MOVIE RECOMMENDATION SYSTEM

A MINI PROJECT REPORT

18CSC207J - ADVANCED PROGRAMMING PRACTICE

Submitted by

PRIT PATEL [RA2111003010237]
JASWANTH YALAVARTHI [RA2111003010238]

Under the guidance of

Jayapradha J

Assistant Professor, Department of Computing Technologies

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled **“Supermarket Management System”** is the bonafide work of **PRIT PATEL (RA2111003010237)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mrs. Jayapradha J
GUIDE
Assistant Professor
Department of Computing Technologies

SIGNATURE

Dr. M. Pushpalatha
HEAD OF THE DEPARTMENT
Professor & Head
Department of Computing Technologies

ABSTRACT

The recommendation system plays an essential role in the modern era and used by many prestigious applications. The recommendation system has made the collection of apps, creating a global village, and growth for abundant information. This paper represents the overview of Approaches and techniques generated in the Collaborative Filtering based recommendation system [1]. The recommendation system derived into Collaborative Filtering, Content-based, and hybrid-based approaches. This paper classifies collaborative filtering using various approaches like matrix factorization, user-based recommendation, item-based recommendation. This survey also tells the road map for research in this area. We extract aspect-based specific ratings from reviews and also recommend reviews to users depends on user similarity and their rating patterns. Finally, validating the proposed movie recommendation system for various evaluation criteria, and also the proposed system shows better result than conventional systems..

Recommender systems are more popular and increase the production costs for many service providers. Today the world is an over-crowded so that the recommendations are required for recommending products or services. However recommender systems minimize the transaction costs and improves the quality and decision making process to users [1], [5], [6]. It is applied in various neighboring areas like information retrieval or human computer interaction (HCI). It gathers huge amount of information about user's preferences of several items like online shopping products, movies, taxi, TV, tourism, restaurants, etc. It stores information of different ways either positive or negative manner. It captures users review for watched movies, traveled places, and purchased products. When compare demand from the shopping products, service providers (travel, and restaurants), movie recommendation system design a big problem since other recommendation systems require fast computation and processing service from service providers and product distributors. To recommend movies, first collects the ratings for users and then recommend the top list of items to the target user [2]. In addition to this, users can check reviews of other users before watching movie. A different recommendation schemes have been presented includes collaborative filtering, content-based recommender system, and hybrid recommender system. However, several issues are raised with users posted reviews.

TABLE OF CONTENTS

ABBREVIATIONS

Abbreviation	**Full Form**
MRS	Movie Recommendation System
PYD	Python Django
IMDb	Internet Movie Database
RT	Rotten Tomatoes
CF	Collaborative Filtering
CBF	Content-Based Filtering
MF	Matrix Factorization
SKL	scikit-learn
TF	TensorFlow
DBMS	Database Management System
UI	User Interface
UX	User Experience
API	Application Programming Interface
CRUD	Create, Read, Update, Delete (database operations)
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
AWS	Amazon Web Services
SQL	Structured Query Language
ORM	Object-Relational Mapping

CHAPTER 1

INTRODUCTION

Reading the local TV guides, renting CDs and DVDs, watching tapes or filmstrip projectors... Today, this is all a relic of the past. The largest movie libraries in the world are all digitized and transferred to online streaming services, like Netflix, HBO, or YouTube. Enhanced with AI powered tools, these platforms can now assist us with probably the most difficult chore of all — picking a movie. A movie recommendation system, or a movie recommender system, is an ML-based approach to filtering or predicting the users' film preferences based on their past choices and behavior. It's an advanced filtration mechanism that predicts the possible movie choices of the concerned user and their preferences towards a domain specific item, aka movie.

Movie Recommender is a movie recommendation system, which recommend users movies which they may like, based on the movies that they previously watched or searched. Every user should have access to the recommender system. The system will go through the movies that user previously searched, then according to those information it should provide movies to the user. The project's main aim is to provide accurate movie recommendations to the user. This project is beneficial for the user as well as the company. For users, they may find movies that they may like without consuming time and even they can encounter new movies which they like from the recommendations. For the company, they make the platform more attractive, so they draw more users to the platform and the system makes the users of the website spend more time online.

