A SOFTWARE ENGINEERING PROJECT

On

MOVIE RECOMMENDATION SYSTEM

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

KOLLA DURGA PRASAD (O200519)

MUPPANENI PRANAY SANKAR (O200341)

DEERGASI.BHANU (O200666)

KOKKILIGADDA HARIKA SRI (O200464)

CHEEMALAMARRI VENKATA ADITHYA (O200211)

Under the Guidance of

Mr. CHANDRASEKHAR N Assistant Professor

Dept. of Computer Science and Engineering

Department of Computer Science and engineering

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE AND TECHNOLOGIES

(Established through Government of A.P Act of 18 of 2008)

ANDHRA PRADESH, INDIA

(Catering to the Educational Needs of Gifted Rural Youth of Andhra Pradesh)

ONGOLE CAMPUS

Kurnool Road, Ongole, Prakasam(Dt.)

Andhra Pradesh-523225

www.rguktong.ac.in

CERTIFICATE

This is to certify that the project entitled "**MOVIE RECOMMENDATION SYSTEM"** being submitted by **K.Durga Prasad** bearing ID Number **O200519** and **M.Pranay Sankar** bearing ID Number **O200341** and **D.Bhanu** bearing ID Number **O200666** and **K.Harika Sri** bearing ID Number **O200464** and **CH.V.Adithya** bearing ID Number **O200211** in partial fulfillment of the requirements for the award of the degree of the Bachelor of Technology in Computer Science and Engineering in Dr. APJ Abdul Kalam, RGUKT-AP,IIIT Ongole is a record of bonafide work carried out by them under my guidance and supervision from July 2024 to November 2024.

The results presented in this project have been verified and found to be satisfactory. The results embodied in this project report have not been submitted to any other University for the award of any other degree or diploma.

Faculty-In-Charge **Head Of The Department**

Internal Examiner **External Examiner**

ACKNOWLEDGEMENT

It is our privilege to express a profound sense of respect, gratitude and indebtedness to our guide **Mr. NANDI MALLIKARJUNA Assistant Professor,** Dept. of Computer Science and Engineering, Dr. APJ Abdul Kalam, RGUKT-AP, IIIT Ongole, for her indefatigable inspiration, guidance, cogent discussion, constructive criticisms and encouragement throughout the dissertation work.

We express our sincere gratitude to **Mr. NANDI MALLIKARJUNA, Asst. Professor** & **Head of Department of Computer Science and Engineering,** Dr. APJ Abdul Kalam, RGUKT-AP, IIIT Ongole, for his suggestions, motivations and co-operation for the successful completion of the work.

We extend our sincere thanks to **Mr. MEESALA RUPAS KUMAR, Dean Academics,** Research and development, Dr. APJ Abdul Kalam, RGUKT-AP, IIIT Ongole, for his encouragement and constant help.

We extend our sincere thanks **DR.BHASKAR PATEL, Director,** Dr. APJ Abdul Kalam. RGUKT-AP,IIIT Ongole for his encouragement.

K Durga Prasad (O200519)

M Pranay Sankar (O200341)

D Bhanu (O200666)

K Harika Sri (O200464)

CH V Adithya (O200211)

Date:

DECLARATION

We hereby declare that the project work entitles **"MOVIE RECOMMENDATION SYSTEM"** submitted to the Rajiv **Gandhi University Of Knowledge Technologies Ongole** Campus in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology (B.Tech)** in Computer Science and Engineering is a record of an original work done by us under the guidance of **Ms. MALLIKARJUNA N Assistant Professor, Dept. Of CSE** and this project work have not been submitted to any university for the award of any other degree or diploma.

K Durga Prasad (O200519)

M Pranay Sankar (O200341)

D Bhanu (O200666)

K Harika Sri (O200464)

CH V Adithya (O200211)

ABSTRACT

This document introduces a movie recommendation system designed to enhance user experience by suggesting movies tailored to individual preferences. The system leverages recommendation algorithms, primarily collaborative filtering, content-based filtering, or a hybrid of both, to analyse user behaviour and movie attributes. Collaborative filtering recommends movies by identifying similar users or patterns in user ratings, while content-based filtering focuses on matching user preferences with specific movie attributes, such as genre, director, and actors.

The recommendation system aims to provide relevant and personalized movie suggestions, adapting to user feedback to refine recommendations over time. Key components of this system include data processing pipelines, which handle user interaction data and movie metadata, as well as scalable algorithms capable of performing in real-time to accommodate high user demand. This recommendation system seeks to improve user engagement by offering accurate, diverse, and timely recommendations, ultimately enhancing satisfaction and helping users discover new content that aligns with their tastes.

A Recommendation System is a filtering program whose primary goal is to predict the “rating” or “preference” of a user towards a domain-specific item. In our case, this domain-specific item is a movie. Hence the main focus of our recommendation system is to provide a total of ten movie recommendations to users who searched for a movie that they like. These results are based on similar traits/demographics of the movie that has been searched. Content based filtering is a technique that is used to recommend movies. Apart from providing recommendations the system also provides information about the searched movie. The additional details include the movie rating, its release date, cast, and genres. The system also provides additional information about the cast and to help the user save time.

CONTENT

1. INTRODUCTION

A recommendation system is a type of suggesting system which makes suggestions based on the user's liking. These systems can be applied to various data. These systems can retrieve and filter data based on user's preferences to give suggestions or recommendations in the upcoming period. To watch a movie the first step is to select a movie that matches the user's liking. Users often waste a lot of time selecting a movie to watch. Here comes the need for a recommendation system. It can Recommend popular movies based on their rating, but what makes the system useful is its ability to recommend movies based on user's liking and preferences. The purpose of this system is to search for content that would be interesting to an individual.

Our recommending system uses cosine similarity which is a type of content-based filtering method to recommend similar movies to the user. Additional information about the searched movie will also be provided. The additional information includes a Movie Poster of the movie. These functions of this system will prove to be very useful to the user and consequently save a lot of time, which the user can invest in actually watching the movie he/she likes.

1.1 MOTIVATION

A movie recommendation system provides a level of comfort 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, it is to filter and predict only those movies that a corresponding user is most likely to want to watch. This is the primary motivation in opting for movie recommendation system.

1.2 PROBLEM DEFINITION

- The reason behind this project is that we lose our quality time in search of movies, so we try to design a movie recommendation system that helps people in finding movies of their interest.
- People needs recommendations in order to do anything new. Which gives likeable recommendations based on the data of other users or choices of the user. System that seeks to predict the "rating" or "preference" a user would give to an item.

1.3. OBJECTIVE OF THE PROJECT

- The goal of the movie recommendation system is to allow the users to reduce their searching time and give them their likely shows.
- The recommendation system analyzes the past preferences of the user concerned, and then it uses this information to try to find similar movies. This information is available in the dataset (e.g. Genre, Duration year of movie, Cast, Crew etc.). After that, the system provides movie recommendations for the user.

2. LITERATURE SURVEY

The first recommendation system was established in 1990 and was based on the e-commerce recommender which is known as the tapestry. The term recommender system was coined by a computer-based librarian named as Grundy in 1979. After that, there was the invention of various recommendation systems using various technologies. Today, there is a large number of recommendation systems available with different technologies and it is available in different fields also.

Nisha Sharma and Mala Dutta [1] proposed an overview survey on the recommendation system which contains all details about the recommendation system.

Gaurav Srivastav [2] proposed the recommendation system using the concept of cosine similarity and the KNN algorithm. Here, we studied cosine similarity.

