“ MOVIESFLIX “

A PROJECT REPORT

<u>SUBMITTED BY</u>

ABHINAV V P	**:**	**OTAVSCS012**
ADITHYAN E S	**:**	**OTAVSCS015**
MUHAMMED JUMAIL K	**:**	**OTAVSCS021**
SNEHANJALI KRISHNA	**:**	**OTAVSCS025**

FOR THE AWARD OF THE DEGREE OF

BACHELOR OF SCIENCE
(BSc)IN COMPUTER
SCIENCE

(University of Calicut)

DEPARTMENT OF COMPUTER SCIENCE
N.S.S COLLEGE
OTTAPALAM

(Affiliated To University of Calicut)

Palappuram , Kerala – 679103
March 2024

Acknowledgement

First and foremost , we thank the almighty God who gave us the knowledge and strength to successfully complete this project

We take this opportunity to express our gratitude to all people who advice and support us . We wish to express our sincere thanks to **Mr. K RADHAKRISHNAN** , Associate professor and Head Of the Department of Computer Science , for giving his expert guidance through the project work and we express our heartfelt gratitude to **Mr. PRIYESH KG** , Associate professor Department of Computer Science for giving us proper guidance as the project coordinator

Date :

Name : **ABHINAV V P**
ADITHYAN E S
MUHAMMED JUMAIL K
SNEHANJALI KRISHNA

Declaration

We hereby declare that this submission is our own work and that to the best of our knowledge and belief , it contains no material previously published or written by another person or material which has been accepted for the award of any degree or diploma of the university or other institute of higher learning , except where due acknowledgement has been made in this text.

Place : **Ottapalam**

Name : **ABHINAV V P**

ADITHYAN E S

MUHAMMED JUMAIL K

SNEHANJALI KRISHNA

Date :

Certificate

This is to certify that the project report entitled “ **MOVIESFLIX** “ submitted by**MUHAMMED JUMAIL K(OTAVSCS021),** to University Of Calicut for the award of degree of Bachelor of Science (B.S.c) in Computer Science is a bonafide record of the project work carried out by him/her under my supervision and guidance . The content of the report , in full or parts have not been submitted to any other institution or University for the award of any degree or diploma.

Mr.PRIYESH K G	**Mr. Radhakrishnan K**
Faculty Guide	**Head Of Department**

DEPARTMENT OF COMPUTER SCIENCE
N.S.S COLLEGE OTTAPALAM

Place : Ottapalam
Date :

Certified that the candidate was examined by us in the Project Viva Voice Examination held on ………………………… and his/her Register Number is ……………………………………..

Examiners :

1.

2.

TABLE OF CONTENTS

MOVIESFLIX

1. ABSTRACT

MOVIE RECOMMENDATION SYSTEM

This project presents a movie recommendation system employing the k-nearest neighbors (KNN) algorithm to deliver personalized movie suggestions based on user preferences and similarities with other users. Leveraging a dataset containing user ratings and movie features, the KNN algorithm identifies the k most similar users to a target user and recommends movies that these similar users have rated highly. Techniques such as cosine similarity and weighted averages are incorporated to enhance recommendation accuracy. Through experimentation and evaluation, the effectiveness of the approach in providing relevant and personalized movie recommendations is demonstrated. The system has the potential to enhance user satisfaction and engagement on movie streaming platforms by offering tailored content suggestions. Additionally, considerations such as feature engineering, data preprocessing, optimal k selection, sparsity handling, evaluation metrics, user interface design, addressing the cold start problem, scalability, and privacy and security measures are discussed to ensure the robustness and effectiveness of the recommendation system. We propose a comprehensive movie recommendation system utilizing the k-nearest neighbors (KNN) algorithm to provide users with personalized movie suggestions. We employ techniques such as cosine similarity and weighted averages to refine the recommendation process, enhancing its accuracy and relevance

2. INTRODUCTION

The proliferation of digital streaming platforms has led to an overwhelming abundance of movie choices, making it increasingly challenging for users to discover content that aligns with their preferences. To address this issue, we present a movie recommendation system powered by the k-nearest neighbors (KNN) algorithm. This system aims to alleviate the burden of choice by delivering personalized movie suggestions tailored to individual user preferences. By analyzing user ratings and movie features from a comprehensive dataset, our recommendation engine identifies users with similar tastes and recommends movies highly rated by these similar users. We incorporate advanced techniques such as cosine similarity and weighted averages to enhance recommendation accuracy. Through rigorous experimentation and evaluation, we assess the effectiveness of our approach in providing relevant and personalized movie recommendations. Ultimately, our system seeks to enhance user satisfaction and engagement on movie streaming platforms by offering curated content suggestions, thus enriching the overall viewing experience. Ultimately, our system has the potential to significantly enhance user satisfaction and engagement on movie streaming platforms by offering tailored content suggestions. By providing users with movies aligned with their preferences, we aim to improve their overall viewing experience and encourage continued engagement with the platform. In the following sections of this report, we will delve into the methodology, implementation, results, and discussions regarding our movie recommendation system, highlighting its significance and implications in the realm of digital content consumption.

3. SYSTEM ANALYSIS

3.1 Existing System

In the existing landscape of movie recommendation systems, several approaches have been employed to assist users in discovering relevant content. One prevalent method is collaborative filtering, which relies on user-item interaction data to make recommendations. While collaborative filtering techniques have been successful in many applications, they also have limitations. For instance, they often struggle with the cold start problem, where new users or items have insufficient data for accurate recommendations. This approach recommends items similar to those previously rated highly by the target user.

3.2 Disadvantages

- New users or items with limited data make it hard for collaborative filtering to suggest accurate recommendations until they have enough interactions.
- Collaborative filtering relies on user-item interactions, often resulting in sparse data. This makes it challenging to accurately compute similarities between users or items, especially in large datasets.
- Collaborative filtering tends to recommend popular items because they have more ratings, which can lead to a lack of diverse suggestions, neglecting niche interests.
- As the number of users grows, user-based collaborative filtering struggles with computation, making it slower and less efficient.
- Collaborative filtering needs good data. If the ratings or item info isn't right, the recommendations might not be any good.

3.3 Proposed System

The proposed movie recommendation system employs the k-nearest neighbors (KNN) algorithm to deliver personalized movie suggestions tailored to individual user preferences. Leveraging a dataset containing user ratings and movie features, the system first analyzes user preferences by identifying the k most similar users to a target user. This is achieved through similarity calculation techniques such as cosine similarity. Once the most similar users are identified, the system recommends movies highly rated by these users to the target user.

User Preference Analysis:

The system first gathers and analyzes user preferences by leveraging a dataset containing user ratings and movie features. This dataset serves as the basis for understanding user behavior and preferences.