CHAPTER 2

LITERATURE SURVEY

From a user's perspective, they are catered to fulfill the user's needs in the shortest time possible. For example, the type of content you watch on Netflix, Amazon Prime or Disney+Hotstar. A person who likes to watch only thriller movies will see titles related to that only but a person who likes to watch Action-based titles will see that on their home screen. From an organization's perspective, they want to keep the user as long as possible on the platform so that it will generate the most possible profit for them. With better recommendations, it creates positive feedback from the user as well. What good it will be to the organization to have a library of 200K+ titles when they cannot provide proper recommendations? The objective of this project is to provide accurate movie recommendations to users. The goal of the project is to improve the quality of movie recommendation system, such as accuracy, quality and scalability of system than the pure approaches

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

The system architecture and design for a Movie Recommendation System can be broken down into several key components, including hardware components and software specifications.

3.1 Data Collection:

1. **Collecting the data sets:** Collection of all the required data set will be from Kaggle website. In this project we require movie.csv, ratings.csv, users.csv.
2. **Data Analysis:** Make sure that that the collected data sets are correct and analyzing the data in the csv file i.e. checking whether the entire column Fields are present in the data sets.

3.2 Hardware Components:

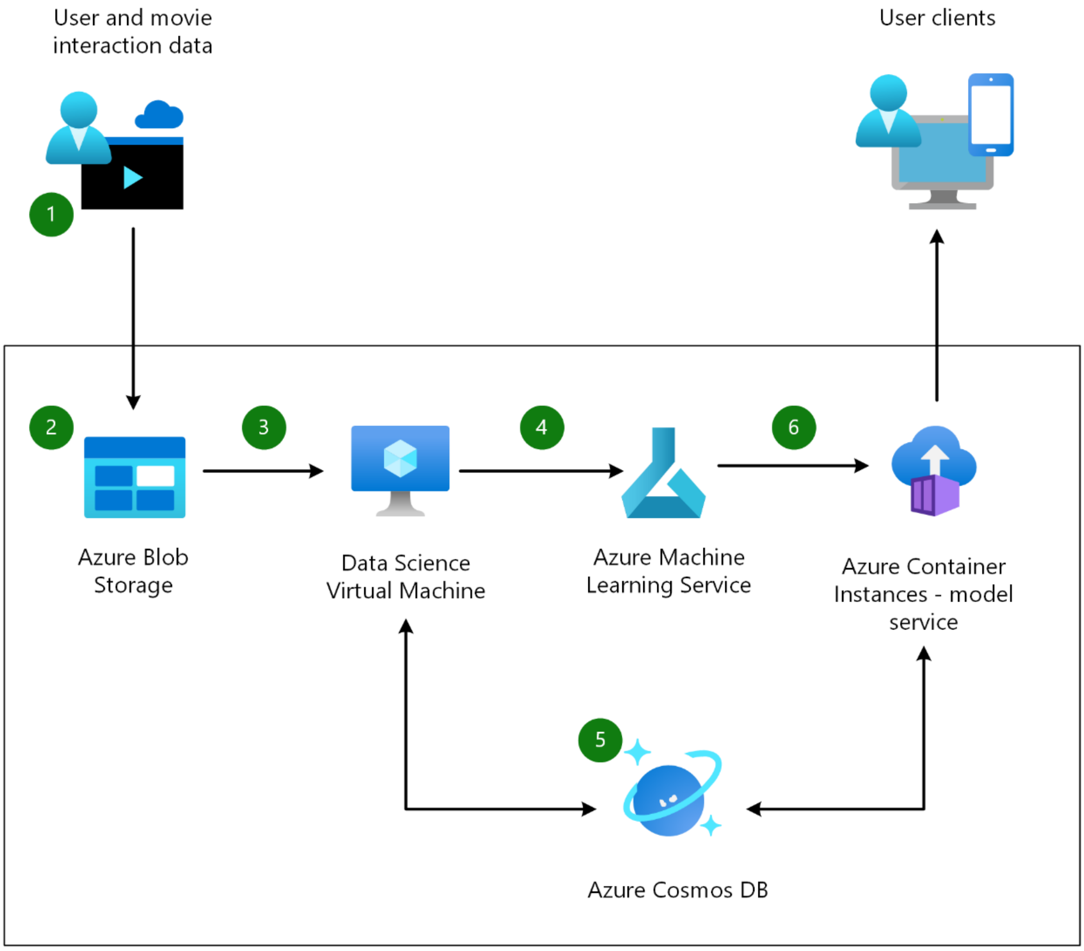
- **Hard Disk**: - 500 GB
- **Processor**: INTEL CORE i3
- **RAM**: - 2 GB
- **Front end**: - Python, PyCharm, Jupyter, Heroku
- **Back end**: - TMDB Dataset

