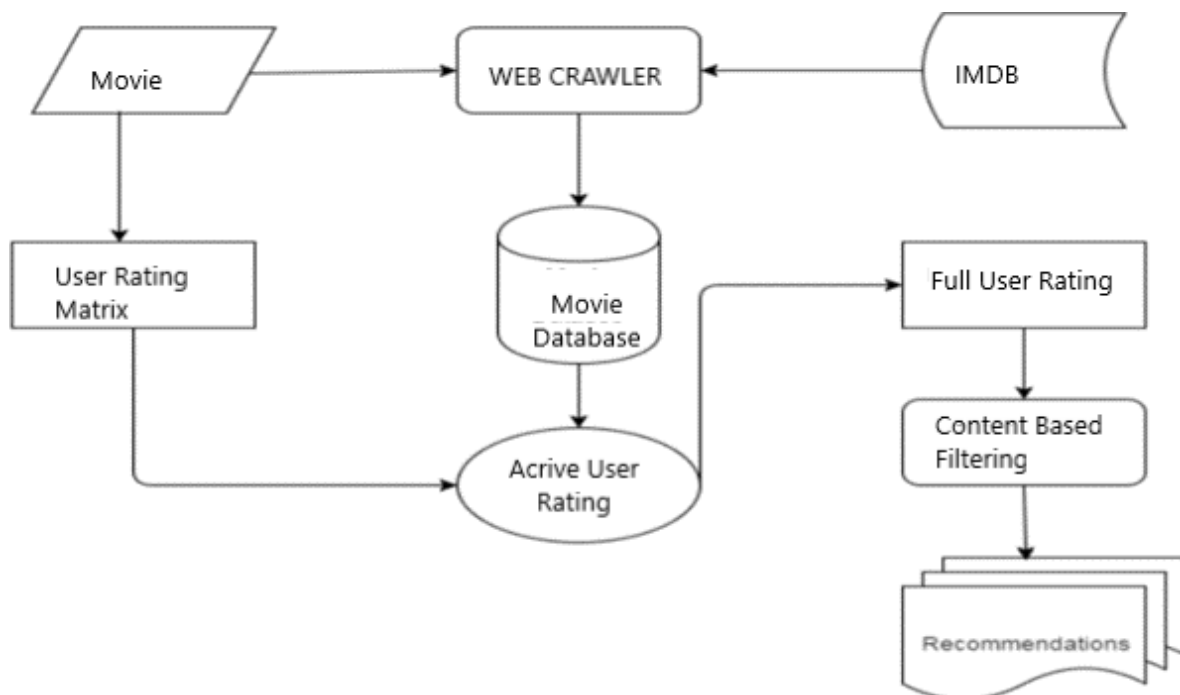
Processes: With the Movie recommendation system comes more efficient and streamlined administrative and customer relations processes.

Hardware/Software: Clients will need to handle no extra software or hardware apart from a stable high-speed Internet connection and a computer device.

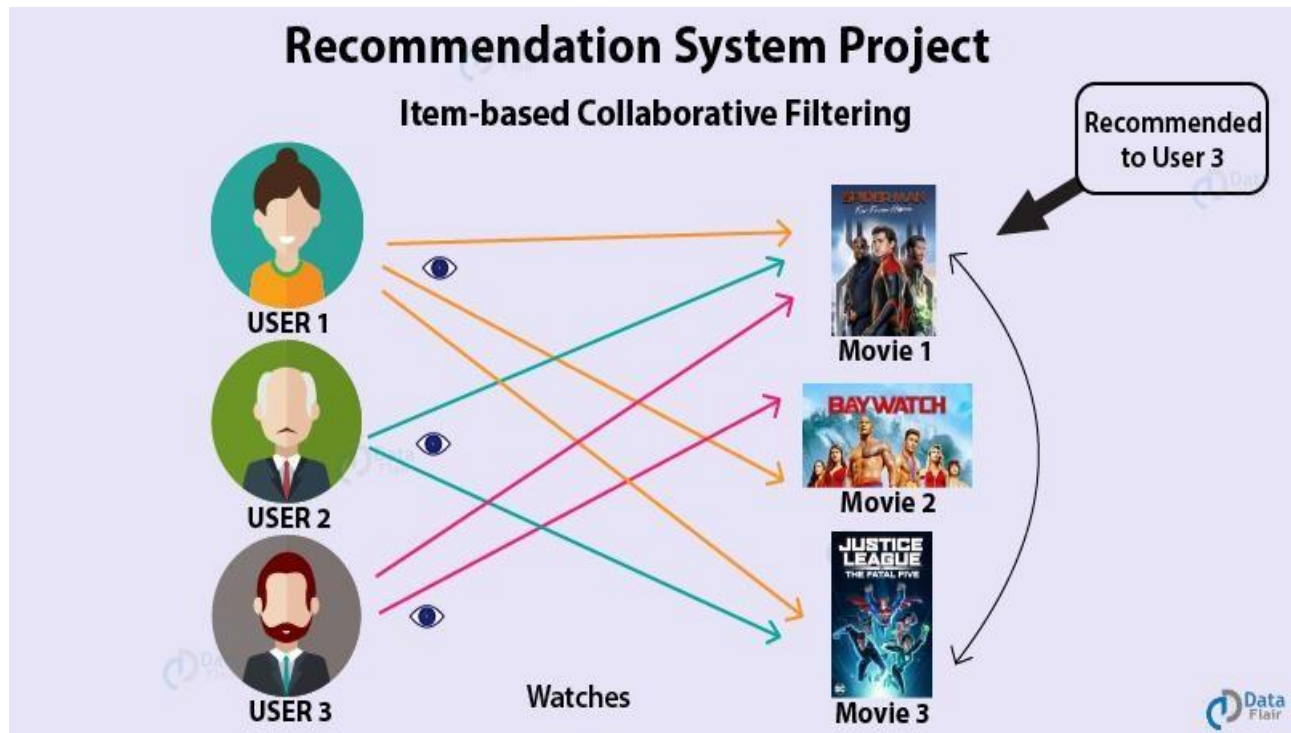
CHAPTER 3: ANALYSIS, ACTIVITY TIME SCHEDULE (PERT)

Problem Formulation

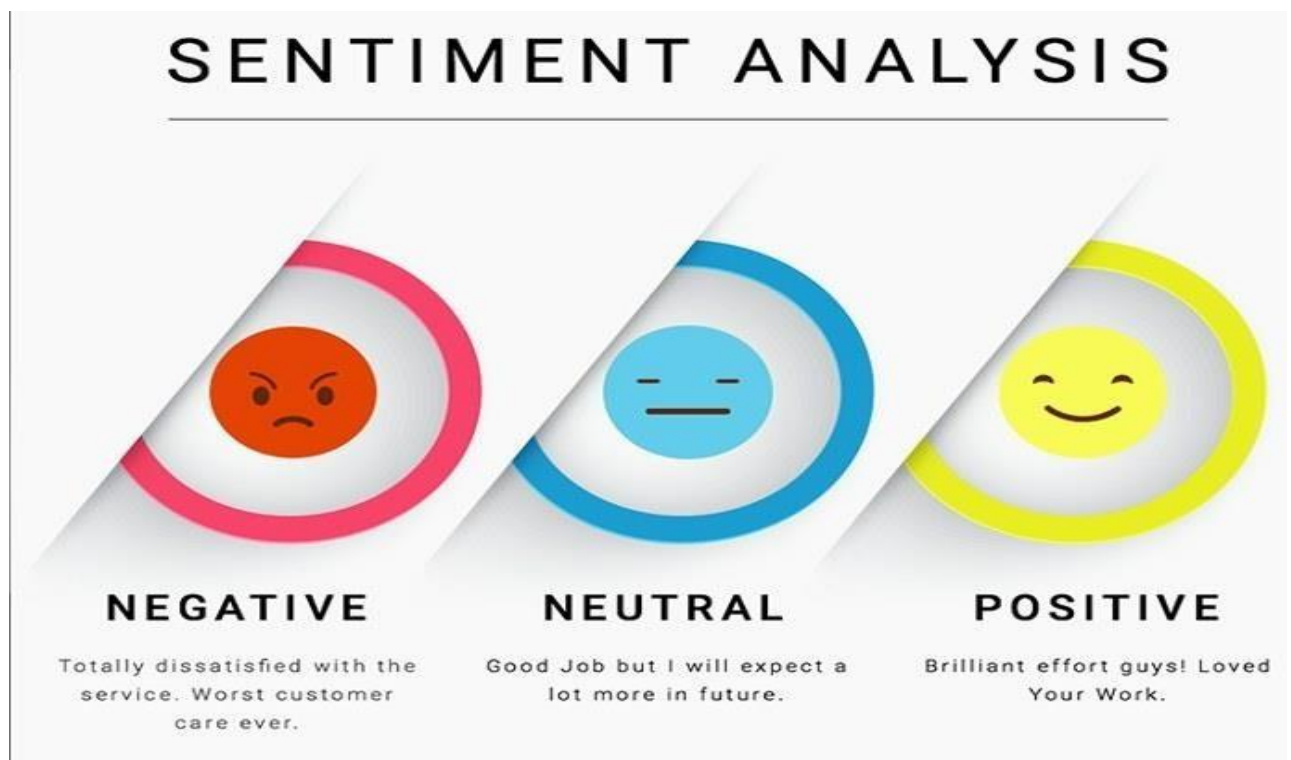
Owing to the various demerits of pure content-based and pure CF based systems, we have proposed a hybrid recommender system which is known as content-boosted collaborative filtering system. This hybrid system takes advantage from both the representation of the content as well as the similarities among users. The intuition behind this technique is to use a content-based predictor to fill the user-rating matrix that is sparsely distributed. A web crawler is used to download necessary movie content for our dataset. After the preprocessing the movie content database is stored. The dataset consists of a user-rating matrix. Content-based predictions are used to train each user-rating vector in the user-rating matrix and convert it into a pseudo rating matrix which combines actual rating with the predicted ratings. Collaborative filtering is then applied to this full pseudo user-rating matrix to make recommendation for an active user.



