GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (Autonomous)
Approved by AICTE, New Delhi and Affiliated to JNTU-Kakinada
Re-accredited by NAAC with "A" Grade with a CGPA of 3.47/4.00
Madhurawada, Visakhapatnam - 530 048.

MOVIE RECOMMENDER SYSTEM

An Industry Oriented Mini Project submitted in partial fulfilment of requirements for the award of degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

BY

POPURI TRIVENI	**(Reg No:19131A05H9)**
METTU AMARNADH REDDY	**(Reg No:19131A05D2)**
ROKALLA SRUTI	**(Reg No:19131A05K6)**
MITTAKOLA HARSHA VARDHINI	**(Reg No:20135A0515)**

COLLEGE OF ENGINEERING
(AUTONOMOUS)

Under the esteemed guidance of

Mrs. V. TULASI

(Assistant Professor)

Department of Computer Science and Engineering

GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (AUTONOMOUS)

(Affiliated to JNTU-K, Kakinada)

VISAKHAPATNAM

2021 – 2022

Gayatri Vidya Parishad College of Engineering (Autonomous)

Visakhapatnam

CERTIFICATE

This report on
"Movie Recommender System"
is a bonafide record of the mini project work submitted

By

POPURI TRIVENI	**(Reg No:19131A05H9)**
METTU AMARNADH REDDY	**(Reg No:19131A05D2)**
ROKALLA SRUTI	**(Reg No:19131A05K6)**
MITTAKOLA HARSHA VARDHINI	**(Reg No:20135A0515)**

in their V semester in partial fulfilment of the requirements for the Award of Degree of
Bachelor of Technology In
Computer Science and Engineering
During the academic year 2021-2022

Mrs. V. TULASI

Assistant Professor

Project Guide

Dr . D.N.D.HARINI

Head of the Department

Computer Science and Engineering

DECLARATION

We hereby declare that this industry oriented mini project entitled **“MOVIE RECOMMENDER SYSTEM”** is a bonafide work done by us and submitted to **Department of Computer Science and Engineering, G.V.P College of Engineering (Autonomous) Visakhapatnam**, in partial fulfilment for the award of the degree of B. Tech is of our own and it is not submitted to any other university or has been published any time before.

PLACE: VISAKHAPATNAM

DATE:

POPURI TRIVENI (Reg No:19131A05H9)

METTU AMARNADH REDDY (Reg No:19131A05D2)

ROKALLA SRUTI (Reg No:19131A05K6)

MITTAKOLA HARSHA VARDHINI (Reg No:20135A0515)

ACKNOWLEDGEMENT

We would like to express our deep sense of gratitude to our esteemed institute **Gayatri Vidya Parishad College of Engineering (Autonomous),** which has provided us an opportunity to fulfil our cherished desire.

We express our sincere thanks to our Principal **Dr . A.B. KOTESWARA RAO, Gayatri Vidya Parishad College of Engineering(Autonomous)** for his encouragement to us during this project, giving us a chance to explore and learn in the form of mini project.

We express our deep sense of Gratitude to **Dr . D.N.D.HARINI, Associate Professor and Head of the Department of Computer science and Engineering, Gayatri Vidya Parishad College of Engineering(Autonomous)** for giving us an opportunity to do the project in college.

We express our profound gratitude and our deep indebtedness to our guide **Mrs. V. Tulasi, Assistant Professor, Department of Computer Science and Engineering**, whose valuable suggestions, guidance and comprehensive assistance helped us a lot in realizing my present project.

We also thank our coordinator, **Dr . CH. SITA KUMARI, Sr. Assistant Professor, Department of Computer Science and Engineering**, for the kind suggestions and guidance for the successful completion of our project work.

Finally, we would also like to thank all the members of the teaching and non-teaching staff of the Computer Science and Engineering Department for all their support in completion of our project.