3.3 Software Specifications:

- **Operating System:** Window/ Linux operating system
- **Anaconda distribution**: Anaconda is a free and open-source distribution of the Python programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management system and deployment. Package versions are managed by the package management system conda. The anaconda distribution includes data-science packages suitable for Windows, Linux and MacOS.3
- **Libraries:** For the computation and analysis we need certain python libraries which are used to perform the analytics. Packages such as NumPy, pandas, Jupyter etc are needed.
- **NumPy**: NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python.
- **Pandas:** Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks.

- **Jupyter:** The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.
- **Dataset:** Used the dataset for movies from IMDB.
- Active Internet connection

System Architecture Design for Movie Recommendation System Looks Similar to below Design:


User and movie interaction data
User clients
1
2
3
4
6
5
Azure Blob Storage
Data Science Virtual Machine
Azure Machine Learning Service
Azure Container Instances - model service
Azure Cosmos DB

CHAPTER 4

METHODOLOGY

The methodology for developing a Movie Recommendation System typically involves several steps, including planning and requirement, analysis and design, implementation, testing, and evaluation.

- **Planning and requirement**: All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **Analysis and design:** The requirement specifications from first phase are studied in this phase and system design is prepared. It helps in specifying hardware and system requirements and also helps in defining overall system architecture.
- **Implementation**: With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.
- **Testing:** All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Evaluation:** This will evaluate the outcome of the processing whether the application is running successfully or not.

4.2 REASON FOR CHOOSING WATERFALL MODEL:

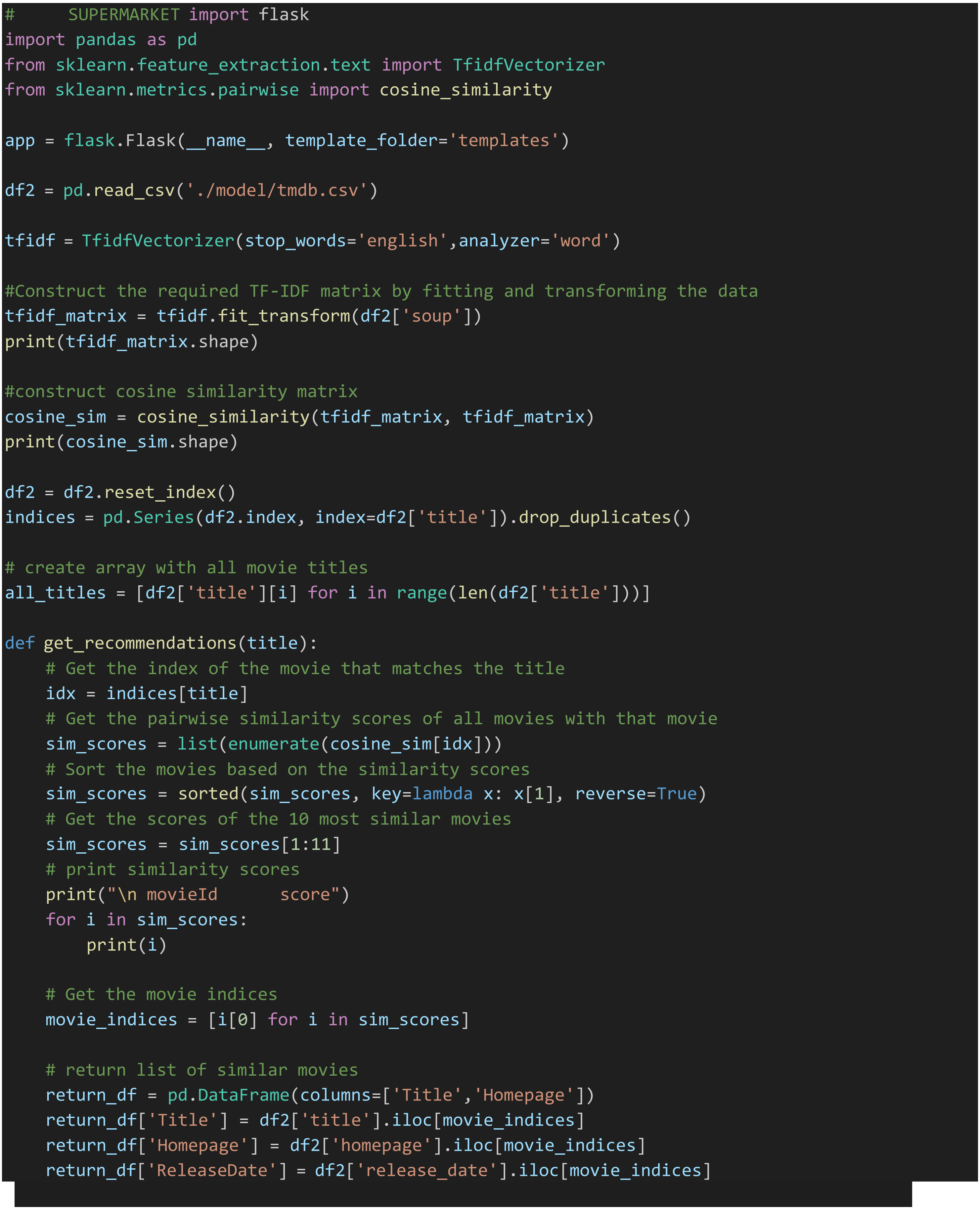
- It is very simple to understand and use.
- As we are new to this it is very difficult to analyse all requirements at once. Building the whole application at once is not possible for us.
- So, by dividing the Project into small parts and implementing them makes easy for us to make this application.