Kumar et al. [3] proposed MOVREC, a movie recommendation system based on collaborative filtering approaches. Collaborative filtering takes the data from all the users and based on that generates recommendations.

Munoz-Organero, Mario [4], in this paper he proposed a Collaborative Recommender System Based on SpaceTime Similarities.

E. Nakhli, H. Moradi, and M. A. Sadeghi [5], in this paper, proposed the recommendation system using the percentage view criteria which helps to get the suggestions.

G. Wang [6] proposed a brief survey on different types of recommender systems on their use of it.

Peng, Xiao, Shao Liangshan, and Li Xiuran [7], in this paper the electronic commerce recommendation system has a similar look and makes a specialty of the collaborative filtering algorithm in the utility of a personalized film recommendation system.

Sharma and Maan [8] in their paper analyzed various techniques used for recommendations, collaborative, hybrid, and content-based recommendations. Also, it describes the pros and cons of these approaches.

Shreya Agrawal and Pooja Jain [9] in their paper give information about the various approaches to making are commander system.

Nagamanjula R and A. Pethalakshmi [10] in their paper proposed a novel scheme for the use of user similarity and opinion mining for the recommender system.

M. Jahrer, A. Toscher, and R. Legenstein [11], in this paper proposed the recommender system for e-commerce facilities which will be used for recommending a product.

Bhavya Ghai, Joydip Dhar, and Anupam Shukla [12], in this paper they examine multi-level ensemble learning with regard to recommender systems and critique traditional ensemble learning. They place greater emphasis on developing recommender systems employing stack generalization in this.

3. ANALYSIS

3.1 EXISTED SYSTEM

As of my last knowledge update in January 2024, several movie recommendation systems were in existence, employed by popular streaming platforms, websites, and applications. Here are a few notable examples:

- **Netflix:** Netflix uses a sophisticated recommendation system that combines various algorithms, including collaborative filtering, content-based filtering, and deep learning. It analyzes user viewing history, preferences, and interactions to suggest movies and TV shows.
- **Rotten Tomatoes:** Rotten Tomatoes suggests movies based on user ratings, critic reviews, and personal preferences. It provides a "Tomatometer" score and audience score to help users make informed decisions.
- **Amazon Prime Video:** Amazon utilizes a recommendation engine that considers both user behavior and item-based collaborative filtering. It suggests movies and TV shows based on a user's past purchases, ratings, and browsing history.
- **IMDb (Internet Movie Database):** IMDb provides movie recommendations based on user ratings, reviews, and watch lists. It also offers personalized suggestions by considering individual preferences and browsing history.

3.2 PROPOSED SYSTEM

In this system we are using Cosine Similarity Algorithm.

- **Data Collection:** Gather data from various sources like TMDB .This dataset should contain movie attributes such as genre, crew, cast, movie type, release year, etc.
- **User Interface:** Design an interface (web or app) where users can input preferences, view recommendations.
- **Scalability and Optimization:** Ensure the system can handle large datasets efficiently and optimize algorithms for faster recommendation generation.

- **Content-Based Filtering:** Analyze movie metadata (genre, crew, cast, release year) using natural language processing. Recommend movies based on user preferences and content features.

3.3. SOFTWARE REQUIREMENT SPECIFICATION

This was the phase that involved the actual realization of the system. It is at this stage that python language and various python modules are used Pickle, Sklearn, Streamlit, Requests, Pandas and NumPy. These technologies are further defined as below.

- **Pickle:** The pickle library in Python is used for serializing and deserializing Python objects, allowing you to save complex data structures like machine learning models, DataFrames, or dictionaries to a file and load them later. In pandas, pickle is often used to save DataFrames efficiently, preserving data and structure without needing to reload or preprocess data each time.
- **Scikit-learn (sklearn):** Scikit-learn is a popular machine learning library in Python. It provides simple and efficient tools for data mining and data analysis. Scikit-learn includes various algorithms for classification, regression, clustering, dimensionality reduction, and more.
- **Streamlit:** Streamlit is a tool that lets you easily turn your Python code into an interactive web app. It's great for data projects because you can add sliders, buttons, and charts quickly, making your results easy to explore and share.
- **Requests:** The requests library in Python simplifies making HTTP requests, allowing users to easily send and receive data from web APIs using methods like GET and POST. Its straightforward syntax makes it easy to manage parameters, headers, and authentication, making it a popular choice for developers working with web services.
- **Pandas:** Pandas is a powerful data manipulation and analysis library for Python. It offers data structures and operations for manipulating numerical tables and time series. Pandas is commonly used for data cleaning, preparation, and analysis in data science workflows.
- **NumPy:** NumPy is a fundamental package for scientific computing in Python. It provides support for multidimensional arrays, matrices, and high-level mathematical functions to

operate on these arrays. NumPy is essential for numerical computations in various fields like physics, engineering, and data analysis.

HARDWARE REQUIREMENTS

- A PC with Windows/Linux OS
- Processor with 1.7-2.4gHz speed
- 2gb Graphics card
- Minimum of 8gb RAM

3.3.1 PURPOSE

Using this type of recommender system, if a user watches one movie, similar movies are recommended. For example, if a user watches a comedy movie starring Adam Sandler, the system will recommend them movies in the same genre or starring the same actor, or both.

3.3.2 SCOPE

The scope of the movie recommendation system is to allow the users to reduce their searching time and give them their likely shows. The movie recommendation system dataset is used in this strategy to analyze the history of a user's preferences & suggest movies that other users with similar interests enjoy.

3.3.3 OVERALL DESCRIPTION

A recommendation system is a type of suggesting system which makes suggestions based on the user's liking. These systems can be applied to various data. These systems can retrieve and filter data based on user's preferences to give suggestions or recommendations. Our recommending system uses cosine similarity which is a type of content-based filtering method to recommend similar movies to the user. Additional information about the searched movie will also be provided. The additional information includes a Movie Poster, an Overview of the movie, a Rating of the movie, Genres, the Run time of the movie.

4. DESIGN

4.1 UML DIAGRAMS

A UML Diagram is based on **UML(Unified Modeling Language)**with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain ,or document information about the system. The UML diagrams are divided into Structural and Behavioral UML Diagrams.

STRUCTURAL UML DIAGRAMS:

Structural diagrams depict a static view of a structure of a system. It is widely used in the Documentation of software architecture. The Structural UML Diagrams involves 6 diagrams

They are:

- Class Diagram
- Object Diagram
- Component Diagram
- Composite Diagram
- Package Diagram
- Deployment Diagram
- Profile Diagram

BEHAVIOURAL UML DIAGRAMS:

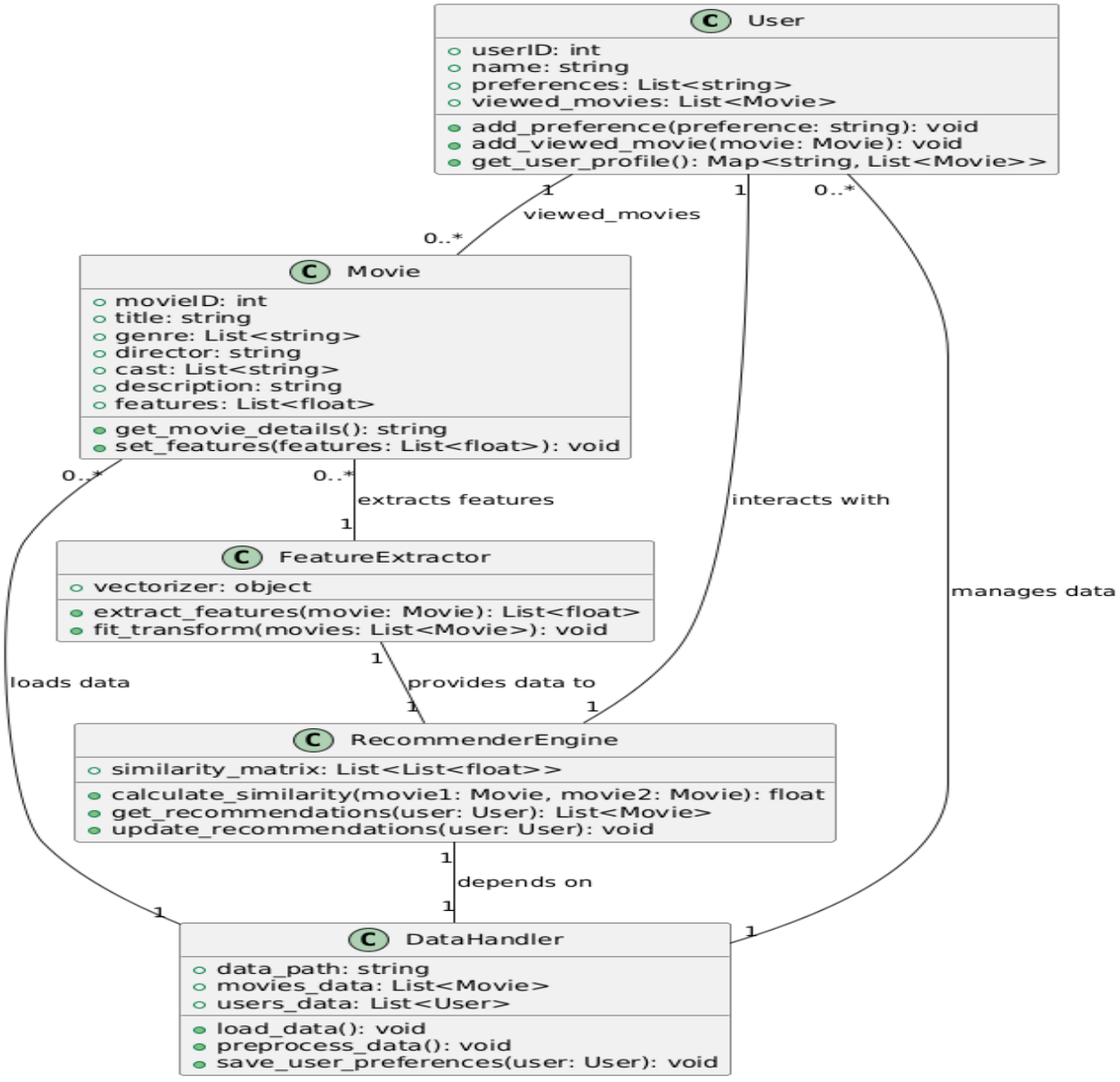
Behavioral diagrams portray a dynamic view of a system or the behavior of a system, Which describes the functioning the system. It involves 7 diagrams.

They are:

- Use Case Diagram
- State Machine Diagram
- Timing Diagram
- Communication Diagram
- Sequential Diagram
- Interaction diagram
- Activity Diagram

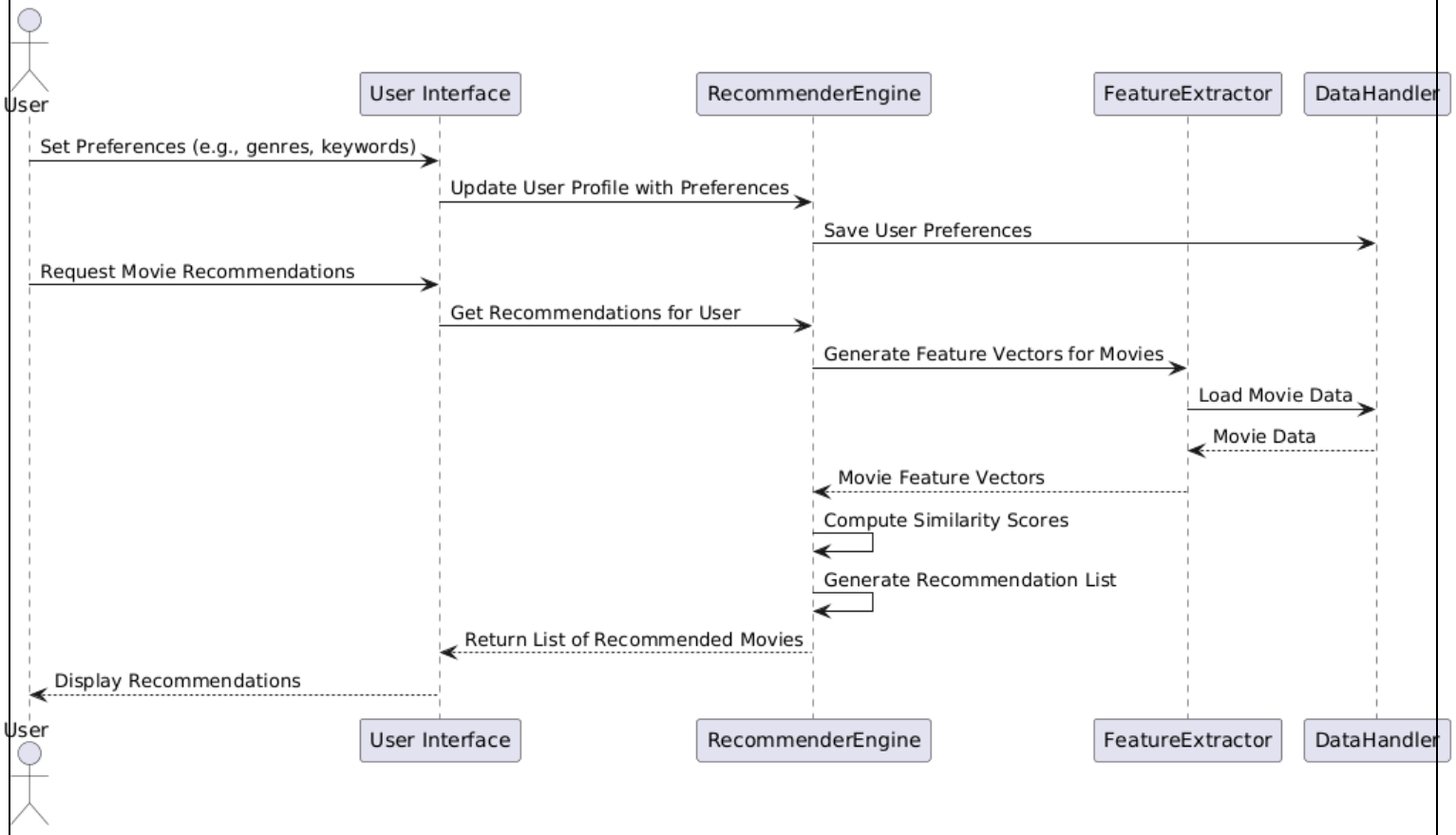
CLASS DIAGRAM

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the structure of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modelling.