Similarity Calculation:

Using the KNN algorithm, the system identifies the k most similar users to a target user based on their preferences. It computes the similarity between users using techniques such as cosine similarity, which measures the angle between user preference vectors.

Recommendation Generation:

Once the most similar users are identified, the system recommends movies that these users have rated highly. By leveraging the collective preferences of similar users, the system aims to provide relevant and personalized movie recommendations to the target user.

Enhanced Recommendation Accuracy:

To further improve recommendation accuracy, the system incorporates techniques such as weighted averages. This allows the system to give more weight to ratings from users who are more similar to the target user, thus refining the recommendation process.

User Engagement Analysis:

Alongside preference analysis, the system also assesses user engagement metrics, such as the frequency of movie ratings, duration of viewing sessions, and interaction patterns with the recommendation interface. This helps in understanding not only what users prefer but also how actively they engage with the platform.

Genre-specific Recommendations:

In addition to general movie recommendations, the system provides genre-specific suggestions tailored to users' preferences. By identifying genres preferred by similar users, it offers a more targeted and diverse selection of movies to choose from.

3.4 Advantages

- Tailored Recommendations: The system suggests movies personalized to each user's preferences, making it easier to find content .
- Better Matches: By analyzing similar users and preferences, the system suggests movies closely aligned with what the user likes, improving the chances of finding a good match.
- Improved Accuracy: Techniques like weighted averages help make recommendations more accurate, ensuring they reflect the user's tastes more precisely.

- Wider Variety: While focusing on personalization, the system still offers a diverse range of movie options, catering to different interests and tastes.
- Always Up-to-Date: Recommendations are continuously updated to keep pace with changing preferences, ensuring they remain relevant and appealing over time.
- Easy to Understand: The system explains why it suggests certain movies, helping users understand the reasoning behind the recommendations.
- Engaging Experience: With features like feedback loops and personalized notifications, users are encouraged to interact more, making the movie-watching experience more enjoyable and interactive.

4. FEASIBILITY STUDY

4.1 Economical Feasibility:

The economic feasibility study involves estimating the costs of system development, including software, hardware, licensing, and personnel, comparing them to projected benefits such as improved efficiency, reduced wastage, and increased customer satisfaction, and conducting a cost-benefit analysis to determine the project's return on investment (ROI) and payback period.

4.2 Technical Feasibility:

Evaluate the technical capabilities and expertise of the development team in implementing the proposed system. Assess the availability of necessary technologies, frameworks, and tools required for system development, including natural language processing libraries, machine learning algorithms, and database management systems. Determine whether the proposed features, such as sentiment analysis and chatbot integration, can be feasibly implemented within the project timeline and budget constraints.

4.3 Operational Feasibility:

Evaluate the impact of the proposed system on existing canteen operations, workflows, and staff responsibilities. Assess the readiness of canteen staff to adapt to the new system, including training needs and change management requirements. Identify potential operational challenges and risks associated with system implementation, such as disruptions during transition periods and resistance to change from stakeholders.

5. SYSTEM SPECIFICATIONS

5.1 Hardware Specification:

- System : I3 or above
- RAM : 4 GB
- Hard Disk : 40 GB
- Floppy Drive: 1.44 MB
- Processor : Intel

5.2 Software Specification:

- Operating System : Windows xp professional
- Front-end : HTML, CSS, JavaScript
- Software used : Visual Studio Code
- Backend : My SQL,Django
- Coding language :Python
- IDE :Python IDLE3

6. SOFTWARE DESCRIPTION

6.1 Front End

HTML

HTML stands for Hypertext Markup Language, and it is the most widely used language to write Web Pages.

- **Hypertext** refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
- As its name suggests, HTML is a **Markup Language** which means you use HTML to simply "mark-up" a text document with tags that tell a Web browser how to structure it to display.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a webpage semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as <**img** /> and <**input** /> directly introduce content into the page. Other tags such as <**p**> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

CSS

Css stands for "Cascading Style Sheet." Cascading style sheets are used to format the layout of Web Pages. They can be used to define text styles, table sizes, and other aspects of Web pages that previously could only be defined in a page's HTML. CSS helps Web developers create a uniform look across several pages of a Web site. Instead of defining the style of each table and each block of text within a page's HTML, commonly used styles need to be defined only once in a CSS document. Once the style is defined in cascading style sheet, it can be used by any page that references the CSS file. Plus, CSS makes it easy to change styles across several pages at once.

JavaScript

JavaScript (JS) is a versatile programming language primarily used for web development. It enables developers to create interactive and dynamic web pages by manipulating the Document Object Model (DOM) and responding to user actions. JavaScript runs on the client-side, meaning it executes on the user's web browser, allowing for real-time updates and interactivity without requiring page reloads. Its syntax is similar to other programming languages like Java and C, making it relatively easy to learn for developers. JavaScript also has a vast ecosystem of libraries and frameworks that extend its capabilities and simplify common tasks, making it a cornerstone of modern web development.

Features of JavaScript

- Client-Side Scripting: JavaScript is primarily used for client-side scripting, allowing developers to create interactive web pages that respond to user actions.
- Dynamic Content: JavaScript enables the manipulation of HTML and CSS elements, facilitating the creation of dynamic and engaging user interfaces.

- Event Handling: It supports event-driven programming, allowing developers to define actions in response to user interactions such as clicks, mouse movements, and keyboard input.
- Cross-Browser Compatibility: JavaScript is supported by all major web browsers, ensuring consistent functionality across different platforms and devices.
- Asynchronous Programming: JavaScript supports asynchronous programming, enabling the execution of tasks without blocking the main thread, which is essential for handling tasks such as fetching data from servers.
- Extensibility: JavaScript has a vast ecosystem of libraries and frameworks like jQuery, React.js, and Vue.js, which extend its capabilities and simplify common tasks.
- Object-Oriented: JavaScript is an object-oriented language, allowing developers to create reusable code through the use of objects and classes.

6.2 Back End

Python

Python is an interpreted high-level general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs as well as its object-oriented approach aim to help object oriented write clear, logical code for small and large-scale projects. Python is dynamically typed and garbage collected. It supports multiple programming paradigms including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Features

- **Easy to code:**

  Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, JavaScript, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer-friendly language.

- **Object-Oriented Language:**

  One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

- **GUI Programming Support:**

  Graphical Use interfaces can be made using a module such as PyQt5, PyQt4, wx python, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

- **Extensible feature:**

  Python is a **Extensible** language. We can write us some Python code into C or C++ language and also we can compile that code in C/C++ language

- **Large Standard Library**

  Python has a large standard library which provides a rich set of module and functions so you do not have to write your own code for every single thing.

Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It follows the "Don't Repeat Yourself" (DRY) principle, enabling developers to build web applications quickly and efficiently by providing a set of tools and libraries for common tasks such as URL routing, database manipulation, and template rendering. Django's "batteries-included" philosophy means it comes with built-in features like authentication, security, and administration panels, reducing the need for external libraries. It also emphasizes scalability, security, and maintainability, making it a popular choice for developing a wide range of web applications, from simple blogs to complex enterprise systems. With its extensive documentation, vibrant community, and active ecosystem of third-party packages, Django remains one of the most powerful and versatile web frameworks available for Python developers.

Features of Django

1. Batteries-included: Comes with built-in features like ORM, authentication, and admin interface.
2. MTV Architecture: Organizes code into models, templates, and views for clarity and maintainability.
3. Automatic Admin Interface: Generates admin panels for managing site data without extra coding.
4. URL Routing and View System: Maps URLs to views for handling HTTP requests efficiently.
5. Security Features: Provides protection against common security threats and built-in support for authentication and session manageme

7. PROJECT DESCRIPTION

7.1. About The Project

The project movie recommendation system is implemented using the k-nearest neighbors (KNN) algorithm. Analyzing user preferences and similarities, it offers personalized movie suggestions. Leveraging user ratings and movie features, the system enhances accuracy with techniques like cosine similarity. Rigorous evaluation ensures effective and relevant recommendations, enhancing user satisfaction on streaming platforms.

.Key Features:

1. Personalized Recommendations: Tailors movie suggestions based on individual user preferences.
2. KNN Algorithm Implementation: Utilizes the k-nearest neighbors algorithm for recommendation generation.
3. User Preference Analysis: Gathers and analyzes user preferences from a dataset containing ratings and movie features.
4. Similarity Calculation: Computes similarity between users using techniques like cosine similarity.
5. Enhanced Recommendation Accuracy: Incorporates techniques such as weighted averages to improve recommendation precision.
6. User Engagement Features: Includes features like feedback loops, personalized notifications, and community-based recommendations to enhance user engagement.
7. Dynamic Recommendation Updates: Continuously updates recommendations to reflect evolving user preferences and new data.

7.2 Module Description

There are three main modules in this project

- Admin module
- User module
- Movie Recommendation Module

ADMIN MODULE

The Admin module provides functionalities for managing the movie recommendation system. Administrators can oversee user management, including adding, updating, or deleting user accounts. They can also manage the movie database, adding new movies, updating movie information, or removing outdated entries. Additionally, administrators have access to system settings, allowing them to configure parameters such as the number of nearest neighbors considered in recommendation generation. The Admin module ensures the smooth operation and maintenance of the recommendation system, enabling efficient management of users, movies, and system configurations.

USER MODULE

The User module serves as the interface for end-users interacting with the movie recommendation system. Users can create accounts, log in, and personalize their profiles by specifying their movie preferences and viewing history. They can search for movies, view detailed information about each movie, and provide ratings and feedback. The User module also facilitates interactions such as bookmarking favorite movies, viewing recommended movies, and accessing personalized recommendations based on their preferences. Through the User module, users can seamlessly explore and engage with the movie streaming platform, enhancing their overall experience.

MOVIE RECOMMENDATION MODULE

The Movie Recommendation module forms the core functionality of the system, utilizing the k-nearest neighbors (KNN) algorithm to generate personalized movie recommendations for users. Leveraging a dataset containing user ratings and movie features, this module identifies the k most similar users to a target user and recommends movies highly rated by these similar users. Techniques such as cosine similarity and weighted averages are incorporated to enhance recommendation accuracy. The Movie Recommendation module ensures that users receive relevant and personalized movie suggestions, ultimately enhancing user satisfaction and engagement on movie streaming platforms.

8. SYSTEM DESIGN

8.1 Data Flow Diagram

Data flow diagrams are used widely for modeling the requirements. DFD's show the flow of data through a system. The system may be a company, an organizational set of procedures, a computer hardware system, a software system or any combination of the proceedings. The DFD also known as a data flow graph or a bubble chart. The following observations about DFDs are important:

1. All names should be unique. This makes easier to refer items in the DFD. Remember that a DFD is not a flow chart.

2. Arrows in a flow chart represent the order of events; arrows in DFD represent flowing data. A DFD doesn't imply any order of events.

3. Suppress logical decisions (A diamond shape box is used in flow chart to represent decision points with multiple exit paths of which only one is taken). This implies an ordering of events, which makes no sense in DFD.

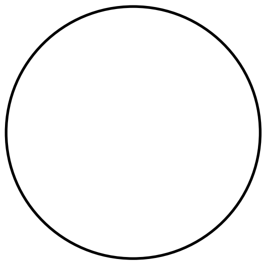
BASIC DFD SYMBOLS

1. ARROW

A data flow is a route, which enables packets of data to travel from one point to another. An arrow indicates data flow-data in motion. It is a pipeline through which information flows.

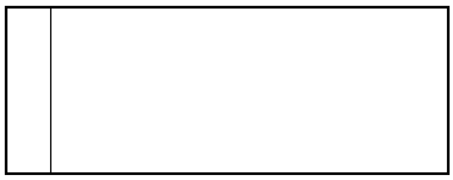
2. PROCESS

A process represents transformations where incoming data flows are changed into outgoing data flows. A “circle” or a “bubble” represents a process that transforms incoming data flows into outgoing data flows



3. DATA STORE

A data store represents a repository of data that is to be stored for use by one or more processes may be symbol as buffer or queue or sophisticated as relational database. They should have clear names



4. SOURCE

A source or sink is a person or part of an organization which enter or receives information from the system but it is considered to be outside the context of data flow models.