4.3 ADVANTAGES OF THIS MODEL:

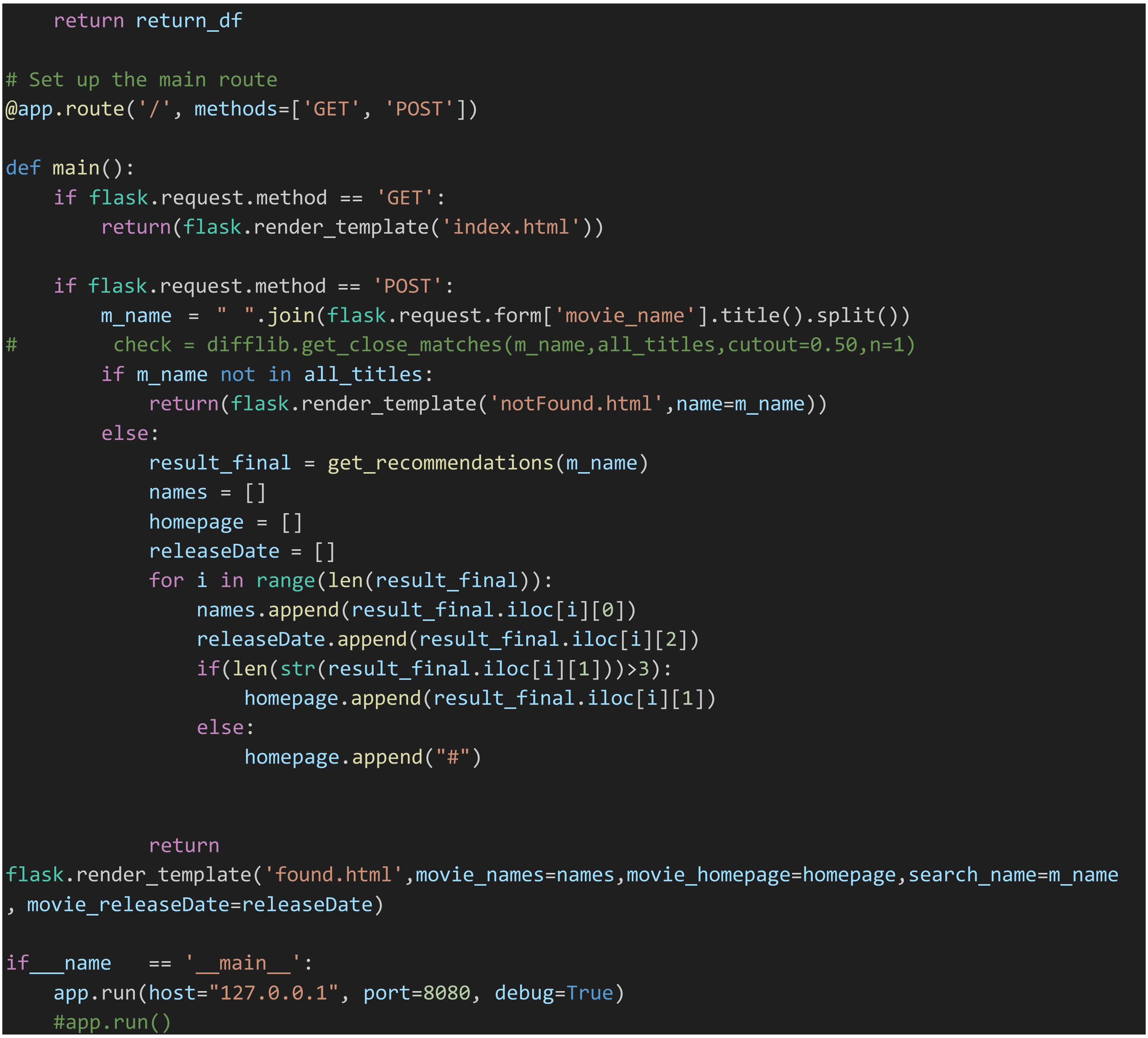
- This model is simple and easy to understand and use.
- It is easy to manage due to the rigidity of the model – each phase has specific deliverables and a review process.
- In this model phases are processed and completed one at a time.