Week 3: Making content-based recommendation system.



Week 4: Making sentiment analysis model using forward propagation



Week 5: Making database and initializing backend server.

Database

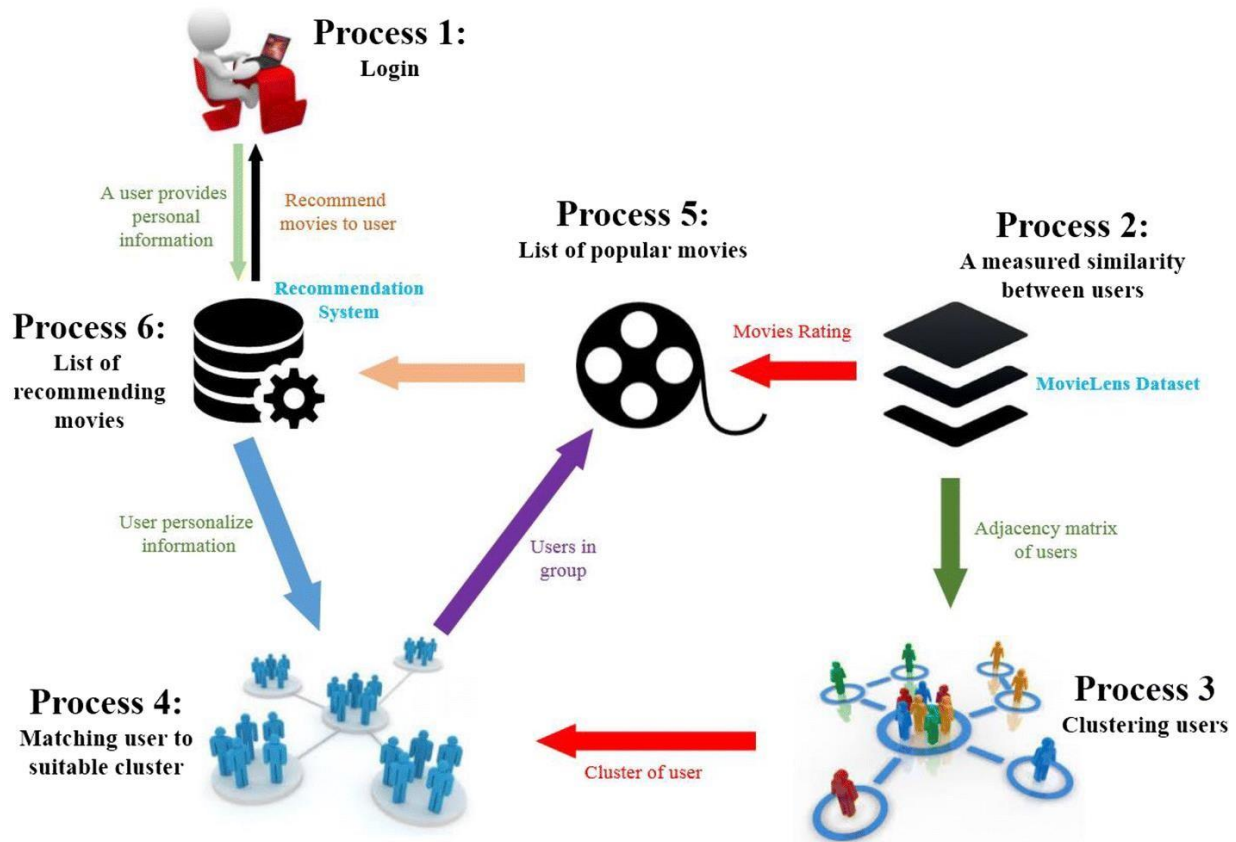
```
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongod
Implicit session: session { "id" : UUID("32502ab2-cb72-4619-89c5-04fb55ea54e1") }
MongoDB server version: 4.0.8
Server has startup warnings:
2020-11-07T12:05:46.570+0530 I CONTROL [initandlisten]
2020-11-07T12:05:46.571+0530 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-11-07T12:05:46.571+0530 I CONTROL [initandlisten] **          Read and write access to data and configuration is u
nrestricted.
2020-11-07T12:05:46.571+0530 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

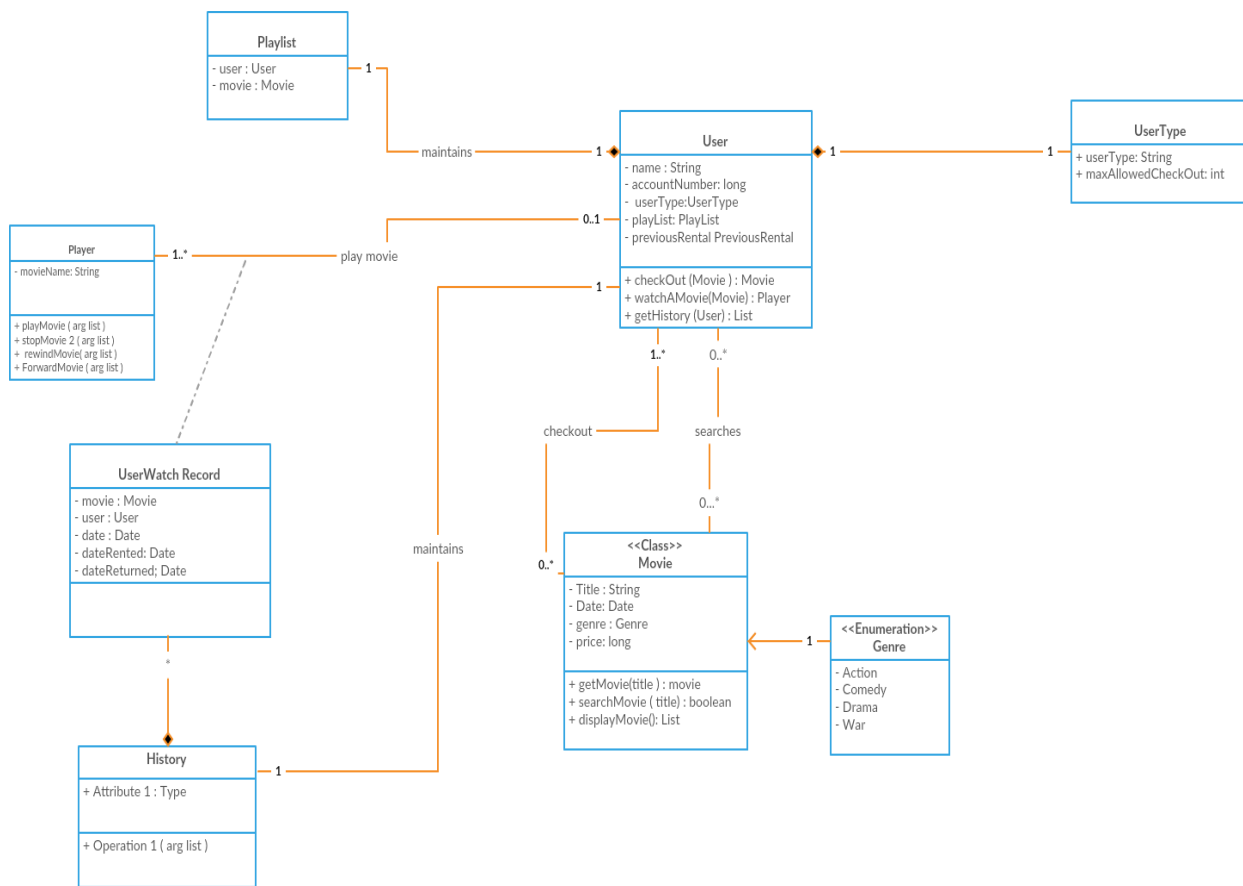
To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> use flick
switched to db flick
> show collections
genres_datas
movies_datas
reviews
user_recommendation_datas
users
>
```