POPURI TRIVENI
METTU AMARNADH REDDY
ROKALLA SRUTI
MITTAKOLA HARSHA VARDHINI

ABSTRACT

MOVIE RECOMMENDER SYSTEM is a machine learning application. Today the amount of information in the internet growth very rapidly and people need some instruments to find and access appropriate information. One of such tools is the recommender system. Movie recommender system aims to recommend movies that users may be interested in.

In our project, we have developed a content-based movie recommender system by creating different attributes elicited from user preference. We have improved the existing content based recommender system by adding some new attributes like cast, crew, user_id, genre, rating and user past experience. We have also developed the popularity-based recommender system which recommends the movies that are in trend and are most popular among the users based on their rating and vote count.

Our main vision is to develop a project to recommend the books to users based on his/her interest by using content based recommender system.

INDEX

CHAPTER 1. INTRODUCTION……………………………………………..............................1

1.1 Objective……………………………..……………………………………………..2

1.2 About the Algorithm……………………..…………....................................2

1.3 Purpose……………………………………………………...................................3

1.4 Scope………………………………………………………….………………….4

CHAPTER 2. SRS DOCUMENT…………………………………………......................5

2.1 Functional Requirements………………………......……......................................5

2.2 Non-functional Requirements……………………….……...................................5

2.3 Minimum Hardware Requirements……………..……….....................................6

2.4 Minimum Software Requirements………………..……......................................6

CHAPTER 3. ALGORITHM ANALYSIS……...7

3.1 Existing Algorithm……………………………………...7

3.2 Proposed Algorithm…………………………….………………………………… 8

3.3 Feasibility Study……………………………………………..................................9

3.4 Cost Benefit Analysis…………………………..…………………………….....12

CHAPTER 4. SOFTWARE DESCRIPTION………..……………….......................13

4.1 Jupyter notebook…………………………...……………………………….....13

4.2 Streamlit………………………………………………….……………………..13

4.3 Numpy……………..14

4.4 Pandas…………………………………………...14

4.5 NLTK………………………………………………..14

4.6 Scikit-learn(sklearn)……………………………………….…………………15

4.7 CountVectorizer……………………………………………………………….....15

4.8 PorterStemmer……………………………………………………………………. 15

4.9 Sklearn cosine similarity……………………………………………………. 15

4.10 Pickle……………………………………………………………………………. 16

4.11 PIL……………………………………………………………………………. 16

4.12 PyCharm…………………………………………………………………………..16

CHAPTER 5. PROJECT DESCRIPTION……………………..…..….........................17
5.1 Problem Definition…………………………………………………………………..17
5.2 Project Overview……………………………………………………………………17
5.3 Module Description…………………………………………………………………19
5.3.1 Streamlit Framework…………………………………………………………..19
5.3.2 Model…………………………………………………………………………….19
CHAPTER 6. SYSTEM DESIGN……………………………………….........................25
6.1 Introduction To UML………………………………………………………….....25
6.2 Building Blocks Of the ML...………………………..…......................................26
6.3 UML Diagrams…………………………..……….…………………………….....32
CHAPTER 7. DEVELOPMENT…………………….……………..….…..........................35
7.1 Datasets used………………………………...35
7.2 Sample Code…………………………………………….………………………….36
7.3 Results…………………………………………………...46
CHAPTER 8. TESTING…………………………….……...……….……....................49
8.1 Introduction Of Testing……………….………...…..49
8.2Test Code………………………………….…..………………………………......53
8.3Test Cases…………………………………….…….……………………………......56
CHAPTER 9. CONCLUSION…………………………………………….......................59
CHAPTER 10. FUTURE SCOPE……….....……………….....……………...................60
CHAPTER 11. REFERENCE LINKS………..…………………….……......................61

1. INTRODUCTION

Now-a-days, many entertainment websites and online stores are available for the people. All entertainment websites or online stores have millions /billions of items. It becomes challenging for the customers to select the right one. At this place, recommender systems come into the picture and help the user to find the right item by minimizing the options. Recommendation system helps the user to select the right item by suggesting a presumable list of items and so it has become an integral part of e-commerce, movie, book and music rendering sites and the list goes on. They are becoming one of the most popular applications of machine learning which has gained importance in recent years. The two most popular ways it can be approached/built are:

1- Content-based Filtering

2- Collaborative Filtering

3- Hybrid Filtering

CONTENT-BASED FILTERING:

Content-based filtering uses item features to recommend other items similar to what the user likes, based on their previous actions or explicit feedback.