ZEROTH LEVEL

LEVEL 0

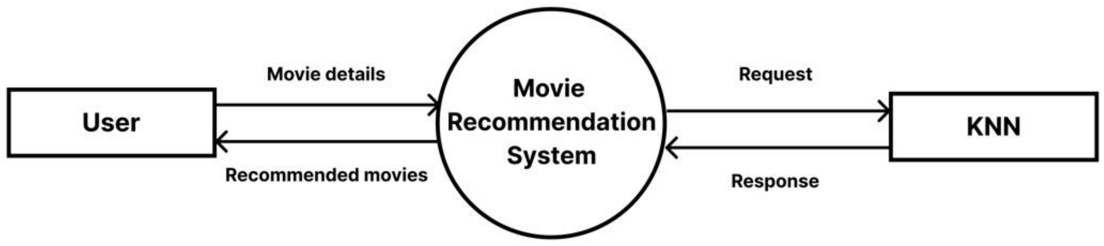


Fig4.1 Level 0

ADMIN MOVIE LEVEL

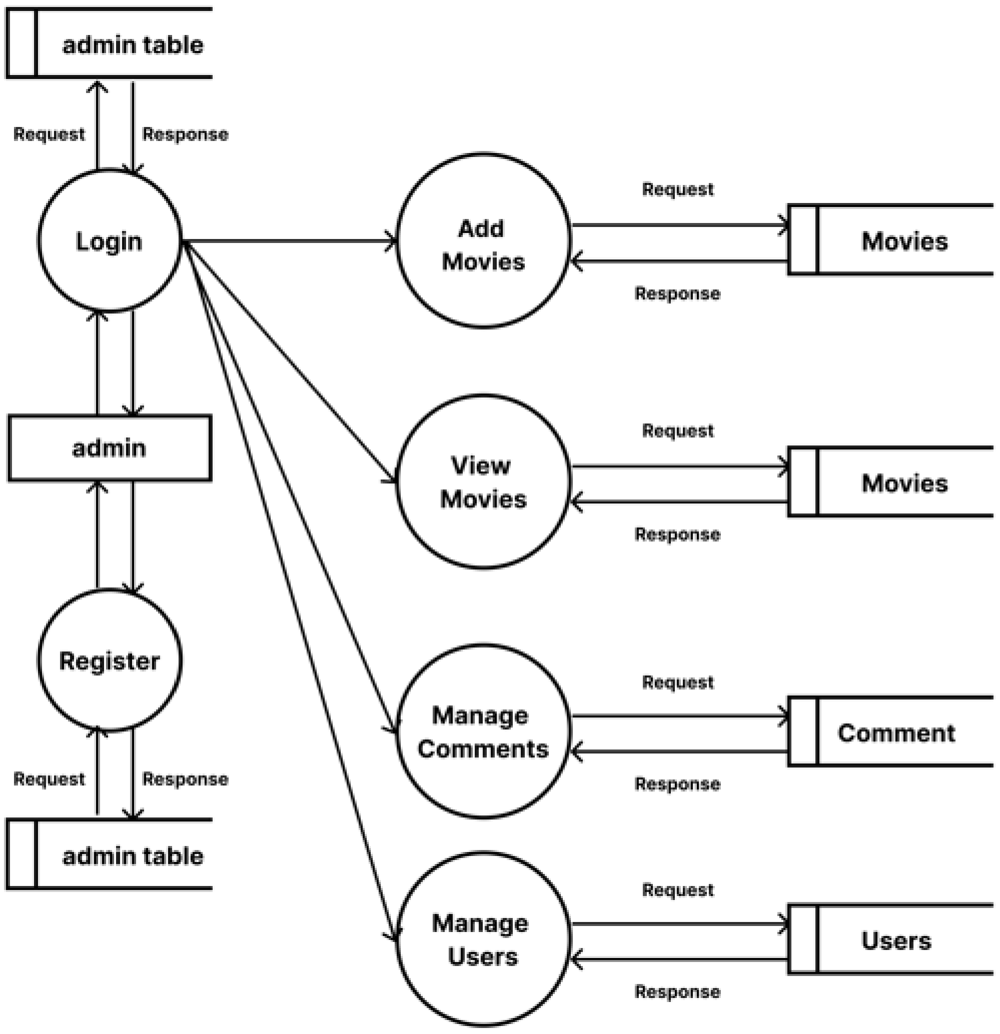


Fig4.2 Admin Level

MOVIE LEVEL

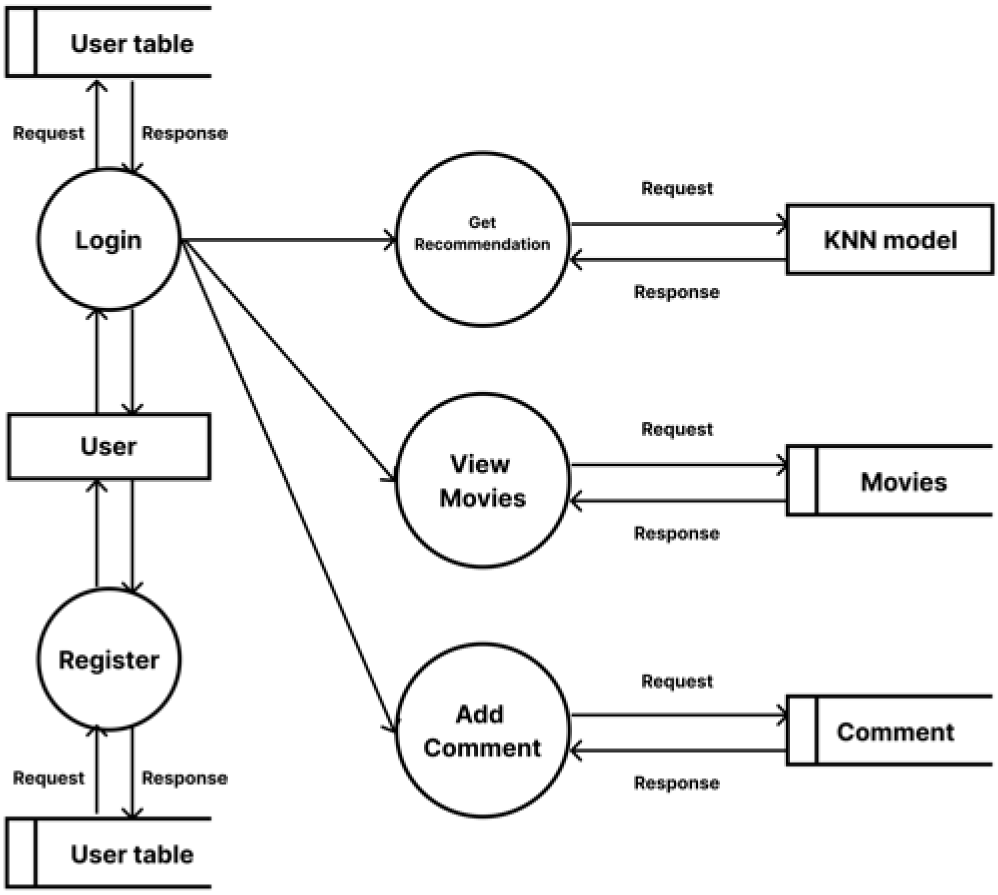
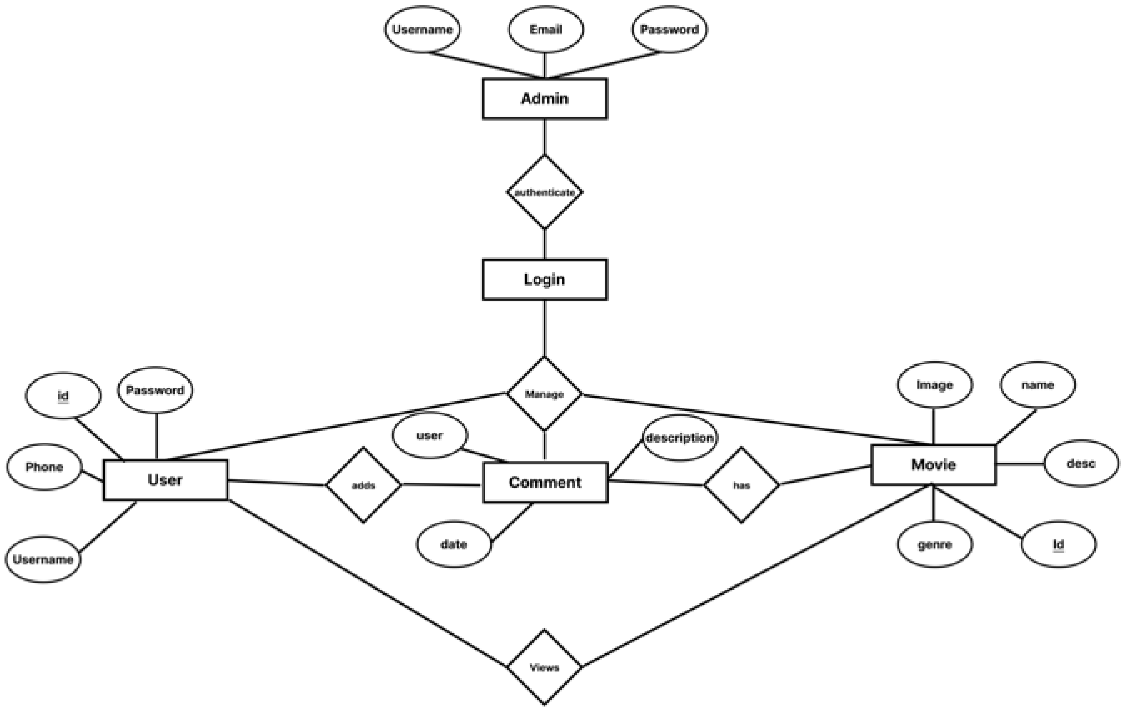