CHAPTER 4: DESIGN

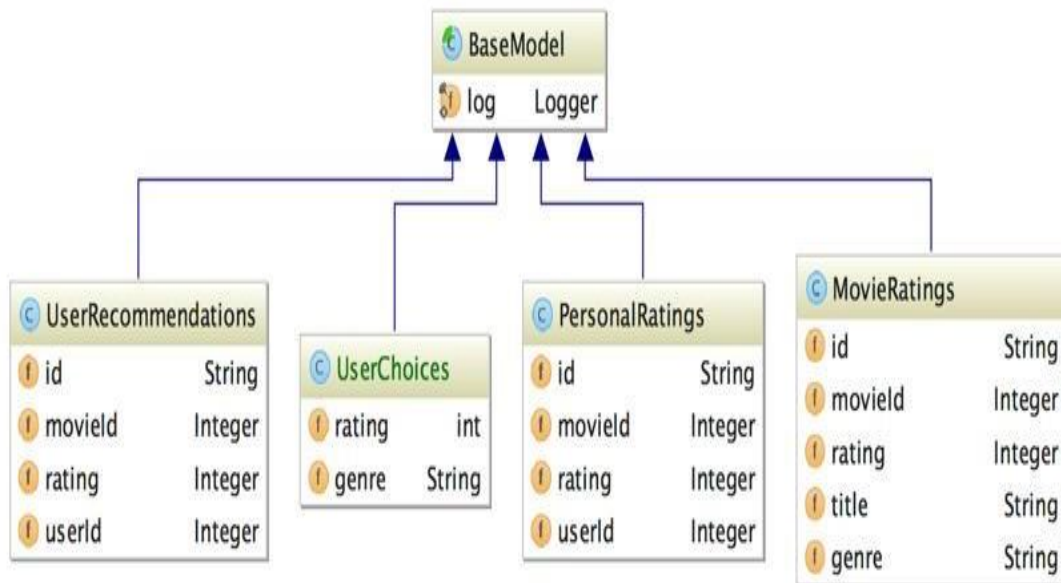
Flow Chart:



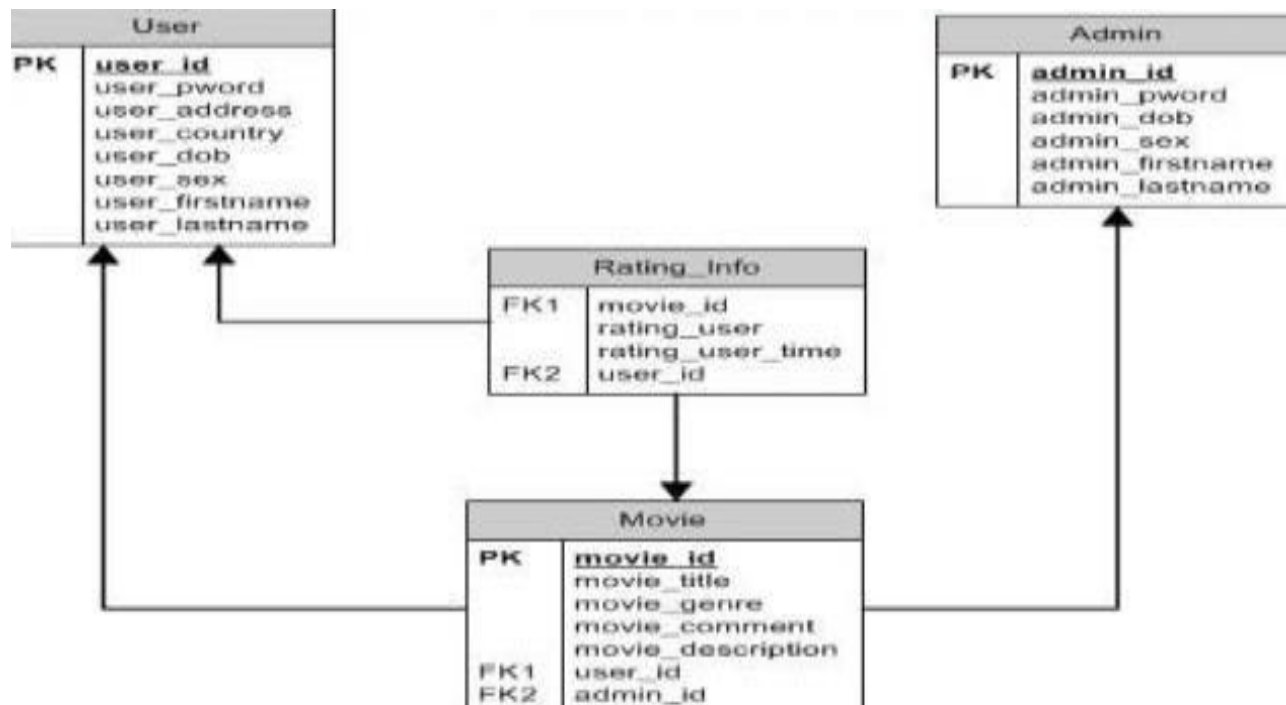
Class Diagram:



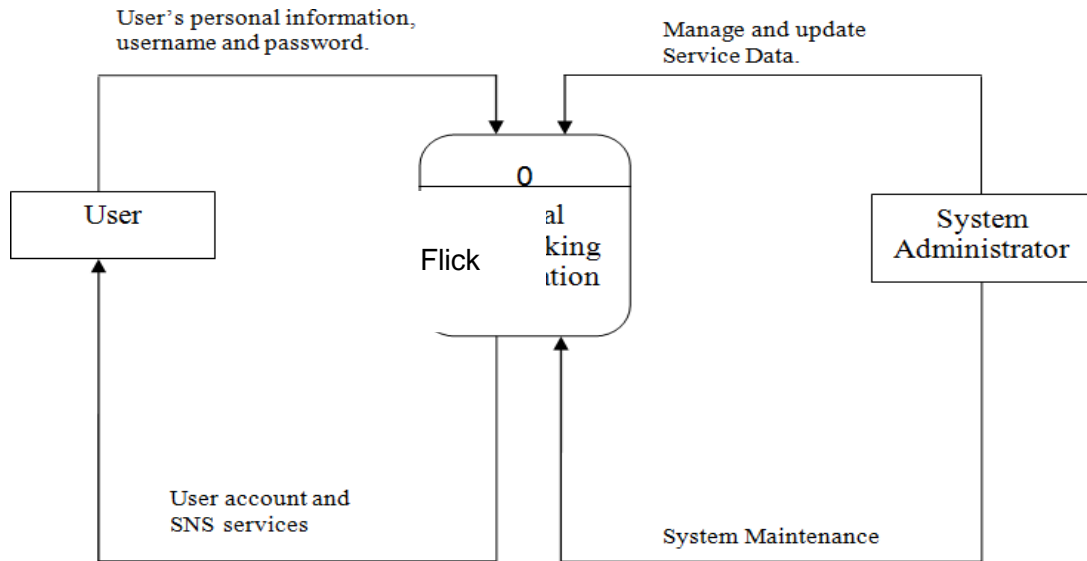
DB-Class-Diagram:



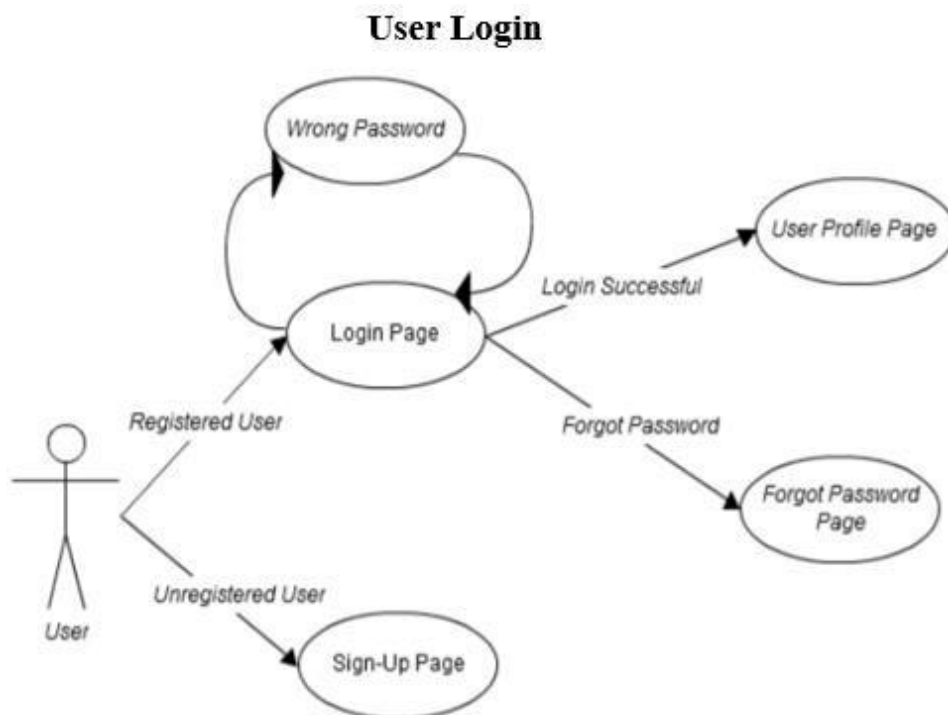
Activity Diagram:



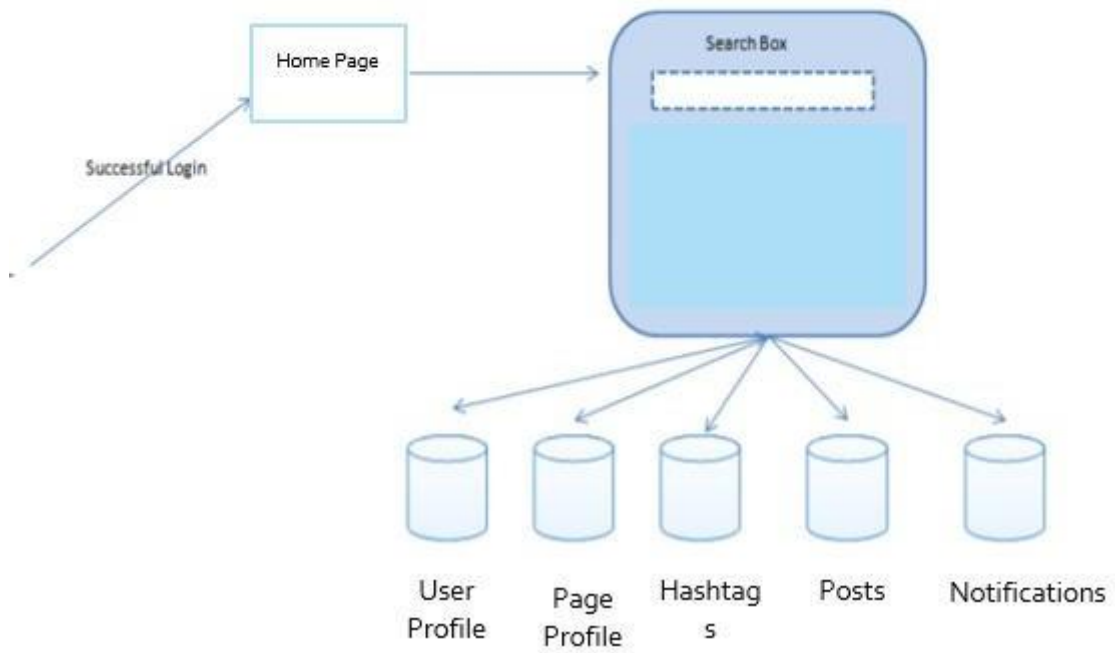
Context Diagram:



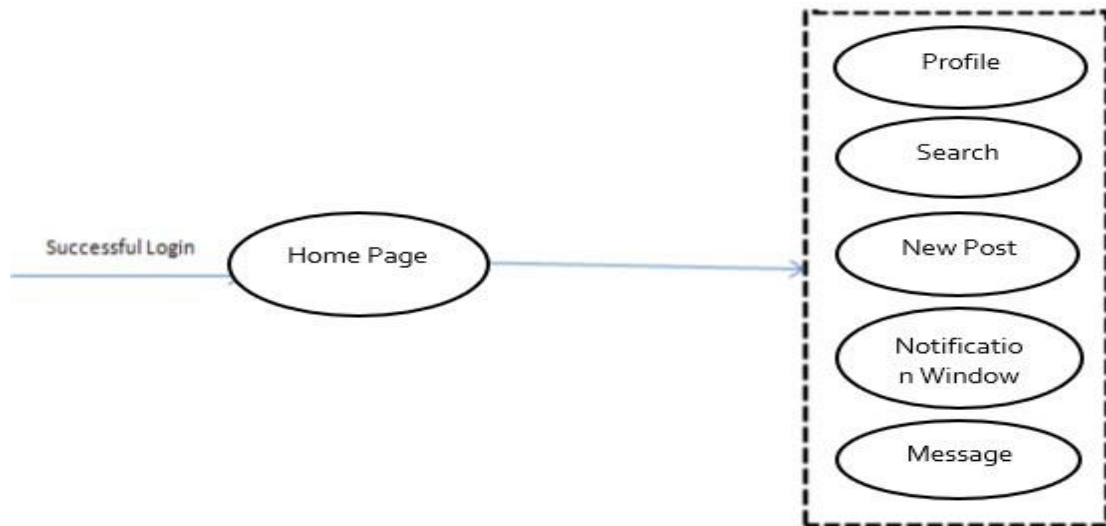
Use Case Diagram:



Search



Common Links

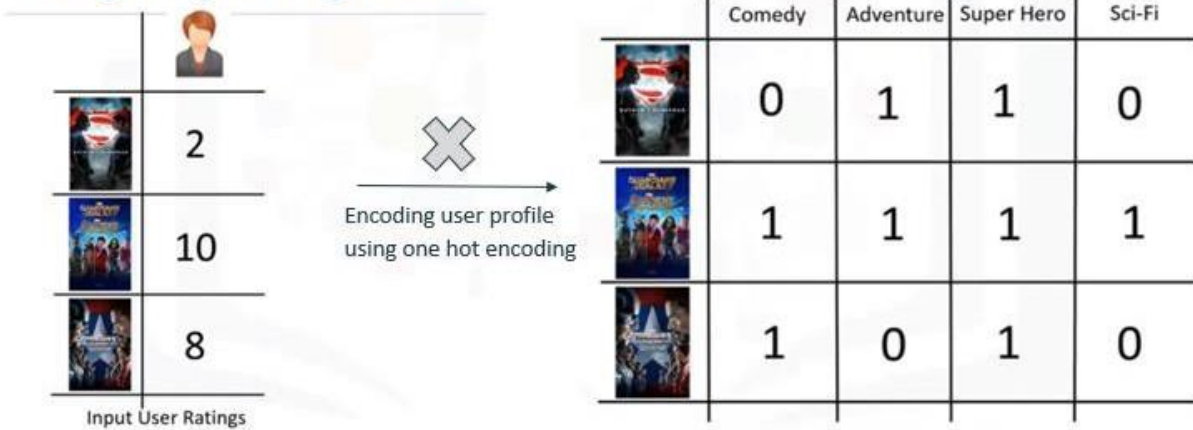


CHAPTER 5. IMPLEMENTATION AND TESTING

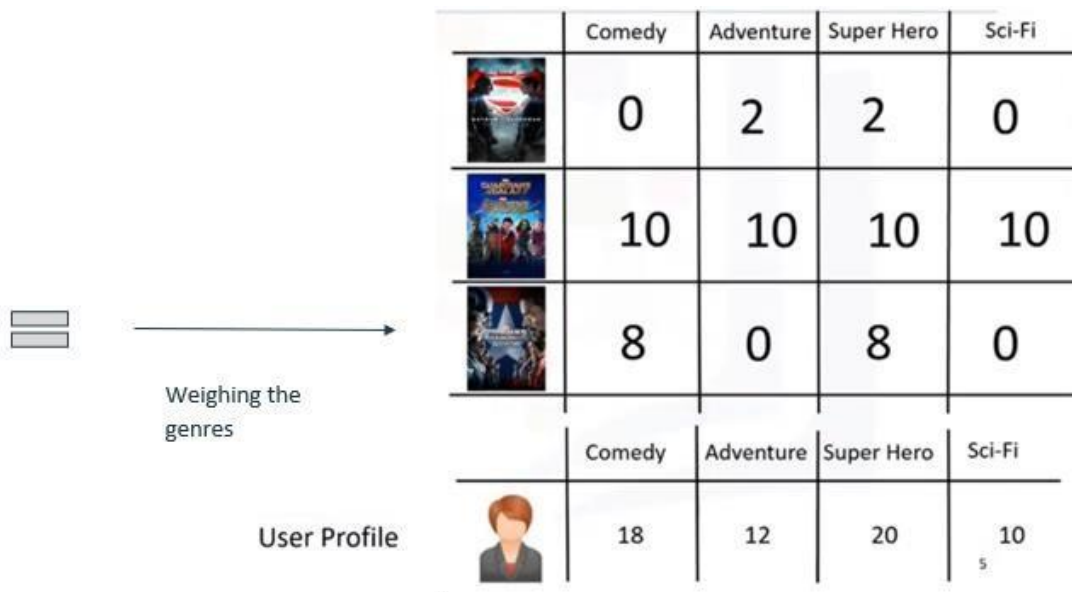
PROJECT IMPLEMENTATION

Step 1 : Weighing the Genres according to rating provided by the user.

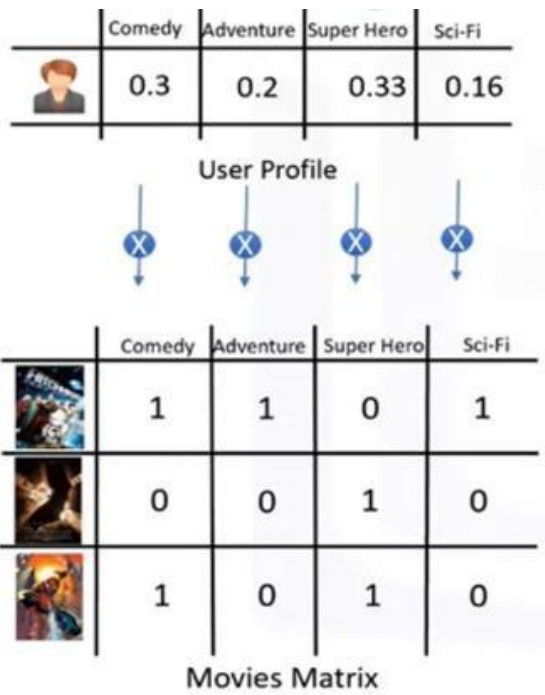
Weighing the genres



Step 2 : User profile created by using one-hot encoding.



Step 3 : Encoding User-Profile matrix with Movies Matrix.

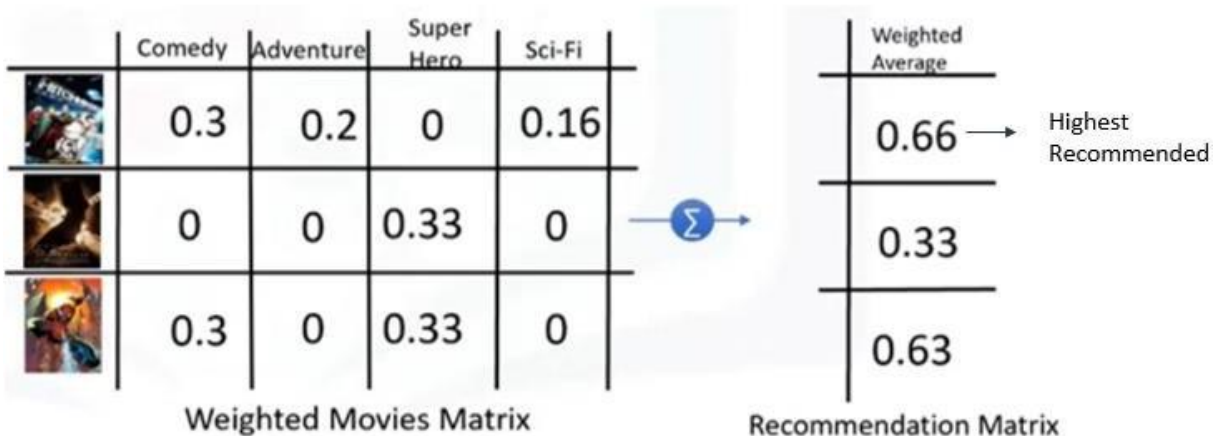


Finding the recommendation



Step 4 : Creating recommendation Matrix.

Finding the recommendations



Implementing Sentiment Analysis:

Sentiment analysis (also known as opinion mining) is a natural language processing problem where text is understood and the underlying intent is predicted. Sentiment analysis is often used to find out the sentiment in customers' feedback.

IMDB dataset contains 25,000 highly polar moving reviews (good or bad) for training and testing. The problem is to determine whether a given moving review has a positive or negative sentiment.

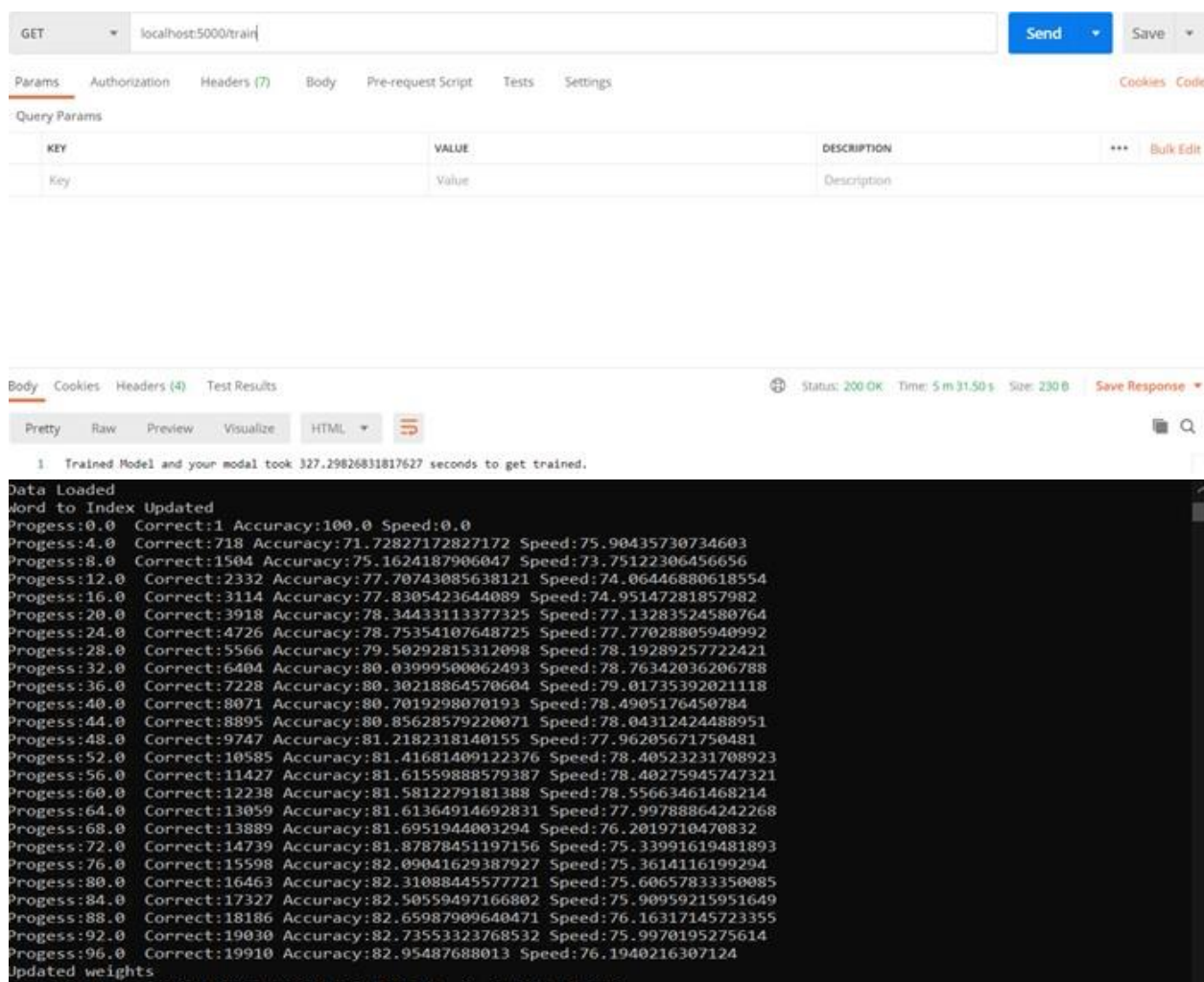
Summing up, sentiment analysis tools serve to extract insights which are crucial to getting what users think and react on time to improve their user experience.