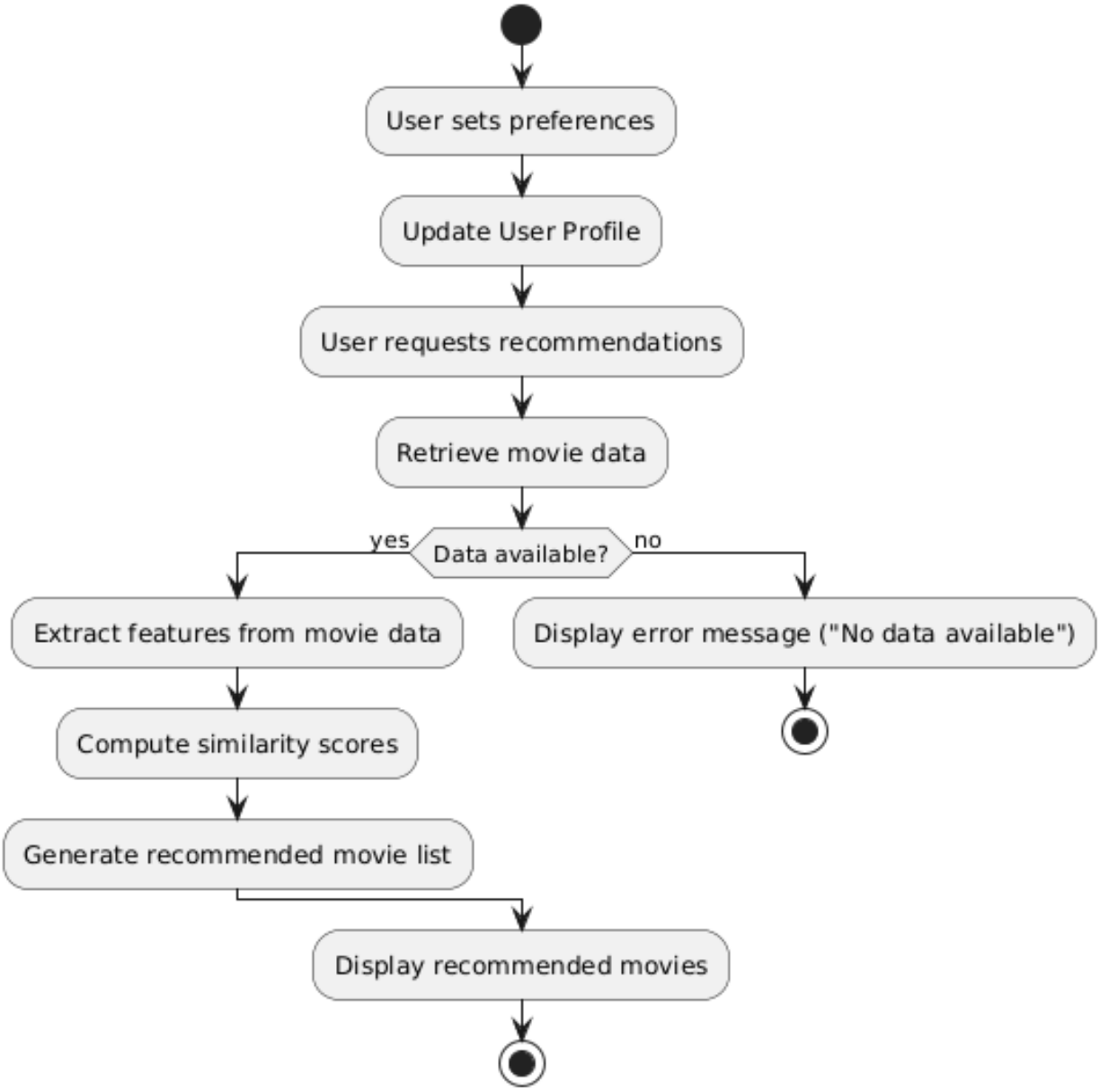
SEQUENTIAL DIAGRAM

A sequence diagram consists of a group of objects that are represented by lifelines and the messages that they exchange over time during the interaction. A sequence diagram shows the sequence of messages passed between objects. Sequence diagrams can also show the control structures between objects.



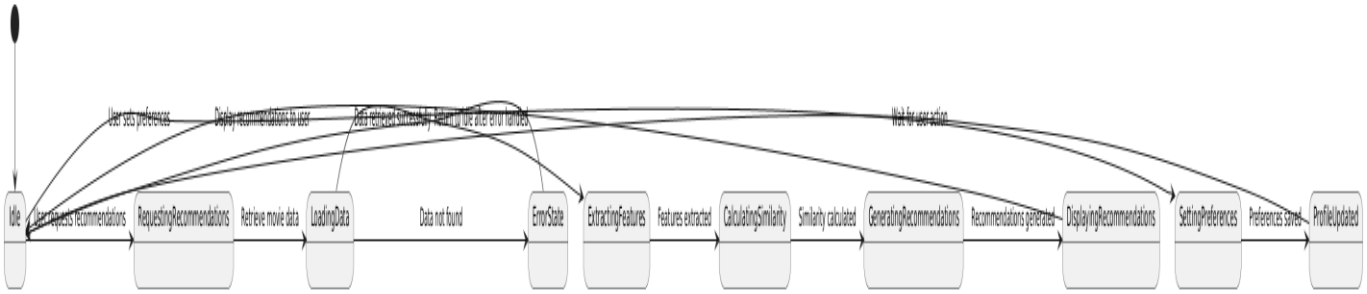
ACTIVITY DIAGRAM

An activity diagram visually presents a series of actions or flow of control in a system similar to a flowchart or a data flow diagram. Activity diagrams are often used in business process modeling. They can also describe the steps in a use case diagram. Activities modeled can be sequential and concurrent.



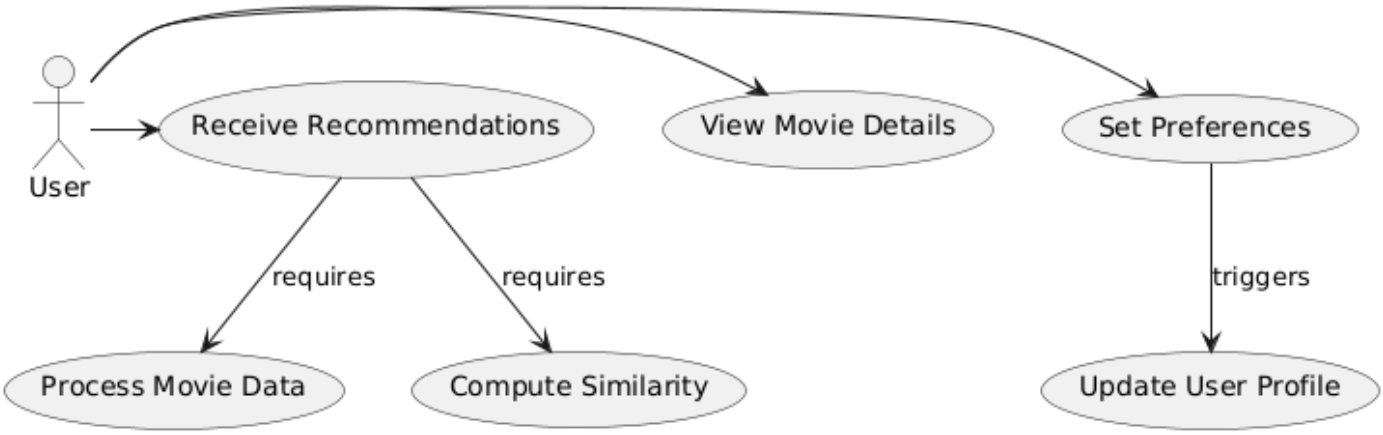
STATE MACHINE DIAGRAM

A state diagram, also known as a state machine diagram or statechart diagram, is an illustration ofthe states an object can attain as well as the transitions between those states in the Unified Modeling Language (UML).