CHAPTER 5

CODING AND TESTING



```
#     SUPERMARKET import flask
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

app = flask.Flask(__name__, template_folder='templates')

df2 = pd.read_csv('./model/tmdb.csv')

tfidf = TfidfVectorizer(stop_words='english',analyzer='word')

#Construct the required TF-IDF matrix by fitting and transforming the data
tfidf_matrix = tfidf.fit_transform(df2['soup'])
print(tfidf_matrix.shape)

#construct cosine similarity matrix
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)
print(cosine_sim.shape)

df2 = df2.reset_index()
indices = pd.Series(df2.index, index=df2['title']).drop_duplicates()

# create array with all movie titles
all_titles = [df2['title'][i] for i in range(len(df2['title']))]

def get_recommendations(title):
    # Get the index of the movie that matches the title
    idx = indices[title]
    # Get the pairwise similarity scores of all movies with that movie
    sim_scores = list(enumerate(cosine_sim[idx]))
    # Sort the movies based on the similarity scores
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    # Get the scores of the 10 most similar movies
    sim_scores = sim_scores[1:11]
    # print similarity scores
    print("\n movieId      score")
    for i in sim_scores:
        print(i)

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

    # return list of similar movies
    return_df = pd.DataFrame(columns=['Title','Homepage'])
    return_df['Title'] = df2['title'].iloc[movie_indices]
    return_df['Homepage'] = df2['homepage'].iloc[movie_indices]
    return_df['ReleaseDate'] = df2['release_date'].iloc[movie_indices]
```