Fig4.3 User Level

8.2 E.R Diagram



8.3 DATABASE DESIGN

Sl.no	Attribute	Type	Constraints	Description
1	Username	VARCHAR(10)	NOTNULL	Username
2	Email	VARCHAR(30)	PRIMARY KEY	Email
3	Password	VARCHAR(10)	NOTNULL	Password

Table no 4.1 USER

Sl.no	Attribute	Type	Constraints	Description
1	Movie Name	VARCHAR(10)	NOTNULL	Movie Name
2	Rating	INT	NOTNULL	Rating
3	Likes	INT	NOTNULL	Likes
4	Description	VARCHAR(10)	NOTNULL	Description

Table no 4.2 MOV

8.4 Input Design

Input design is the process of converting user-oriented inputs to a computer-based format. The quality of the system input determines the quality of the output. Inaccurate input data are the most common cause of errors in data processing. Errors entered by the data entry operator can be controlled by input design. The goal of designing input data is to make the processing easy and free from errors

Input design is part of overall system design, which requires careful attention. Input can be categorized as internal, external, operational, computerized and interactive. The analysis phase should consider the impact of the inputs on the system as a whole and on the other system.

Login forms

User registration form

8.5 Output Design

In the output design, it is determined how the information is to be displayed for intermediate need. Computer output is the most important and direct source of information to the users. The right output must be developed while ensuring that each output element is designed so that people will find easy to use the system.

The successes and failure of the system depends on the output, though a system looks attractive and user friendly, the output it produces decides upon the usage of the system. The outputs generated by the system are checked for its consistency, and output is provided simple so that user can handle them with ease.

View user form

View register for

9. SYSTEM TESTING

9.1 UNIT TESTING

It is the process of taking each program module and it in isolation from the rest of the modules, by using prepared inputs and comparing the actual results with the results predicated by the specifications and design of modules. This enables the tester to detect errors in coding and logic that are contained within that module alone.

The software units in a system are modules and routines that are assembled and integrated to perform a specific function. Unit testing focuses first on modules, independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained within each module. This testing includes entering data and ascertaining if the value matches to the type and size supported by python. The various controls are tested to ensure that each performs its action as required. This is known as “Module testing”. This testing is carried out during programming stage.

Project aspect: Front-end design consists of various forms. They are tested for data acceptance. Similarly the back-end that is the database was also tested for successful acceptance and retrieval of data. It first checks the design module, to conform all the graphical animated images are working properly. Then it checks the dictionary module, to conform all the phrases or words available. It also checks if still images and turnoff are working properly.

9.2 INTEGRATION TESTING

It is the schematic technique for constructing the program structure while at the same time conducting tests to see uncovered errors associated with interfacing. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation. The primary concern is the capability of individual modules.

Data can be lost across any interface; one module have an adverse effect on another and sub-functions when combined may not produce the desired major functions. Integration testing is a systematic testing to discover errors associated within interface. The objective is to take unit tested modules and build a program structure. All the modules are combined and tested and as a whole

We followed bottom-up integration testing. Bottom-up integration testing as its name implies begins construction and test with atomic modules. Because components are integrated from bottom-up, processing required for components subordinate to a given level is always available and the need for stubs is eliminated. The bottom-up integration testing is done from the fault and fault free module is integrated with the work stealing module

9.3 VALIDATION TESTING

Validation testing can be defined in many ways, but a simple definition is that validation succeeds when the software functions in manner that is reasonably expected by the customer. Software validation is achieved through a series of black box tests that demonstrate conformity with requirement. After validation test has been conducted, one of two condition occurs.

- The function or performance characteristics confirm to specifications and are accepted.
- A validation from specification is uncovered and a deficiency created

Deviations or errors discovered at this step in the project is corrected prior to completion of the project with the help of the user by negotiating to establish a method for resolving deficiencies. Thus the proposed system under consideration has been tested by using validation testing and found to be working satisfactorily.

9.3 BLACK BOX TESTING

Black box testing for the College Canteen App evaluates functionality without accessing internal code. It ensures reliability and effectiveness by validating ordering processes, feedback submission accuracy, and chatbot interaction. User authentication and authorization tests secure login and access control. Cross-platform compatibility ensures consistent performance. Performance and scalability tests assess system capability under varying loads, while security testing identifies vulnerabilities to safeguard data. Through comprehensive testing, we ensure the system meets high standards of functionality, usability, security, and reliability, providing an efficient solution for canteen management.