COLLABORATIVE FILTERING:

Collaborative filtering models can recommend an item to user A based on the interests of a similar user B. There are two classes of collaborative filtering.

They are: 1. User-based collaborative filtering

2. Item-based collaborative filtering.

HYBRID FILTERING:

Hybrid Filtering is a combination of both content-based filtering, collaborative filtering, and other approaches. Hybrid recommender systems combine two or more recommendation strategies in different ways to benefit from their complementary advantages.

1.1 OBJECTIVE

“MOVIE RECOMMENDATION SYSTEM” helps us to recommend the movies the users according to the user’s choices. The movie recommender system filters the preferences according to the user’s choice and helps the user to select a movie based on his/her interest. There are many digital platforms like Amazon prime, Netflix etc that makes use of these movie recommender systems.

In our project, we are going to develop a content-based movie recommender system by creating different attributes elicited from user preference. We are just going to improve the already existed content-based recommender system by adding some new attributes like cast, crew, rating, language etc., to the system. So, By using this, we can suggest the movies to the user’s based on his profile/description and also based on the content what the user has already watched.

1.2 ABOUT THE ALGORITHM

In our project, we will be using the cosine similarity to calculate a numeric quantity that denotes the similarity between two movies. We use the cosine similarity score since it is independent of magnitude and is relatively easy and fast to calculate. Mathematically, it is defined as follows:

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

These are the following steps we'll follow :-

- Get the index of the movie given its title.
- Get the list of cosine similarity scores for that particular movie with all movies. Convert it into a list of tuples where the first element is its position and the second is the similarity score.
- Sort the aforementioned list of tuples based on the similarity scores; that is, the second element.
- Get the top 10 elements of this list. Ignore the first element as it refers to self (the movie most similar to a particular movie is the movie itself).
- Return the titles corresponding to the indices of the top elements.

While our system has done a decent job of finding movies with similar plot descriptions, the quality of recommendations is not that great. "The Dark Knight Rises" returns all Batman movies while it is more likely that the people who liked that movie are more inclined to enjoy other Christopher Nolan movies. This is something that cannot be captured by the present system.

1.3 PURPOSE

Now-a-days, many entertainment websites and online stores are available for the people. All entertainment websites or online stores have millions /billions of items. It becomes challenging for the customers to select the right one. At this place, recommender systems come into the picture and help the user to find the right item by minimizing the options. Recommendation system helps the user to select the right item by suggesting a presumable list of items and so it has become an integral part of e-commerce, movie, book and music rendering sites and the list goes on.
Recommender systems are becoming increasingly important in today’s extremely busy world. People are always short on time with the myriad tasks they need to accomplish in the limited 24 hours. Therefore, the recommender systems are important as they help them make the right choices, without having to expend their cognitive resources. Movie recommender system recommends the movies to the users based on their profile, search/browsing history, what other people with similar traits/demographics are watching, and how likely are you to watch those movies. Recommendations is not a new concept. Even when e-commerce was not that Prominent, the sales staff in retail stores recommended items to the customers for the purpose of upselling and cross-selling, ultimately maximise the profit. The aim of the movie recommender systems is also the same. Another objective of the recommendation system is to achieve customer loyalty by providing relevant content and maximising the time spent by a user on your website or channel. This also helps in increasing customer engagement.