USECASE DIAGRAM

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors. The use cases and actors in usecase diagrams describe what the system does and how the actors use it, but not how the system operates internall.



5. IMPLEMENTATION

5.1. MODULES:

- Manage Dataset
- Update Movies
- Recommended System

5.2. MODULE DESCRIPTION:

➔ **Manage Dataset:** Managing a dataset for a movie recommendation system involves several steps to ensure the data is organized, clean, and ready for analysis. Here's a general outline of what's typically involved:

1. Data Collection:

Source Data: Gather information from various sources like IMDb, TMDb, user ratings, reviews, etc.

APIs or Datasets: Utilize APIs or existing datasets available for movies.

2. Data Cleaning:

Handling Missing Values: Check for missing data (null values) and decide how to handle them (remove, impute, etc.).

Data Standardization: Ensure consistency in data formats (e.g., dates, genres).

Duplicates: Remove duplicate entries to maintain data integrity.

3. Feature Selection:

Feature Engineering: Create new features that might be useful for recommendation algorithms (e.g., user preferences, movie genres).

Identify Relevant Features: Choose the most relevant features for building the Recommendation model.

➔ **Update Movies:** Updating a movie recommendation dataset is crucial to ensure that the system remains relevant and up-to-date with new movies, user preferences, and changing trends.

1. Data Integration and Merging:

Merge New Data: Incorporate new information with the existing dataset, ensuring consistency and avoiding duplicates.

Update Existing Records: Update existing movie information if there are changes or corrections in the data source.

2. Data Quality Checks:

Validation: Verify the integrity and quality of the updated data. Check for inconsistencies, missing values, or anomalies.

Duplicate Handling: Ensure that new entries don't duplicate existing records.

3. Monitoring and Evaluation:

Monitoring Performance Metrics: Continuously monitor the performance of the recommendation system with the updated dataset to ensure it meets the desired standards.

➔ **Recommendation Generation:**

Feature-Based Recommendations: Recommends movies based on their attributes such as genre, director, actors, plot keywords, and other metadata. It suggests movies similar to those a user has previously enjoyed.

5.3 INTRODUCTION OF TECHNOLOGIES USED:

Introduction to Python programming Language:

Python (language): Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

This was the phase that involved the actual realization of the system. It is at this stage that python language and various python modules are used say- **Pickle, Sklearn, Streamlit, Requests, Pandas and NumPy.**

5.4 SAMPLE CODE

Home page:

```
import streamlit as st
import pandas as pd
import pickle
import requests

def fetch_poster(movie_id):

response=requests.get('https://api.themoviedb.org/3/movie/{}?api_key=8265bd1679663a7ea12ac168da84d2e8&language=en-US'.format(movie_id))
    data=response.json()
    return "https://image.tmdb.org/t/p/w500/"+data["poster_path"]


def recommend(movie):
    movie_index = movies[movies["title"] == movie].index[0]
    distances = similarity[movie_index]
    movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[0:6]
    recommend_movies=[]
    recommend_movies_posters=[]
    for i in movies_list:
        movie_id=movies.iloc[i[0]].movie_id


        recommend_movies.append(movies.iloc[i[0]].title)
        # fetch poster from API
        recommend_movies_posters.append(fetch_poster(movie_id))
    return recommend_movies,recommend_movies_posters
```

```
movies_dict=pickle.load(open("movies_dict.pkl","rb"))
movies=pd.DataFrame(movies_dict)
similarity=pickle.load(open("similarity.pkl","rb"))



st.title("MOVIE RECOMMENDATION SYSTEM")
selected_movie_name = st.selectbox(
    "Which movie do you want to recommend?",
    movies["title"].values
)

if st.button("Recommend"):
    names,posters=recommend(selected_movie_name)

    col1, col2, col3,col4,col5,col6 = st.columns(6)
    with col1:
        st.text (names[0])
        st.image (posters[0])

    with col2:
        st.text (names[1])
        st.image (posters[1])

    with col3:
        st.text (names[2])
        st.image (posters[2])

    with col4:
```

```
    st.text (names[3])
    st.image (posters[3])

with col5:
    st.text (names[4])
    st.image (posters[4])

with col6:
    st.text (names[5])
    st.image (posters[5])
```

Recommendation page:

```
import numpy as np
import pandas as pd
credits=pd.read_csv("tmdb_5000_credits.csv")
movies=pd.read_csv("tmdb_5000_movies.csv")
movies.head(1)
credits.head(1)
movies=movies.merge(credits,on="title")
movies.head()
movies=movies[["movie_id","title","overview","genres","keywords","cast","crew"]]
movies.info()
movies.head()
movies.isnull().sum()
movies.dropna(inplace=True)
movies.duplicated().sum()
movies.iloc[0].genres
def convert(obj):
    L=[]
    for i in ast.literal_eval(obj):
        L.append(i["name"])
    return L
movies["genres"]=movies["genres"].apply(convert)
movies.head()
movies["keywords"]=movies["keywords"].apply(convert)
movies.head()
def convert3(obj):
    L=[]
    count=0
    for i in ast.literal_eval(obj):
        if(count!=3):
```