9.4 WHITE BOX TESTING

White box testing for the College Canteen Management System involves examining its internal code and structure to ensure thorough test coverage. This method focuses on validating the correctness of individual components, including algorithms, data structures, and logic pathways. Tests are designed to verify the accuracy of calculations, proper handling of user inputs, and adherence to coding standards. Additionally, code coverage analysis is performed to ensure that all sections of the code are tested adequately. Through rigorous white box testing, we aim to identify and address any underlying issues or vulnerabilities within the system's architecture, ensuring its stability, reliability, and efficiency.

10. SYSTEM IMPLEMENTATION

10.1 System Maintenance

System maintenance for the College Canteen App involves a comprehensive approach to ensure its reliability, security, and efficiency over time. This includes regular updates to incorporate new features and bug fixes, keeping the system up-to-date and functional. Continuous performance monitoring helps identify and address any bottlenecks or slowdowns, optimizing system resources for efficient operation. Data backup and recovery procedures are essential to prevent loss in case of system failures or data corruption. Applying security patches and updates protects against evolving threats, safeguarding sensitive data and ensuring compliance with regulations. Ongoing user training and support are provided to ensure effective system utilization and address user concerns. Feedback collection and analysis help identify areas for improvement, while scalability planning ensures the system can accommodate future growth effectively. Through these maintenance practices, the College CanteenApp remains robust and reliable, providing consistent service to users.

11. CONCLUSION AND FUTURE EHANCEMENTS

11.1Conclusion

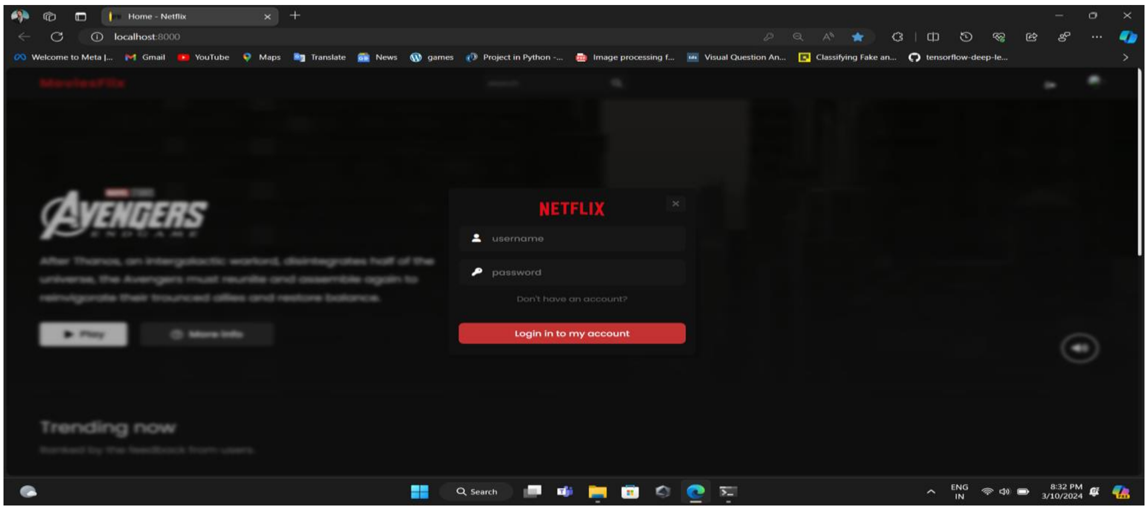
In conclusion, the movie recommendation system utilizing the k-nearest neighbors (KNN) algorithm has demonstrated its effectiveness in delivering personalized and relevant movie suggestions to users. Through the analysis of user preferences and similarities with other users, the system offers tailored content recommendations, thereby enhancing user satisfaction and engagement on movie streaming platforms. By incorporating techniques such as cosine similarity and weighted averages, the system improves recommendation accuracy, ensuring that users receive suggestions aligned with their tastes. Rigorous experimentation and evaluation have validated the efficacy of the approach in providing personalized movie recommendations. Moving forward, the system holds the potential to further enhance user experience by implementing additional features such as content-based filtering, collaborative filtering, and hybrid recommendation approaches.

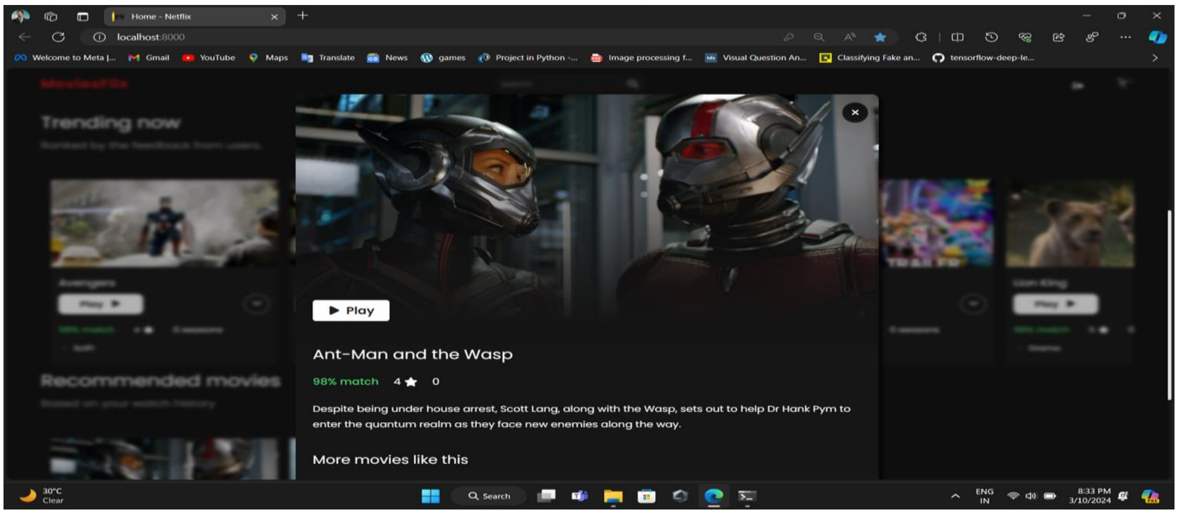
11.2 Scope For The Future Enhancements

TIn the future, the movie recommendation system could benefit from several enhancements to further improve its functionality and user experience. Integrating content-based filtering techniques would enable the system to consider movie attributes such as genre, director, and cast, providing more diverse and nuanced recommendations. Collaborative filtering methods could be incorporated to leverage user-item interactions and enhance recommendation accuracy, especially for new users or items. Additionally, hybrid recommendation approaches combining collaborative filtering, content-based filtering, and matrix factorization techniques could be explored to leverage the strengths of each method and provide even more personalized recommendations. Furthermore, incorporating real-time updates and user feedback mechanisms would ensure that recommendations remain relevant and reflective of evolving user preferences over time.