1.4 SCOPE

Movie recommender system is one of the top research areas, currently. Due to the impact of high internet speeds, multimedia has become one of the best entertainments. Recommender system has its applications like movie recommendations, course recommendations, e-commerce etc.. Movie recommendation system scope is not limited to entertainment, but also in information sharing. Movie recommendation systems suffer from problems like Cold-start problem, Sparsity, Long-tail problem, Grey sheep problem etc.. Some of these problems can be solved or at least be minimized if we take the right decisions on what kind of movies to ignore, what movies to consider.

Now-a-days, many digital platforms use recommender systems to recommend movies to the users. Some of the digital platforms which use the recommender systems are Netflix, Amazon Prime, Aha, Hotstar etc.

2. SRS DOCUMENT

A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform.

2.1 FUNCTIONAL REQUIREMENTS

A Functional Requirement (FR) is a description of the service that the software must offer. It describes a software system or its component. A function is nothing but inputs to the software system, its behaviour, and outputs. It can be a calculation, data manipulation, business process,
user interaction, or any other specific functionality which defines what function a system is likely to perform. Functional Requirements are also called Functional Specification.

- User should select the appropriate input from the dropdown menus that was provided in the webpage and then click on the recommend button.
- The movies are recommended to the user based on the input chosen by the user.

2.2 NON-FUNCTIONAL REQUIREMENTS

NON-FUNCTIONAL REQUIREMENT (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability. Non-functional requirements are called qualities of a system, there are as follows:

- **Performance -** The average response time of the system is less.
- **Reliability** - The system is highly reliable.
- **Operability** - The interface of the system will be consistent.
- **Efficiency** - Once user has learned about the system through his interaction he can perform the task easily.
- **Understandability** - Because of user friendly interfaces, it is more understandable to the users.
- **Scalability –** The system will provide good quality and timely recommendations.

2.3 MINIMUM HARDWARE REQUIREMENTS

- **Processor:** Intel core i5
- **Hard Disk :** 1TB
- **RAM:**8GB
- **Architecture:**32bit or 64bit

2.4 MINIMUM SOFTWARE REQUIREMENTS

- **Programming Language**: Python
- **Operating System:** Windows 10
- **IDE:** Jupyter Notebook, Pycharm
- **Packages**: Pandas, Streamlit, Numpy, Pickle, CountVectorizer, Image, PorterStemmer, CosineSimilarity

3.ANALYSIS

3.1 EXISTING SYSTEMS

The idea behind Content-based movie recommender system is to recommend a movie based on a comparison between the content of the movies and a user profile/description.

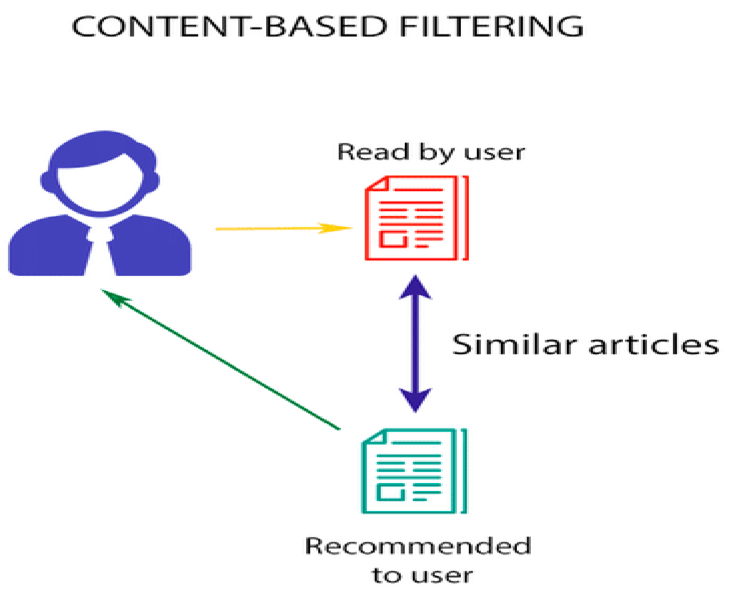


Figure3.1.1 content-based filtering

CONTENT-BASED RECOMMENDER SYSTEM THAT RECOMMENDS THE MOVIES THAT ARE SIMILAR TO THE ONES THAT A USER HAS LIKED IN THE PAST:

- This content-based recommender system is based on the data provided about the items.
- The content-based filtering algorithm recommends movies that are similar to the ones that a user has liked in the past.
- For example, if a user likes movies such as ‘Avatar’, then the recommender system tries to find similar movies to ‘Avatar’ by using the information available in the database such as leading actors, the director, genre of the film, production house etc

DRAWBACKS OF EXISTING ALGORITHM:

- The existing content-based recommender system recommends the movies to the users based on their past or previous experiences only.
- In the existing content-based recommender system, user is not much exposed to different types of movies.
- Only similar type of movies is recommended to the user.
- New users will not have any past or previous history. So, it is also one of the drawbacks of the existing content-based recommender system.
- Users will only get the recommendations related to their preferences in their profile, and recommender engine may never recommend any item with other characteristics.

3.2 PROPOSED ALGORITHM

- Users can watch the movie of his/her own preference in digital platforms contain recommender systems.
- In our project, we are going to improve the already existed content-based recommender system by adding some new attributes like cast, crew, genres etc.
- Not only content-based recommender system, we will also include popular-based recommender system in our project.
- For new users, we can recommend the top 10 movies by popular-based recommender system.
- The new users can also get the recommendations by giving the inputs like cast(actor), crew(director), genres.
- Movies to the existing users will be recommended based on the input given by the user.

ADVANTAGES OF PROPOSED MODEL:

The proposed model has many advantages. Some of the advantages of the proposed model are mentioned below:

- In our project, the existing users will be exposed to different types of movies i.e., the movies will be recommended to the user based on the input given by the user. In our project, the input that will be given by the user may be cast(actor), crew(director), genres, user-id, title of the movie.
- For new users also, our work will be useful because the new users can get the 10 popular recommendations of the movies.
- New users can also give the inputs like cast, crew, genres.

3.3 FEASIBILITY STUDY

A feasibility study is an analysis that takes all a project's relevant factors into account including economic, technical, legal, and scheduling considerations to ascertain the likelihood of completing the project successfully. A feasibility study is important and essential to evolute any proposed project is feasible or not. A feasibility study is simply an assessment of the practicality of a proposed plan or project.

The main objectives of feasibility are mentioned below:

To determine if the product is technically and financially feasible to develop, is the main aim of the feasibility study activity. A feasibility study should provide management with enough information to decide:

- Whether the project can be done.
- To determine how successful your proposed action will be.
- Whether the final product will benefit its intended users.
- To describe the nature and complexity of the project.
- What are the alternatives among which a solution will be chosen (During subsequent phases)
- To analyse if the software meets organizational requirements.

There are various types of feasibility that can be determined. They are:

Operational - Define the urgency of the problem and the acceptability of any solution, includes people-oriented and social issues: internal issues, such as manpower problems, labor objections, manager resistance, organizational conflicts, and policies; also, external issues, including social acceptability, legal aspects, and government regulations.

Technical: Is the feasibility within the limits of current technology? Does the technology exist at all? Is it available within a given resource?

Economic - Is the project possible, given resource constraints? Are the benefits that will accrue from the new system worth the costs? What are the savings that will result from the system, including tangible and intangible ones? What are the development and operational costs?

Schedule - Constraints on the project schedule and whether they could be reasonably met.

3.3.1 Economic Feasibility:

Economic analysis could also be referred to as cost/benefit analysis. It is the most frequently used method for evaluating the effectiveness of a new system. In economic analysis the procedure is to determine the benefits and savings that are expected from a candidate system and compare them with costs. Economic feasibility study related to price, and all kinds of expenditure related to the scheme before the project starts. This study also improves project reliability. It is also helpful for the decision-makers to decide the planned scheme processed latter or now, depending on the financial condition of the organization. This evaluation process also studies the price benefits of the proposed scheme. Economic Feasibility also performs the following tasks.