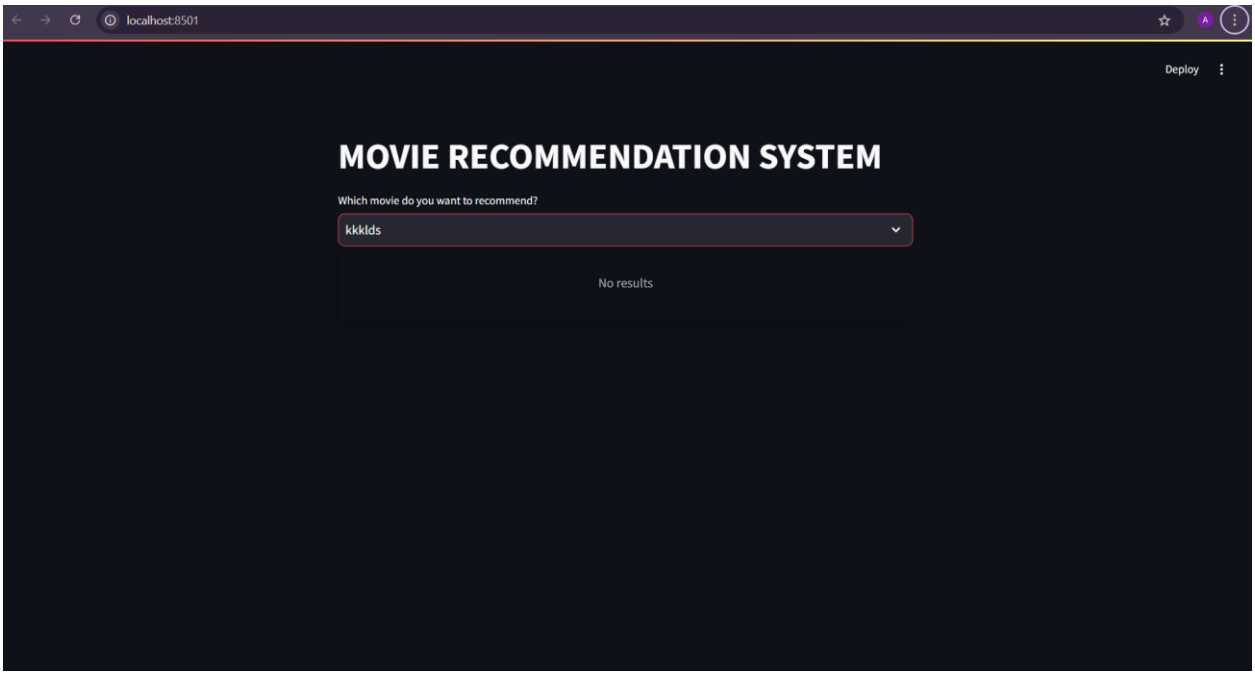
```
            L.append(i["name"])
            count+=1
        else:
            break
    return L
movies["cast"]=movies["cast"].apply(convert3)
movies.head()
def fetch_director(obj):
    L=[]
    for i in ast.literal_eval(obj):
        if(i["job"]=="Director"):
            L.append(i["name"])
            break

    return L
movies["crew"]=movies["crew"].apply(fetch_director)
movies.head()
movies["overview"][0]
movies["overview"]=movies["overview"].apply(lambda x: x.split())
movies.head()
movies["genres"]=movies["genres"].apply(lambda x:[i.replace(" ","") for i in x])
movies["keywords"]=movies["keywords"].apply(lambda x:[i.replace(" ","") for i in x])
movies["cast"]=movies["cast"].apply(lambda x:[i.replace(" ","") for i in x])
movies["crew"]=movies["crew"].apply(lambda x:[i.replace(" ","") for i in x])
movies["tags"]=movies["overview"]+movies["genres"]+movies["keywords"]+movies["cast"]+mo
vies["crew"]
new_df=movies[["movie_id","title","tags"]]
new_df["tags"]=new_df["tags"].apply(lambda x:" ".join(x)) #string converstion
new_df["tags"]=new_df["tags"].apply(lambda x: x.lower())
#vectorization
```

```
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=5000,stop_words="english")
vectors=cv.fit_transform(new_df["tags"]).toarray()
vectors.shape
len(cv.get_feature_names())
vectors[0]
import nltk
from nltk.stem.porter import PorterStemmer
ps=PorterStemmer()
def stem(text):
    y=[]
    for i in text.split():
        y.append(ps.stem(i))
    return " ".join(y)
new_df["tags"]=new_df["tags"].apply(stem)
#cosine similarity
from sklearn.metrics.pairwise import cosine_similarity
cosine_similarity(vectors)
similarity=cosine_similarity(vectors)
def recommend(movie):
    movie_index=new_df[new_df["title"]==movie].index[0]
    distances=similarity[movie_index]
    movies_list=sorted(list(enumerate(distances)),reverse=True,key=lambda x: x[1])[1:6]
    for i in movies_list:
        print(new_df.iloc[i[0]].title)
sorted(list(enumerate(similarity[0])),reverse=True,key=lambda x: x[1])[1:6]
recommend("Batman Begins")
import pickle
pickle.dump(new_df.to_dict,open("movies_dict.pkl","wb"))
pickle.dump(similarity,open("similarity.pkl","wb"))
```

6. TESTING

6.1 BLACK BOX TESTING

Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time, usability issues and reliability issues. Black box testing is a powerful testing technique because it exercises a system end-to-end. Just like end-users “don’t care” how a system is coded or architected, and expect to receive an appropriate response to their requests, a tester can simulate user activity and see if the system delivers on its promises. Along the way, a black box test evaluates all relevant subsystems, including UI/UX, web server or application server, database, dependencies, and integrated systems.



6.2 WHITE BOX TESTING

This document outlines the steps and areas covered in white-box testing to ensure our movie recommendation system works accurately and efficiently.

1. Algorithm Testing:

- **Collaborative Filtering**: Check if the system correctly identifies similar users or movies based on ratings.
- **Content-Based Filtering**: Verify that the system recommends movies that match user preferences using movie features (like genre, actors).
- **Hybrid Models**: If we use a combination, make sure both types of recommendations are combined correctly.

2. Code Coverage

- Aim to test every major part of the recommendation logic.
- Use tools to confirm that we have tested most of the code, especially the recommendation generation parts.

3. Data Flow Testing

- Ensure user data (like ratings or preferences) flows correctly through the system.
- Check that data is transformed consistently (e.g., ratings are normalized).

4. Edge Cases and Boundaries

- Test for users with no history to ensure they still get recommendations.
- Handle new movies with no ratings.
- Check if the system can manage both highly rated and low-rated movies properly.

5. Loops and Conditionals

- Test loops that process user or movie data to handle a large number of records.
- Check all conditional paths (if-statements) to confirm the system's behavior under different scenarios.

6. Performance

- Test if the recommendation logic is efficient enough to generate recommendations quickly.
- Identify any slow parts of the code and optimize if needed.

7. Feedback and Model Updates

- If user interactions (like ratings) update recommendations, test that these changes are correctly applied.
- Ensure the system adapts to user feedback in future recommendations.

8. Database and Caching

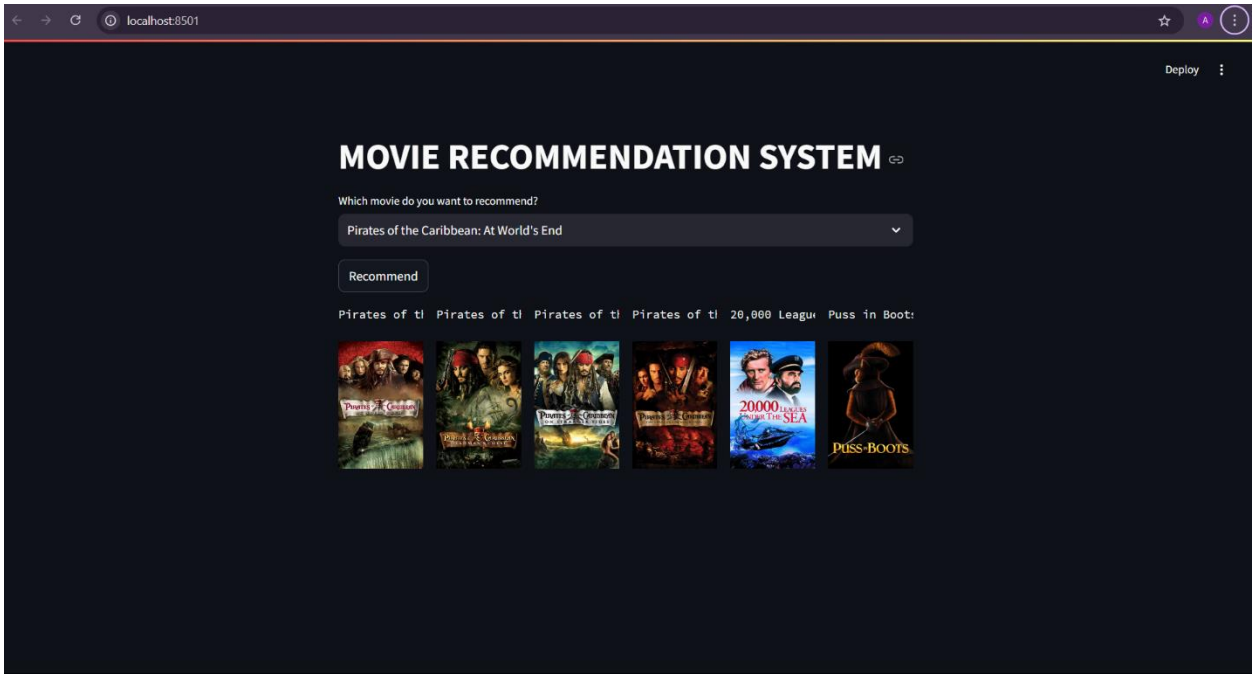
- Ensure data is retrieved and stored correctly.
- Check if caching works effectively to speed up frequent recommendations.