APPENDIX A

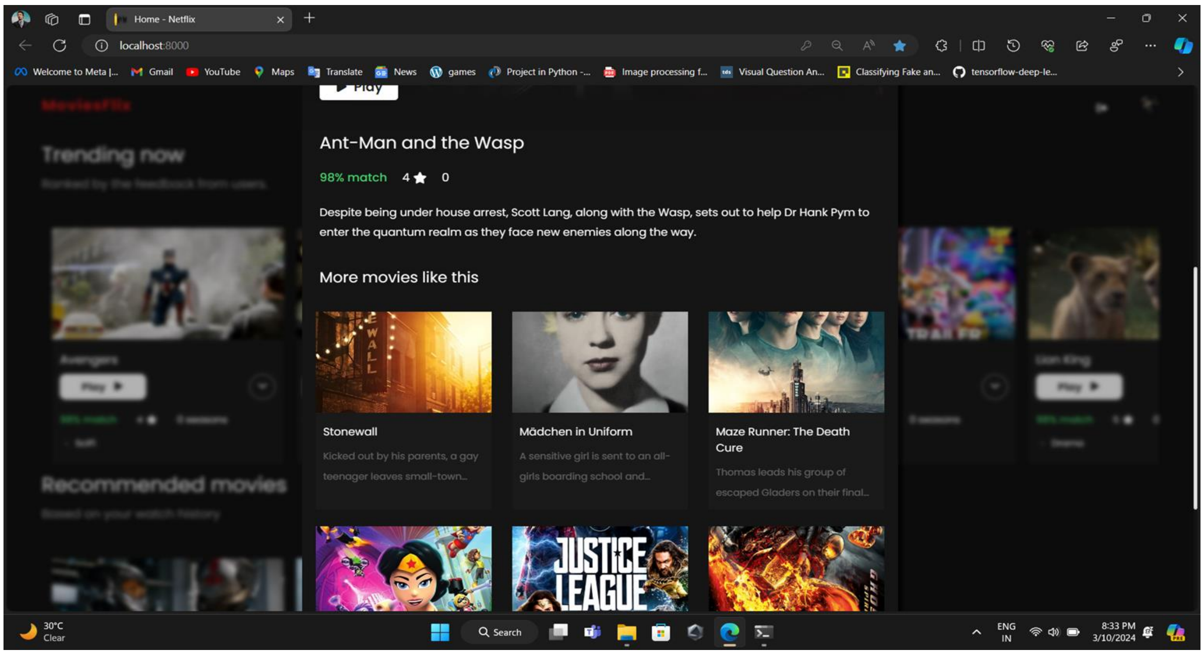
SCREENSHOTS





Ant-Man and the Wasp

Despite being under house arrest, Scott Lang, along with the Wasp, sets out to help Dr Hank Pym to enter the quantum realm as they face new enemies along the way.


Ant-Man and the Wasp

98% match 4 ★ 0

Despite being under house arrest, Scott Lang, along with the Wasp, sets out to help Dr Hank Pym to enter the quantum realm as they face new enemies along the way.

More movies like this

Stonewall

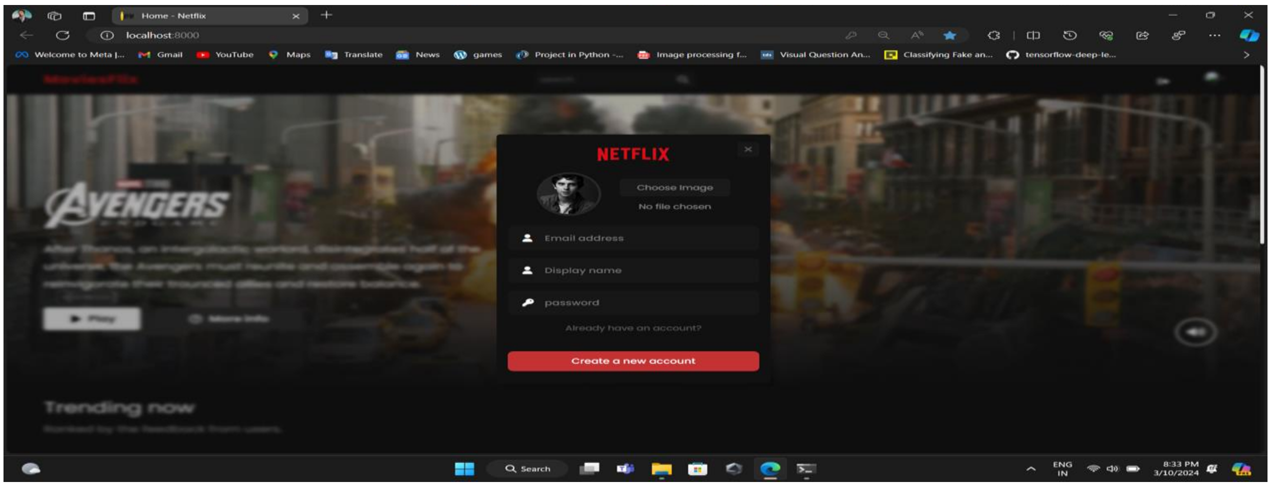
Kicked out by his parents, a gay teenager leaves small-town...

Mädchen in Uniform

A sensitive girl is sent to an all-girls boarding school and...

Maze Runner: The Death Cure

Thomas leads his group of escaped Gladers on their final...