- Cost of packaged software/ software development.
- Cost of doing full system study.
- Is the system cost Effective?

3.3.2 Technical Feasibility:

A large part of determining resources has to do with assessing technical feasibility. It considers the technical requirements of the proposed project. The technical requirements are then compared to the technical capability of the organization. The systems project is considered technically feasible if the internal technical capability is sufficient to support the project requirements. The analyst must find out whether current technical resources can be where the expertise of system analysts is beneficial, since using their own experience and their contact with vendors they will be able to answer the question of technical feasibility.

Technical Feasibility also performs the following tasks.

- Is the technology available within the given resource constraints?
- Is the technology have the capacity to handle the solution
- Determines whether the relevant technology is stable and established.
- Is the technology chosen for software development has a large number of users so that they can be consulted when problems arise, or improvements are required?

3.3.3 Operational Feasibility:

Operational feasibility is a measure of how well a proposed system solves the problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development. The operational feasibility refers to the availability of the operational resources needed to extend research results beyond on which they were developed and for which all the operational requirements are minimal and easily accommodated. In addition, the operational feasibility would include any rational compromises farmers make in adjusting the technology to the limited operational resources available to them. The operational Feasibility also perform the tasks like

- Does the current mode of operation provide adequate response time?
- Does the current of operation make maximum use of resources.
- Determines whether the solution suggested by the software development team is acceptable.
- Does the operation offer an effective way to control the data?

Our project operates with a processor and packages installed are supported by the system.

3.4 COST BENEFIT ANALYSIS

The financial and the economic questions during the preliminary investigation are verified to estimate the following:

- The cost of the hardware and software for the class of application being considered.
- The benefits in the form of reduced cost.
- The proposed system will give the minute information, as a result.
- Performance is improved which in turn may be expected to provide increased profits.
- This feasibility checks whether the system can be developed with the available funds.
- This can be done economically if planned judicially, so it is economically feasible.
- The cost of the project depends upon the number of man-hours required.

4. SOFTWARE DESCRIPTION

4.1 JUPYTER NOTEBOOK

Jupyter notebook is an open-source web application for R and Python programming language for machine learning as well as data science projects.

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. It is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Jupyter Notebook runs via a web browser, the notebook itself could be hosted on your local machine or on a remote server. Originally developed for data science applications written in Python, R, and Julia, Jupyter Notebook is useful in all kinds of ways for all kinds of projects:

Data visualizations: Jupyter Notebook lets you author visualizations, but also share them and allow interactive changes to the shared code and data set.

Code sharing: Cloud services like GitHub and Pastebin provide ways to share code, but they're largely non-interactive. With a Jupyter Notebook, you can view code, execute it, and display the results directly in your web browser.

Documenting code samples: If you have a piece of code and you want to explain line-by-line how it works, with live feedback all along the way, you could embed it in a Jupyter Notebook.

4.2 STREAMLIT

Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. Streamlit allows you to write an app the same way you write a python code. Streamlit makes it seamless to work on the interactive loop of coding and viewing results in the web app. It can deploy any machine learning model and any python project with ease and without worrying about the frontend. It provide tools, libraries and technologies that allow you to build a web app.

A few of the advantages of using Streamlit tools like Dash and Flask:

- It embraces Python scripting; No HTML,CSS,JavaScript knowledge is needed!
- Less code is needed to create a beautiful application
- No callbacks are needed since widgets are treated as variables
- Data caching simplifies and speeds up computation pipelines.

4.3 NUMPY

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python. In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

4.4 PANDAS

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python. read_csv is an important pandas function to read csv files and do operations on it. Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

4.5 NLTK

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It is an essential library supports tasks such as classification, stemming, tagging, parsing, semantic reasoning, and tokenization in Python. It's basically the main tool for natural language processing and machine learning.