9. Explainability

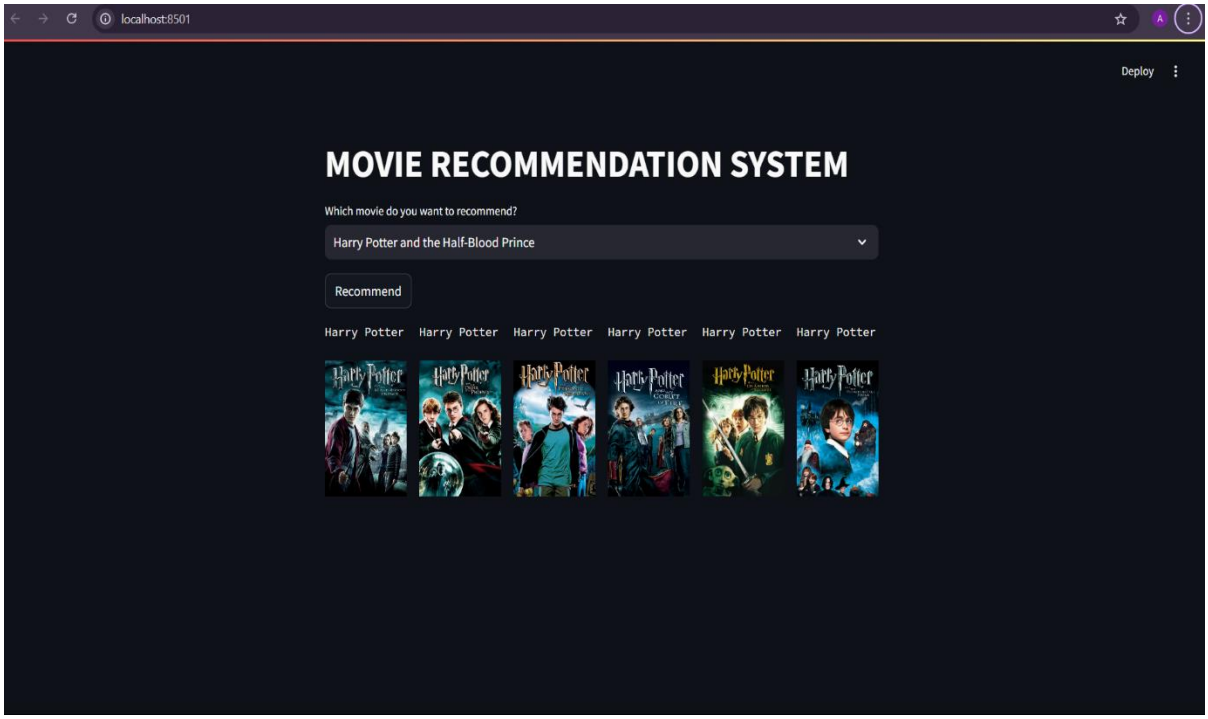
- If the system provides explanations for recommendations, make sure they are accurate and match the recommendation logic.