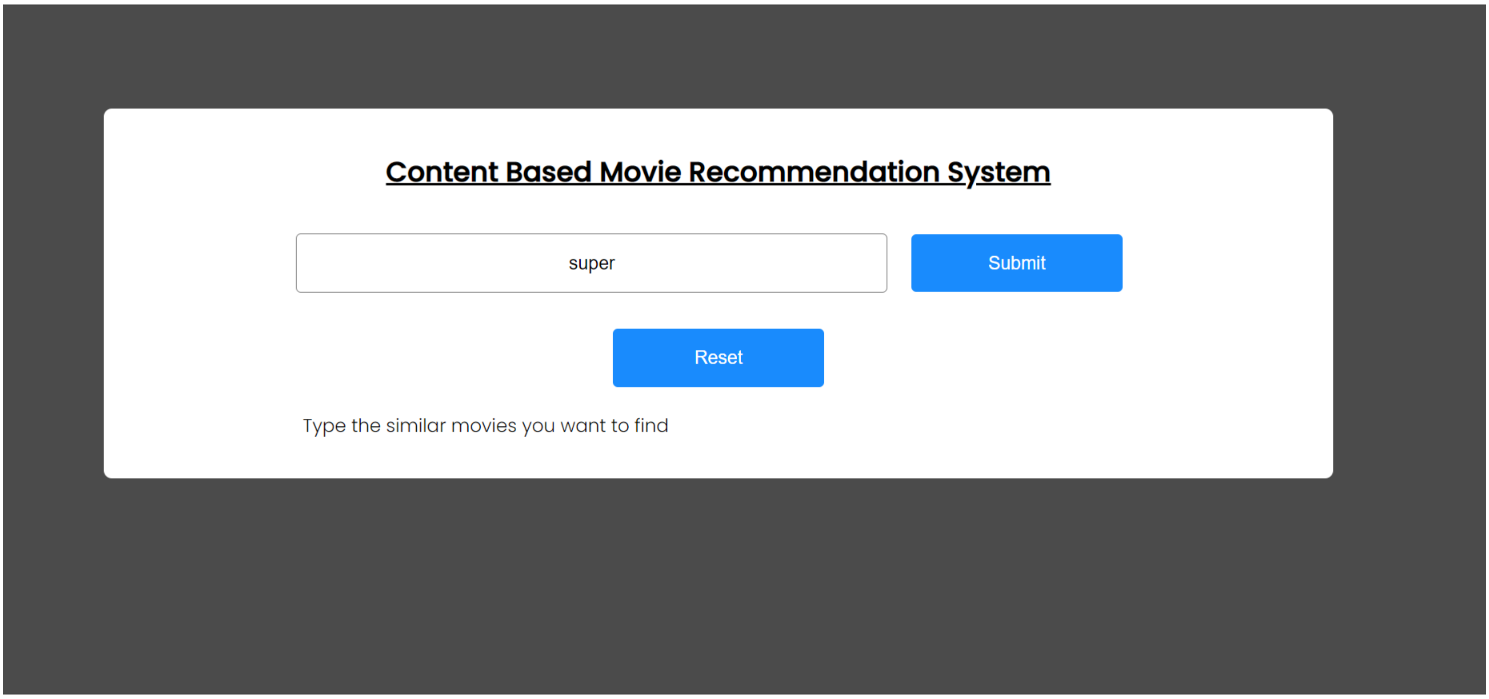


```
    return return_df

# Set up the main route
@app.route('/', methods=['GET', 'POST'])

def main():
    if flask.request.method == 'GET':
        return(flask.render_template('index.html'))

    if flask.request.method == 'POST':
        m_name = " ".join(flask.request.form['movie_name'].title().split())
#         check = difflib.get_close_matches(m_name,all_titles,cutout=0.50,n=1)
        if m_name not in all_titles:
            return(flask.render_template('notFound.html',name=m_name))
        else:
            result_final = get_recommendations(m_name)
            names = []
            homepage = []
            releaseDate = []
            for i in range(len(result_final)):
                names.append(result_final.iloc[i][0])
                releaseDate.append(result_final.iloc[i][2])
                if(len(str(result_final.iloc[i][1]))>3):
                    homepage.append(result_final.iloc[i][1])
                else:
                    homepage.append("#")


            return
flask.render_template('found.html',movie_names=names,movie_homepage=homepage,search_name=m_name
, movie_releaseDate=releaseDate)

if___name    == '__main__':
    app.run(host="127.0.0.1", port=8080, debug=True)
    #app.run()
```