The model is trained on the dataset of movie reviews using neural networks having accuracy of 85%.

Future Work: To improve accuracy of the model using reinforcement learning.

Demo:

Training the model



The screenshot shows a REST client interface with a GET request to localhost:5000/train. The response status is 200 OK. The response body contains a progress log for training a model, showing accuracy increasing from 0% to 82% over 96 steps.

```
1 Trained Model and your modal took 327.29826831817627 seconds to get trained.
Data Loaded
word to Index Updated
Progress:0.0 Correct:1 Accuracy:100.0 Speed:0.0
Progress:4.0 Correct:718 Accuracy:71.72827172827172 Speed:75.90435730734603
Progress:8.0 Correct:1504 Accuracy:75.1624187906047 Speed:73.75122306456656
Progress:12.0 Correct:2332 Accuracy:77.70743085638121 Speed:74.06446880618554
Progress:16.0 Correct:3114 Accuracy:77.8305423644089 Speed:74.95147281857982
Progress:20.0 Correct:3918 Accuracy:78.34433113377325 Speed:77.13283524580764
Progress:24.0 Correct:4726 Accuracy:78.75354107648725 Speed:77.77028805940992
Progress:28.0 Correct:5566 Accuracy:79.50292815312098 Speed:78.19289257722421
Progress:32.0 Correct:6404 Accuracy:80.03999500062493 Speed:78.76342036206788
Progress:36.0 Correct:7228 Accuracy:80.30218864570604 Speed:79.01735392021118
Progress:40.0 Correct:8071 Accuracy:80.7019298070193 Speed:78.4905176450784
Progress:44.0 Correct:8895 Accuracy:80.85628579220071 Speed:78.04312424488951
Progress:48.0 Correct:9747 Accuracy:81.2182318140155 Speed:77.96205671750481
Progress:52.0 Correct:10585 Accuracy:81.41681409122376 Speed:78.40523231708923
Progress:56.0 Correct:11427 Accuracy:81.61559888579387 Speed:78.40275945747321
Progress:60.0 Correct:12238 Accuracy:81.5812279181388 Speed:78.55663461468214
Progress:64.0 Correct:13059 Accuracy:81.61364914692831 Speed:77.99788864242268
Progress:68.0 Correct:13889 Accuracy:81.6951944003294 Speed:76.2019710470832
Progress:72.0 Correct:14739 Accuracy:81.87878451197156 Speed:75.33991619481893
Progress:76.0 Correct:15598 Accuracy:82.09041629387927 Speed:75.3614116199294
Progress:80.0 Correct:16463 Accuracy:82.31088445577721 Speed:75.60657833350085
Progress:84.0 Correct:17327 Accuracy:82.50559497166802 Speed:75.90959215951649
Progress:88.0 Correct:18186 Accuracy:82.65987909640471 Speed:76.16317145723355
Progress:92.0 Correct:19030 Accuracy:82.73553323768532 Speed:75.9970195275614
Progress:96.0 Correct:19910 Accuracy:82.95487688013 Speed:76.1940216307124
Updated weights
```

Predicting the review:

Collaborative filters

Collaborative Filters work with an interaction matrix, also called rating matrix. The aim of this algorithm is to learn a function that can predict if a user will benefit from an item-meaning the user will likely buy, listen to, watch this item.

Among collaborative-based systems, we can encounter two types: **user-item** filtering and **item-item** filtering.

What algorithms do collaborative filters use to recommend new songs? There are several machine learning algorithms that can be used in the case of collaborative filtering. Among them, we can mention nearest-neighbor, clustering, and matrix factorization.

K-Nearest Neighbors (kNN) is considered the standard method when it comes to both user-based and item-based collaborative filtering approaches.

We'll go through the steps for generating a music recommender system using a k-nearest algorithm approach.

Importing required libraries

First, we'll import all the required libraries.

```
In [1]: import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

```
In [2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [3]: from scipy.sparse import csr_matrix
```

```
In [4]: from recommenders.knn_recommender import Recommender
```

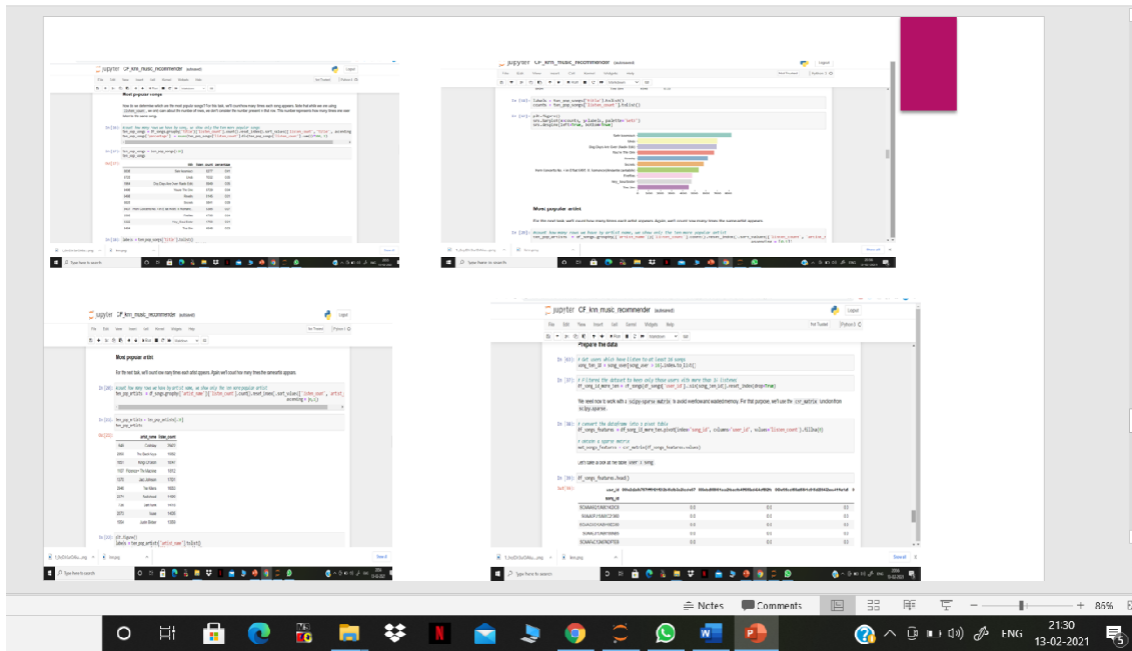
Show all



Slides | Font | Paragraph | Drawing | Editing

The thumbnails show the following content:

- Slide 1:** Introduction to Collaborative Filters, including the definition of the interaction matrix and the goal of the algorithm.
- Slide 2:** Overview of collaborative-based systems (user-item vs item-item filtering) and a list of algorithms used (nearest-neighbor, clustering, matrix factorization).
- Slide 3:** Focus on K-Nearest Neighbors (kNN) as the standard method for collaborative filtering.
- Slide 4:** The 'Importing required libraries' section, showing the code for importing warnings, pandas, numpy, matplotlib, seaborn, and the custom Recommender class.



jupyter CF_knn_music_recommender (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

In [42]: `model = Recommender(metric='cosine', algorithm='brute', k=20, data=mat_songs.features, decode_id_song=decode_id_song)`

In [43]: `song = 'I believe in miracles'`

In [44]: `new_recommendations = model.make_recommendation(new_song=song, n_recommendations=10)`

I believe in miracles
Starting the recommendation process for I believe in miracles ...
... Done

In [62]: `print(f"The recommendations for {song} are:")
print(f"{new_recommendations}")`

The recommendations for I believe in miracles are:
Nine Million Bicycles
If You Were A Sailboat
Shy Bly
I Cried for You
Spider's Web
Piece By Piece
On The Road Again
Blues In The Night
Blue Shoes
Thank You Stars

1_0yUJrUrU...png | knn.png Show all

Source code:-