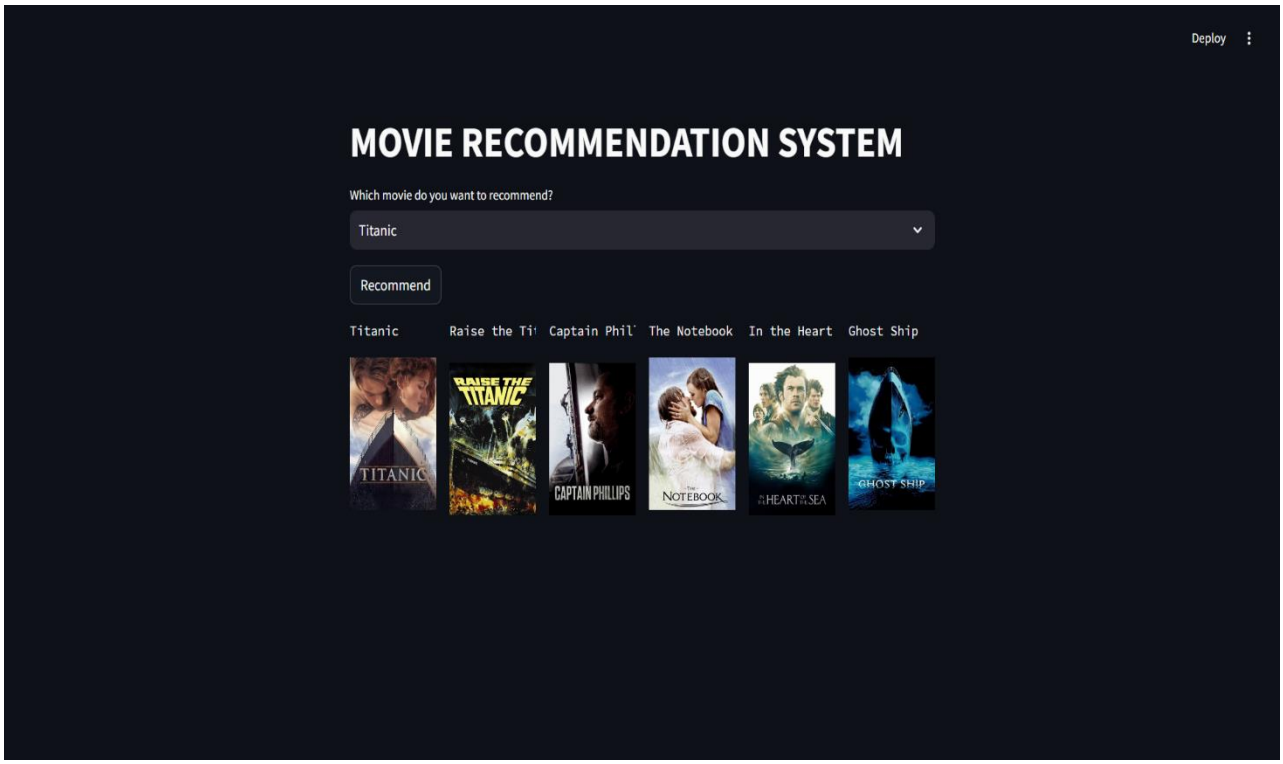
10. Security and Privacy

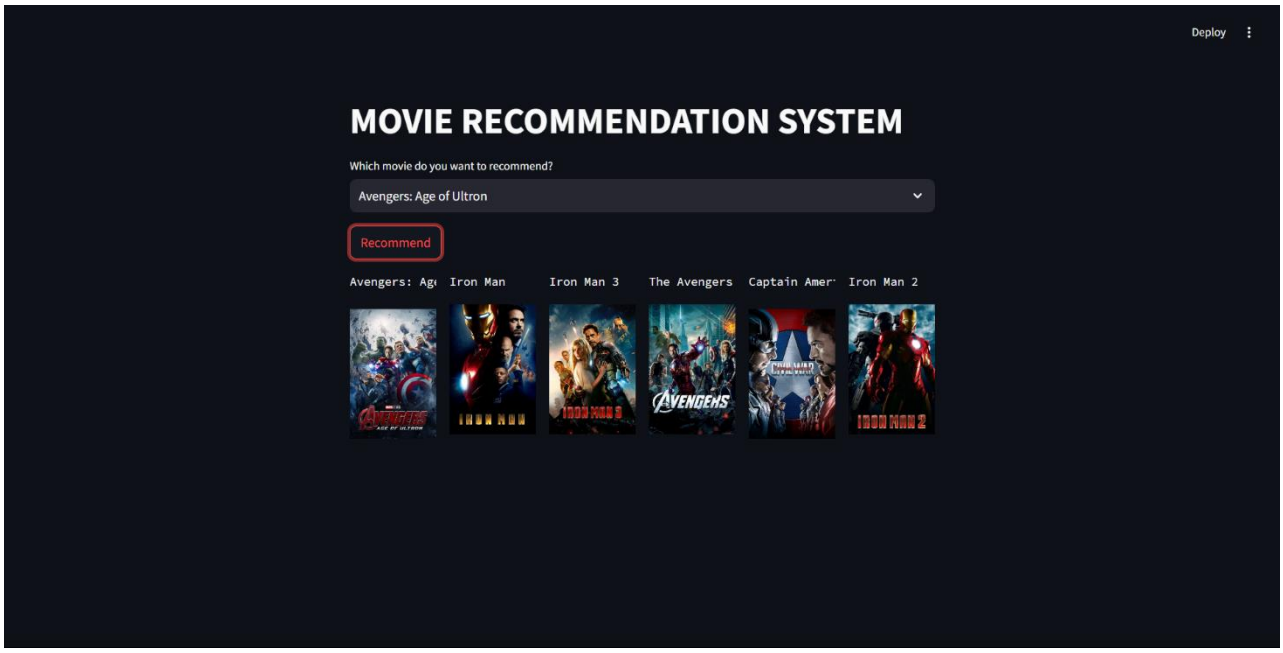
- Test that user data is processed securely and that no sensitive information is exposed.


localhost:8501
Deploy
MOVIE RECOMMENDATION SYSTEM
Which movie do you want to recommend?
Pirates of the Caribbean: At World's End
Recommend
Pirates of tl
Pirates of tl
Pirates of tl
Pirates of tl
20,000 Leagu
Puss in Boots
20000 LEAGUES UNDER THE SEA
PUSS IN BOOTS

7. RESULT


localhost:8501
Deploy
MOVIE RECOMMENDATION SYSTEM
Which movie do you want to recommend?
Harry Potter and the Half-Blood Prince
Recommend
Harry Potter
Harry Potter
Harry Potter
Harry Potter
Harry Potter
Harry Potter


Deploy
MOVIE RECOMMENDATION SYSTEM
Which movie do you want to recommend?
Titanic
Recommend
Titanic
Raise the Ti
Captain Phil
The Notebook
In the Heart
Ghost Ship


Deploy
MOVIE RECOMMENDATION SYSTEM
Which movie do you want to recommend?
Avengers: Age of Ultron
Recommend
Avengers: Ag
Iron Man
Iron Man 3
The Avengers
Captain Amer
Iron Man 2

8. CONCLUSION

The utilization of cosine similarity in movie recommendation systems represents a sophisticated and effective approach to connecting viewers with films that resonate with their preferences. By mathematically measuring the similarity between vectors representing user tastes and movie features, this method empowers systems to generate personalized recommendations. Its efficiency lies in its ability to discern nuanced similarities, ensuring that suggested movies align closely with individual preferences, even amidst a vast and diverse cinematic landscape.

This technique's beauty is in its adaptability and scalability—able to accommodate large datasets and diverse user preferences while continuously refining its suggestions. As technology evolves and datasets expand, cosine similarity remains a robust tool, continually enhancing the movie-watching experience by facilitating tailored suggestions that enrich and diversify each user's cinematic journey. Its contribution to the seamless alignment of viewer tastes with an extensive array of films underscores its pivotal role in the evolution of movie recommendation systems.

As a result, the live working of the system generates accurate and personalized recommendations along with the analysis of sentiments for the end users. It is also concluded that Cosine Similarity provides better and efficient results for a recommender system.

9. FUTURE ENHANCEMENT

- **Dynamic User Profiles**: Implement dynamic user profiles that adapt over time based on recent interactions and evolving preferences. This ensures that the recommendations remain up-to-date and reflective of users' changing tastes.
- **Content Freshness Considerations**: Incorporate a temporal aspect to the recommendation system, considering the freshness of content. This can involve promoting recently released movies or introducing mechanisms to prevent recommendations of outdated or less relevant content.
- **Context-Aware Recommendations**: Integrate contextual information, such as the time of day, location, or user activity, to provide recommendations that align with the user's current context. For example, suggesting light comedies on weekends or family-friendly movies for users with children.
- **Interactive Interfaces:** Implementing more interactive interfaces where users can provide richer feedback, enabling them to better articulate their preferences beyond simple ratings

10. BIBLIOGRAPHY

1. Adomavicius, G., & Tuzhilin, A. (2005). Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, 17(6). This paper provides a comprehensive overview of recommender system methodologies, including collaborative and content-based filtering, and discusses potential improvements in the field.

2. Ricci, F., Rokach, L., Shapira, B., & Kantor, P. B. (2011). Recommender Systems Handbook. Springer. This book serves as a detailed guide covering various recommendation techniques, implementation methods, and applications, with a section dedicated to movie recommendations.

3. Resnick, P., & Varian, H. R. (1997). Recommender Systems. Communications of the ACM, 40(3). A foundational article introducing recommender systems and discussing early applications in movies and other media.

4. Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. Computer, 42(8). This influential paper explains matrix factorization, a technique widely used in collaborative filtering to improve recommendation accuracy, especially in movie recommendation applications.

5. Breese, J. S., Heckerman, D., & Kadie, C. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence (UAI). A foundational study on collaborative filtering techniques and algorithms, presenting evaluations that are still relevant for building movie recommendation systems.

6. Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction, 12(4). This paper examines hybrid recommendation systems, which combine multiple recommendation approaches (e.g., collaborative and content-based) to enhance performance, a common practice in movie recommendations.

7. Linden, G., Smith, B., & York, J. (2003). Amazon.com Recommendations: Item-to-Item Collaborative Filtering. IEEE Internet Computing, 7(1). Though focused on retail, this paper discusses item-to-item collaborative filtering, a technique adaptable to movies and known for its efficiency in real-time recommendation.

8. Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). E-Commerce Recommendation Applications. Data Mining and Knowledge Discovery, 5(1-2). This article discusses the application of recommendation systems in various domains, including movies, and explores the techniques used in commercial recommendation systems.

9. Aggarwal, C. C. (2016). Recommender Systems: The Textbook. Springer. This textbook provides an in-depth look into different types of recommendation systems, including detailed sections on collaborative filtering, content-based methods, and hybrid approaches.

10. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001). Item-Based Collaborative Filtering Recommendation Algorithms. Proceedings of the 10th International Conference on World Wide Web (WWW), 285-295. A widely cited paper on item-based collaborative filtering algorithms, which are efficient and commonly applied to large-scale movie recommendation systems.

11. Ricci, F. (2011). Context-Aware Movie Recommendation. In Recommender Systems Handbook (pp. 421-455). Springer. This book chapter explores the integration of context, such as location or mood, into movie recommendation systems, enhancing the relevance of recommendations.

12. Netflix Prize Dataset and Competition (2006-2009). Netflix's open competition and dataset spurred significant advancements in recommendation algorithms, including improvements in matrix factorization and collaborative filtering techniques for movie recommendations.