CHAPTER 6

SCREENSHOTS AND RESULTS

6.1 Search Bar

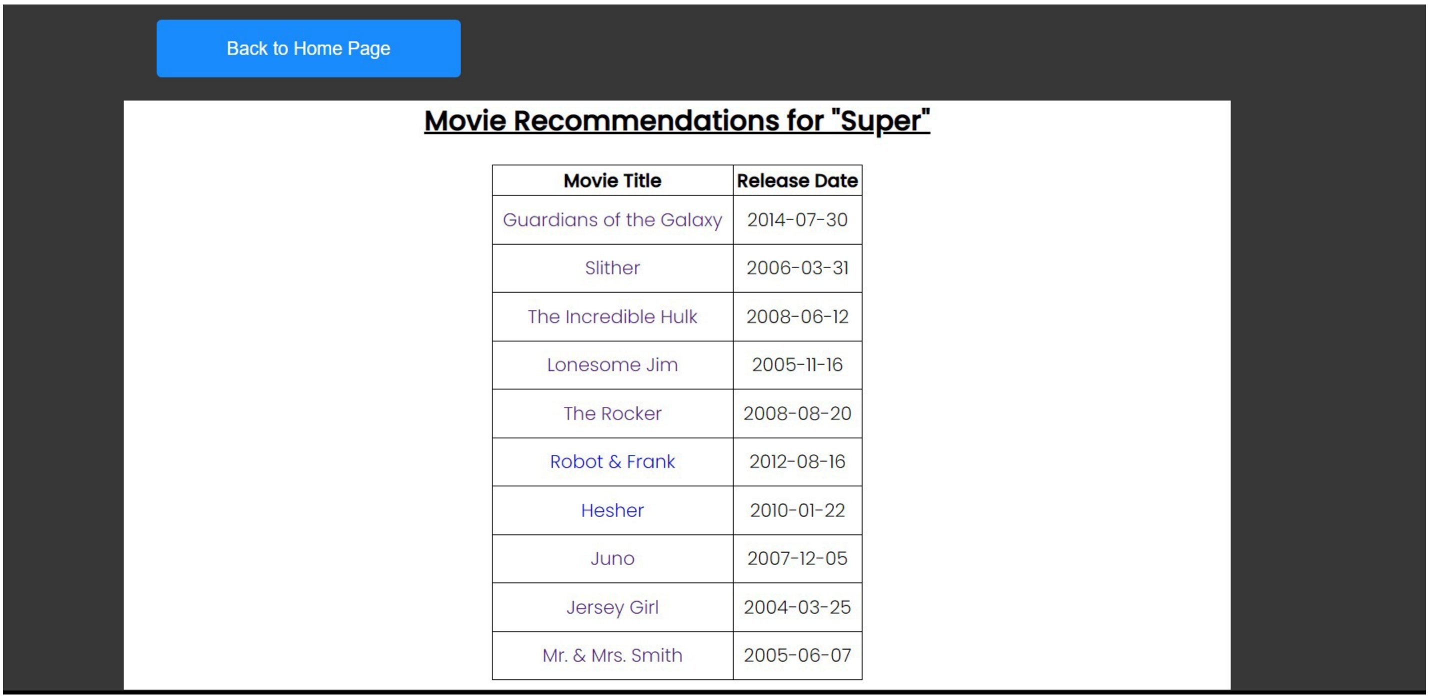


Content Based Movie Recommendation System

super

Type the similar movies you want to find

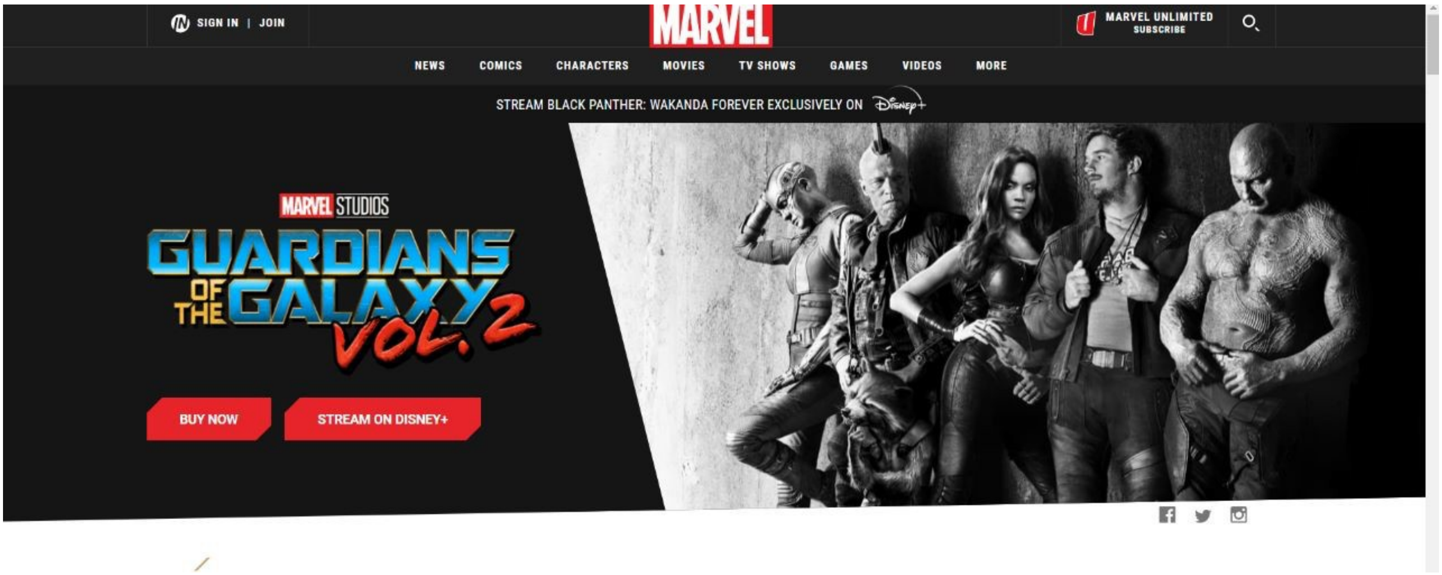
6.2 Recommendation



Movie Recommendations for "Super"

Movie Title	Release Date
Guardians of the Galaxy	2014-07-30
Slither	2006-03-31
The Incredible Hulk	2008-06-12
Lonesome Jim	2005-11-16
The Rocker	2008-08-20
Robot & Frank	2012-08-16
Hesher	2010-01-22
Juno	2007-12-05
Jersey Girl	2004-03-25
Mr. & Mrs. Smith	2005-06-07

6.3 Home Page


SIGN IN | JOIN
MARVEL
MARVEL UNLIMITED
SUBSCRIBE
NEWS
COMICS
CHARACTERS
MOVIES
TV SHOWS
GAMES
VIDEOS
MORE
STREAM BLACK PANTHER: WAKANDA FOREVER EXCLUSIVELY ON Disney+
MARVEL STUDIOS
GUARDIANS OF THE GALAXY VOL. 2
BUY NOW
STREAM ON DISNEY+

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENTS

A movie recommendation system is a powerful tool for suggesting movies to users based on their interests and preferences. In this mini project, we used a collaborative filtering approach to build a movie recommendation system that recommends movies to users based on their ratings and preferences. We used the MovieLens dataset to train our model, and evaluated its performance using various metrics such as RMSE and precision-recall curve. Overall, our model performed well, and we were able to recommend movies that were relevant and appealing to users.

There are several ways in which we can enhance our movie recommendation system in the future, including:

1. Implementing content-based filtering: In addition to collaborative filtering, we can also incorporate content-based filtering into our recommendation system. Content-based filtering uses features such as movie genre, director, and actors to recommend movies to users based on their preferences.