```
using MovieRecommendationSystem.Infrastructure;
using MovieRecommendationSystem.Models;
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using TinyCsvParser;

namespace MovieRecommendationSystem
{
    class Program
    {
        static void Main(string[] args)
        {
            if (!File.Exists("data/data-full.txt"))
            {
                Console.WriteLine("Could not find file \"data-full.txt\" in the relative directory \"data\". Please make
sure the required data is located in said directory.");
                Console.ReadKey();
                return;
            }

            var csvOptions = new CsvParserOptions(true, ',');
            var csvMovieMapping = new CsvMovieMapping();
            var csvMovieDescriptionMapping = new CsvMovieDescriptionMapping();
            var movieParser = new CsvParser<Movie>(csvOptions, csvMovieMapping);
            var descriptionParser = new CsvParser<MovieDescription>(csvOptions,
csvMovieDescriptionMapping);

            Console.WriteLine("Reading movie data in from \"data/data-full.txt\"...");
            var elapsed = TimeUtilities.MeasureDuration(() => movieParser.ReadFromFile("data/data-full.txt",
Encoding.ASCII).ToList(), out var data);
            var movies = data.Where(x => x.IsValid).Select(x => x.Result).ToList();
            //Console.WriteLine($"Data loaded in {elapsed.TotalSeconds} second(s).");

            List<MovieDescription> descriptions = new List<MovieDescription>();
            if (File.Exists("data/movie_names.txt"))
            {
                Console.WriteLine("Reading optional movie description data in from \"data/movie_names.txt\"...");
                elapsed = TimeUtilities.MeasureDuration(() =>
```

```

descriptionParser.ReadFile("data/movie_names.txt", Encoding.ASCII).ToList(), out var
descriptionData);
    descriptions.AddRange(descriptionData.Where(x => x.IsValid).Select(x => x.Result));
    //Console.WriteLine($"Data loaded in {elapsed.TotalSeconds} second(s).");
}

Console.WriteLine("Training the recommendation system...");
var recommendationSystem = new RecommendationSystem<Movie>(x => x.MovieId, x => x.UserId,
x => x.Rating,
    x => descriptions.FirstOrDefault(y => y.MovieId == x)?.Title ?? x.ToString());
elapsed = TimeUtilities.MeasureDuration(() => recommendationSystem.LoadModel(movies));
//Console.WriteLine($"Recommendation system trained in {elapsed.TotalMinutes} minute(s).");

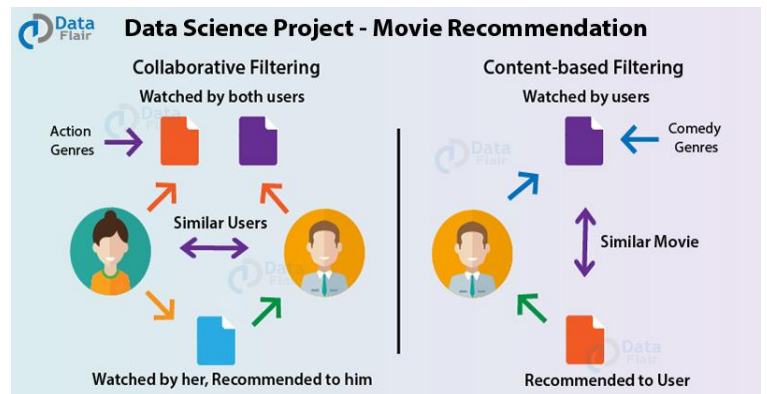
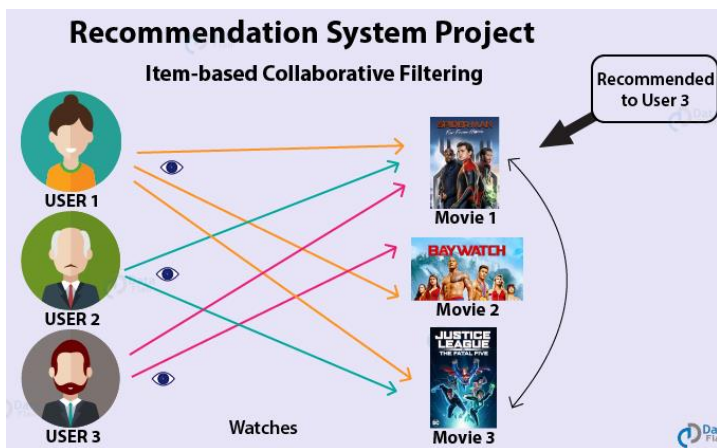
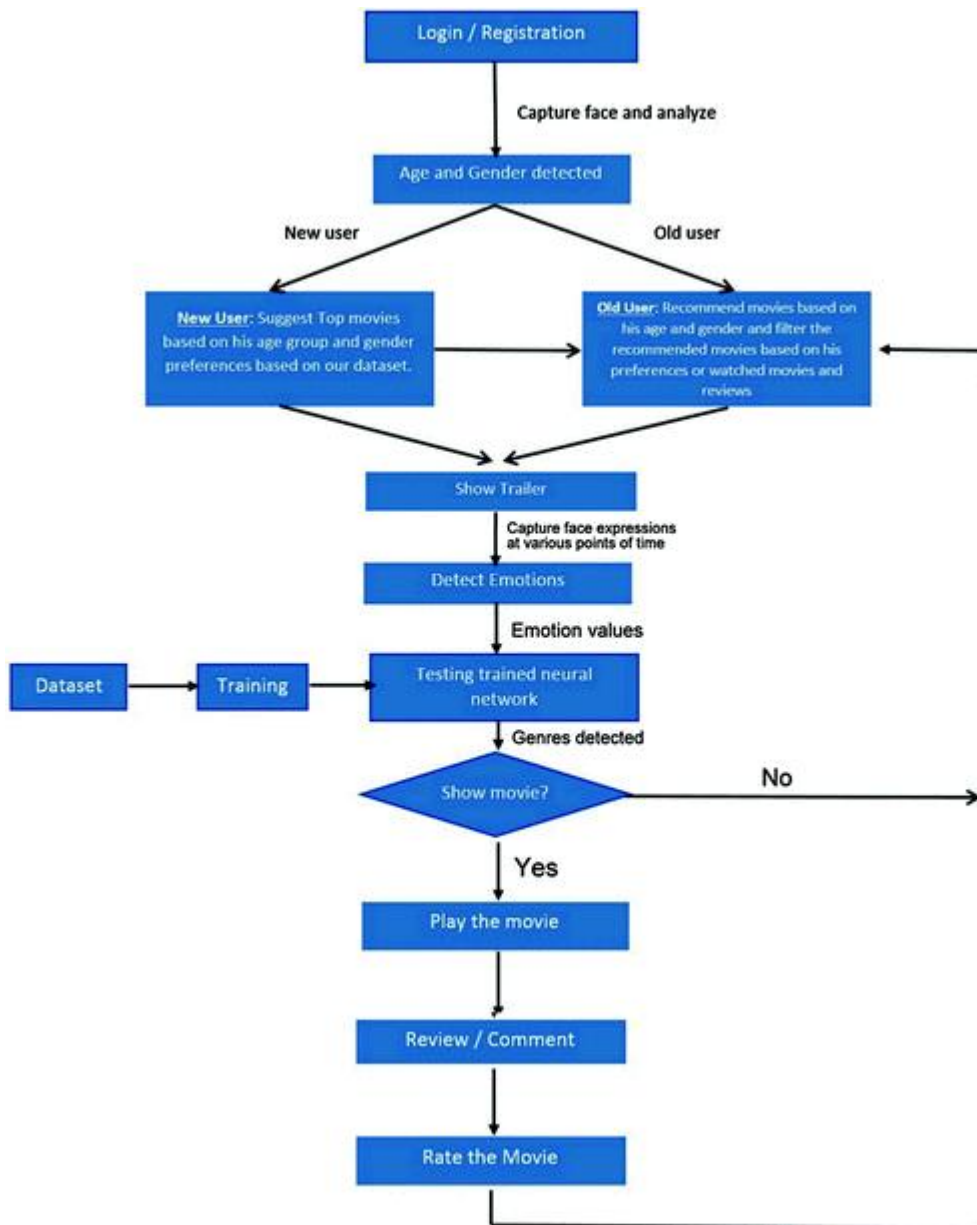
var continueLoop = true;
while (continueLoop)
{
    Console.WriteLine("\nEnter a user ID to predict a rating for:");
    var userParseSuccess = int.TryParse(Console.ReadLine(), out var userId);
    Console.WriteLine("Enter a movie ID to predict a rating for:");
    var movieParseSuccess = int.TryParse(Console.ReadLine(), out var movieId);