SOURCE CODE

View code - views.py

```
from django.shortcuts import render

from .models import Movie, userProfile,commentUser, Genre

from django.contrib.auth.models import User

from django.http import HttpResponseRedirect, HttpResponse, JsonResponse

from django.urls import reverse

from django.contrib.auth import authenticate,logout,login

from django.contrib.auth.decorators import login_required

import pandas as pd

import numpy as np

from scipy.sparse import csr_matrix

from sklearn.neighbors import NearestNeighbors

import matplotlib.pyplot as plt

import seaborn as sns

movies = pd.read_csv("model/movies (1).csv")

ratings = pd.read_csv("model/ratings.csv")

import pickle
```

```
from django.core.files.storage import FileSystemStorage

import random

from sklearn.neighbors import NearestNeighbors

import numpy as np

# Create your views here.

import openai


openai.api_key = 'sk-
qPsFuhET3cl5J0NWz6sqT3BlbkFJwmG2gIgZIGYy4FkLn1j4'


def homepage(request):

    try:

        profile = userProfile.objects.filter(user=request.user).first()

        favorite_genres = profile.genres.all()

        user_data = []

        for genre in favorite_genres:

            movies = Movie.objects.filter(genre=genre)

            for movie in movies:

                user_data.append([profile.id, movie.id, 1])
        user_data = np.array(user_data)



        movie_features = [...]



        knn = NearestNeighbors(n_neighbors=5, metric='cosine')

        knn.fit(movie_features)



        _, indices = knn.kneighbors(user_data[:, 1:])

        recommended_movies = set()

        for index_list in indices:

            for index in index_list:

                recommended_movies.add(movie_features[index])



    except:
        recommended_movies = []


    main_movie = Movie.objects.all().first()

    movies = Movie.objects.all()


    return render(request, 'homepage.html', {"main_movie": main_movie,
"movies": movies, "recommended_movies": recommended_movies})
def checkSignup(request):


    username = request.POST.get('username')

    password = request.POST.get('password')




    u = User.objects.filter(username = username).first()


    if u == None:

        message = 0

    else:

        message = 1
    return JsonResponse({"message":message})


def register(request):
    if request.method == 'POST':
        try:
            file = request.FILES.get('file')
            fss = FileSystemStorage()
            filename = fss.save(file.name,file)
            url = fss.url(filename)
        except:
            pass

        user = User.objects.create(username = request.POST.get('username'),email
= request.POST.get('email'))
        user.set_password(request.POST.get('password'))
        user.save()

        user = User.objects.filter(username=request.POST.get('username')).first()
        try:

            profile = userProfile.objects.create(user=user,  image=url)

            profile.save()

        except:

            profile = userProfile.objects.create(user=user)

            profile.save()




    return HttpResponseRedirect(reverse('homepage'))



def checkLogin(request):


    username = request.POST.get('username')

    password = request.POST.get('password')


    user = authenticate(username = username,password = password)
    if user:

        print(username)

        return JsonResponse({"message":0})



    else:

        print("No user found")

        return JsonResponse({"message":1})



def user_login(request):



    if request.method == 'POST':

        username = request.POST.get('username')

        password = request.POST.get('password')



        user = authenticate(username = username,password = password)

        if user:



            if user.is_active:
                login(request, user)

                print("login success!!!")

                return HttpResponseRedirect(reverse('homepage'))

            else:



              print("No such user")





    return HttpResponseRedirect(reverse('homepage'))



@login_required

def user_logout(request):



    logout(request)





    return HttpResponseRedirect(reverse('homepage'))
```

```
final_dataset = ratings.pivot(index='movieId',columns='userId',values='rating')

final_dataset.fillna(0,inplace=True)


no_user_voted = ratings.groupby('movieId')['rating'].agg('count')

no_movies_voted = ratings.groupby('userId')['rating'].agg('count')


final_dataset=final_dataset.loc[:,no_movies_voted[no_movies_voted >
50].index]


sample = np.array([[0,0,3,0,0],[4,0,0,0,2],[0,0,0,0,1]])

sparsity = 1.0 - ( np.count_nonzero(sample) / float(sample.size) )


csr_sample = csr_matrix(sample)

csr_data = csr_matrix(final_dataset.values)

final_dataset.reset_index(inplace=True)
```

```
knn = NearestNeighbors(metric='cosine', algorithm='brute', n_neighbors=20,
n_jobs=-1)
knn.fit(csr_data)


def get_movie_recommendation(movie_name):
    n_movies_to_reccomend = 10
    movie_list = movies[movies['title'].str.contains(movie_name)]
    if len(movie_list):
        movie_idx= movie_list.iloc[0]['movieId']
        movie_idx = final_dataset[final_dataset['movieId'] == movie_idx].index[0]
        distances , indices =
knn.kneighbors(csr_data[movie_idx],n_neighbors=n_movies_to_reccomend+1)
        rec_movie_indices =
sorted(list(zip(indices.squeeze().tolist(),distances.squeeze().tolist())),key=lambd
a x: x[1])[:0:-1]
        recommend_frame = []
        for val in rec_movie_indices:
            movie_idx = final_dataset.iloc[val[0]]['movieId']
            idx = movies[movies['movieId'] == movie_idx].index
```

```
recommend_frame.append({'Title':movies.iloc[idx]['title'].values[0],'Distance':v
al[1]})
        df =
pd.DataFrame(recommend_frame,index=range(1,n_movies_to_reccomend+1))
        return recommend_frame
    else:
        return "No movies found. Please check your input"



# movies = get_movie_recommendation('Edge of Tomorrow')
# print(movies)



def getMovieDetails(request):

    movie = Movie.objects.filter(id = request.POST.get('movie-pk')).first()
    profile = userProfile.objects.filter(user = request.user).first()
    print(str(movie.genre))
    genre = Genre.objects.filter(name = movie.genre.first().name).first()

    profile.genres.add(genre)

    profile.save()

    print(genre)

    rec = get_movie_recommendation(movie.name)

    comment = commentUser.objects.filter(movie = movie)

    cm = []

    for c in comment:


cm.append({"user":c.user.username,"image":c.image,"text":c.text,"sentiment":c
.sentiment})

    print(cm)

    return
JsonResponse({"name":movie.name,"rating":movie.rating,"image":movie.imag
e.url,"video":movie.video.url,"description":movie.description,"seasons":movie.s
easons,"recommended":rec,"comments":cm,"pk":movie.pk})


def search(request):

    if request.POST.get('search') == "":
        result = 0
    else:
        movie = Movie.objects.filter(name__icontains =
str(request.POST.get('search'))).first()
        try:
            result =
{"name":movie.name,"ratings":movie.rating,"image":movie.image.url,"seasons
":movie.seasons,"pk":movie.pk}
        except:
            result = 0
    return JsonResponse({"result":result})


def comments(request):


    profile = userProfile.objects.filter(user=request.user).first()
    movie = Movie.objects.filter(pk = request.POST.get('pk')).first()


    response = openai.ChatCompletion.create(
    model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You will be provided with a review you
need to identify if its positive or negative or neutral.Just show if its positive or
negative or neutral nothing else"},
            {"role": "user", "content": request.POST.get('comment')},
        ]
        )
    resmsg = response['choices'][0]['message']['content']
    c = commentUser.objects.create(user = request.user,image =
profile.image.url,text = request.POST.get('comment'),movie = movie,sentiment
= resmsg)
    c.save()
    return HttpResponseRedirect(reverse('homepage'))


def play(request):
    movie = Movie.objects.filter(pk = request.POST.get('pk')).first()


    return JsonResponse({"result":str(movie.video.url)})
```

BIBLIOGRAPHY

- Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295).
- Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), 5-53.
- Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.
- Leskovec, J., Rajaraman, A., & Ullman, J. D. (2014). Mining of massive datasets. Cambridge University Press.
- Raschka, S., & Mirjalili, V. (2019). Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2 (3rd ed.). Packt Publishing.
- Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: data mining, inference, and prediction (2nd ed.). Springer.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.
- Netflix Prize. (n.d.). Retrieved from https://www.netflixprize.com/
- IMDb Dataset. (n.d.). Retrieved from https://www.imdb.com/interfaces/