2. Incorporating deep learning techniques: We can also explore the use of deep learning techniques such as neural networks and convolutional neural networks to improve the accuracy of our recommendation system.
3. Integrating real-time data: Currently, our model is trained on historical data from the MovieLens dataset. However, we can also incorporate real-time data such as user behavior and ratings to make more personalized recommendations.
4. Incorporating diversity and novelty: To avoid recommending the same types of movies over and over again, we can also incorporate diversity and novelty into our recommendation system. This can be done by recommending movies that are similar but not identical to the ones the user has previously rated.

Overall, there are many ways in which we can enhance our movie recommendation system, and we look forward to exploring these possibilities in the future.

REFERENCES

- **https://techvidvan.com/tutorials/movie-recommendation-system-python-machine- learning/**
- **https://docs.streamlit.io/library/get-started**
- K.K.Aggarwal, “Software Engineering”
- Manoj Kumar, D.K. Yadav, Ankur Singh and Vijay Kr. Gupta (2015), “A Movie Recommender System: MOVREC”, Internatio**nal Journal of Computer Applications (0975 – 8887) Volume 124 – No.3.**
- **https://www.kaggle.com/**
- **sklearn.feature_extraction.text.CountVectorizer — scikit-learn 1.1.2 documentation**
- **Machine Learning for Recommender systems — Part 1 (algorithms, evaluation and cold start) | by Pavel Kordík | Recombee blog | Medium**
- **How to build Recommender System using Machine Learning? (enjoyalgorithms.com)**
- **https://www.themoviedb.org/**
- **https://www.geeksforgeeks.org/**
- **https://pythonprogramming.net/stemming-nltk-tutorial/**
- **https://www.machinelearningplus.com/nlp/cosine-similarity/**
- **https://www.youtube.com/**