    var rating = recommendationSystem.PredictUserRating(userId, movieId);
    Console.WriteLine($"User \"{userId}\" would most likely rate the movie " +
        $"{descriptions.FirstOrDefault(x => x.MovieId == movieId)?.Title ?? movieId.ToString()}\"
{Math.Round(rating, 2)} out of 5. ");

    Console.WriteLine("Enter another? (Y/n)");
    continueLoop = Console.ReadLine().Trim().ToUpperInvariant() == "Y";
}
Console.WriteLine("Press any key to exit...");
Console.ReadKey();
}
}
}

```

Output:



CHAPTER 6: LIMITATIONS AND FUTURE SCOPE OF THE PROJECT

One of the greatest problems of content-based recommender systems, is the vast size of the item set. Since we need to find items in a set that correlate the most with the user's interests, we're obliged to examine all the items. In any other case, we cannot eliminate the possibility that items we haven't examined are not more relevant than the ones we did. Moreover, in the case of content-based recommender systems we must examine the content of every item in order to make a recommendation, whereas in collaborative filtering systems, we only need to examine their ratings by the users. Therefore, the number of items rises very quickly, as is the case in most e-commerce services, the performance of a content-based system decreases. As a result, when we intend to use a recommender system for a web service with a vast number of either old or new items, the solution of a content-based system would not be likely to meet our expectations in terms of performance.

However, the size of the item set as a whole is not the only problem of content-based systems. In addition, every item has its own content that the algorithm must use. Although this usually is not a problem of time or computer resources, since most item representations tend to be small compared with the number of items, how this content is derived from the original item is one of the most important problems we encounter when building a content-based system. In the previous chapter, we've analyzed how items are represented and the different techniques. We also noted that unstructured data is not easy to handle, especially when it comes to multimedia data. However, multimedia data are prevalent in today's Web 2.0, while a vast number of new multimedia items are being added to the Web every day. While we still do not possess techniques that produce satisfying results, the user's need for recommending multimedia items increases every day. Although several attempts have been made to solve this problem and progress has been made, the problem still remains. Therefore, the content analysis needed by content-based systems in order to make recommendations, is an inherent problem that might discourage their use when we deal with multimedia items.

Although many of the content-based advantages derive from the fact that the recommendations for every single user are independent to the user's preferences, that might be the cause for one of its disadvantages. Content-based systems emerged over a decade ago, when the Web was still young and not widely adopted. User communities were more primitive, where the user's profile consisted of a few fields providing information like his name and his age, whereas communications was restricted to text. However, nowadays, with the use of social networks, the Web is more and more used in the context of a community, where user profiles are extensive and constantly updated with information about them, while every user is connected with hundreds of other users and communication between them is achieved using a variety of ways. Although content-based systems allow us to make recommendations to users with unique interests, they fail to group users that have the same interests. Therefore, the lack of a content-based recommendation system to create a group of users that share common interests, might prove to be a great drawback. However, collaborative filtering systems cannot create user groups either, because they do not know the common interests of the users, but only their existence. Therefore, the hybrid model has again proven superior. By combining the item's content with the user stereotypes, it allows us to create clusters of users that share interests, as a result making it possible to create communities of users that share common interests. The result of this feature is not only to improve the user's experience of using our service, but also providing the accuracy of our recommender, since through social interaction the user's provide information that might prove very useful to the recommender system. This realization has created a new trend in recommender systems, where an item is recommended to a group rather than to individuals, thus increasing the possibility that the results are accurate, at least for most of the users of the community.

Recommendation Process Web 2.0 is a term describing the trend in the use of World Wide Web technology that aims at promoting information sharing and collaboration among users. According to Tim O'Reilly⁸, the term "Web 2.0" means putting the user in the center, designing software that critically depends on its users

since the content, as in Flickr, Wikipedia, Del.icio.us, or YouTube, is contributed by thousands or millions of users. That is why Web 2.0 is also called the “participative Web”. O’Reilly 9 also defined Web 2.0 as “the design of systems that get better the more people use them”. One of the forms of User Generated Content (UGC) that has drawn more attention from the research community is folksonomy, a taxonomy generated by users who collaboratively annotate and categorize resources of interests with freely chosen keywords called tags. Despite the considerable number of researches done in the context of recommender systems, the specific problem of integrating tags into standard system algorithms, especially content-based ones, is less explored than the problem of recommending tags (i.e. assisting users for annotation purposes). Folksonomies provide new opportunities and challenges in the field of recommender systems (see Chapter 19). It should be investigated whether they might be a valuable source of information about user interests and whether they could be included in user profiles. Indeed, several difficulties of tagging systems have been identified, such as polysemy and synonymy of tags, or the different expertise and purposes of tagging participants that may result in tags at various levels of abstraction to describe a resource, or the chaotic proliferation of tags.

CONCLUSION:

In this project we have implemented a music recommendation engine / system using simple recommendations, content-based filtering. In addition, a movie recommendation engine has been developed using different method prediction methods. This model is implemented in the python programming language. We have observed that the **RMSE**(Root Mean Square Error) value of the proposed technique is healthier than the current technology after implementing the system with the help of python programming language.

REFERENCES :

- <https://scholar.google.com/scholar?q=Choi%2C%20S.-M.%2C%20Han%2C%20Y.-S.%3A%20A%20content%20recommendation%20system%20based%20on%20category%20correlations.%20In%3A%20The%20Fifth%20International%20Multi-Conference%20on%20Computing%20in%20the%20Global%20Information%20Technology%2C%20pp.%201257%E2%80%931260%20%282010%29>
- <https://scholar.google.com/scholar?q=Popescu%2C%20A.%2C%20Ungar%2C%20L.H.%2C%20Pennock%2C%20D.M.%2C%20Lawrence%2C%20S.%3A%20Probabilistic%20models%20for%20unified%20collaborative%20and%20content-based%20recommendation%20in%20sparse-data%20environments.%20In%3A%20Proceedings%20of%20the%2017th%20Conference%20in%20Uncertainty%20in%20Artificial%20Intelligence%2C%20pp.%20437%E2%80%93444%20%282001%29>
- https://scholar.google.com/scholar_lookup?title=Sparsity%20reduction%20in%20collaborative%20recommendation%3A%20A%20case-based%20approach&author=D.C.%20Wilson&author=B.%20Smyth&author=D.%20O%E2%80%99Sullivan&journal=IJPRAI&volume=17&issue=5&pages=863-884&publication_year=2003