4.6 SCIKIT-LEARN (SKLEARN)

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

4.7 COUNTVECTORIZER

CountVectorizer is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. This is helpful when we have multiple such texts, and we wish to convert each word in each text into vectors (for using in further text analysis).

4.8 PORTERSTEMMER

nltk.stem is a package that performs stemming using different classes. PorterStemmer is one of the classes. The Porter stemming algorithm (or 'Porter stemmer') is a process for removing the commoner morphological and inflexional endings from words in English. Stemming is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language.

4.9 SKLEARN COSINE SIMILARITY

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity. We can import cosine similarity function from sklearn.metrics.pairwise. It will calculate cosine similarity between two numpy array. cosine similarity is one the best way to judge or measure the similarity between documents. Irrespective of the size, This similarity measurement tool works fine.

4.10 PICKLE

Pickle is the standard way of serializing objects in Python. You can use the pickle operation to serialize your machine learning algorithms and save the serialized format to a file. Later you can load this file to deserialize your model and use it to make new predictions. The pickle module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure. Pickle model provides the following functions –

- pickle.dump to serialize an object hierarchy, you simply use dump().
- pickle.load to deserialize a data stream, you call the loads() function.

4.11 PIL

PIL is the Python Imaging Library which provides the python interpreter with image editing capabilities. The Image module provides a class with the same name which is used to represent a PIL image. It incorporates lightweight image processing tools that aids in editing, creating and saving images. It is the de facto image processing package for Python language.

4.12 PYCHARM

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a wide range of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development.

Features of Pycharm :

- Coding assistance and analysis, with code completion, syntax and error highlighting, linter integration, and quick fixes
- Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages
- Python refactoring: includes rename, extract method, introduce variable, introduce constant, pull up, push down and others
- Support for web frameworks: Django, web2py and Flask.
- Integrated Python debugger ,Integrated unit testing, with line-by-line code coverage.
- Google App Engine Python development
- Version control integration: unified user interface for Mercurial, Git, Subversion, Perforce and CVS with change lists and merge.
- Support for scientific tools like matplotlib, numpy and scipy.

5. PROJECT DESCRIPTION

5.1 PROBLEM DEFINITION

The Movie Recommender System is an intelligent algorithm which reduces the overhead of the people. This provides benefit to the user by recommending movies to the users based on his/her interest. The E-commerce site to network security, all demands the need for the recommended system to increase their revenue rate. We use cosine similarity algorithm which is a nearest neighbour approach for prediction of movies to the users. Movie Recommender system is technology which aids people to filter out particular movies according to specific user preference from the overloaded information. Our main objective is to develop a movie recommender system for online movies users which considers their preferences and choices.

5.2 PROJECT OVERVIEW

The main objective is to create a machine learning model to recommend relevant movies to users based on popularity and user interests. In addition to the ML Model prediction, we also have taken into account the movie recommendation for a totally new user.

The main aim of our project is to recommend the movies to the users by considering their preferences and choices. In our project, we have developed a content-based movie recommender system by creating different attributes elicited from user preference. Our project comprises of two parts, first one is we will ask the user to give the desired attributes like cast, crew, genre etc. Based on the user's interests, the similar movies are recommended to the user. The movie is also recommended based on user past experience. If the user select the past movie then similar movies to his past movie is recommended. We have improved the existing content based recommender system by adding some new attributes like cast, crew, genre, rating and user past experience. We used **Cosine similarity** that uses **nearest neighbor approach** which is a machine learning algorithm to identify the similarity of two or more movies to each other based on algorithmic distance functions. This similarity algorithm compute the similarity of pairs of nodes using different vector-based metrics.

The second part is we have also developed the popularity-based recommender system which recommends the movies that are in trend and are most popular among the users based on their rating and vote count. We used **weighted rating technique** and **popularity score** to maximize accuracy recommend popular movies to the users.

The steps involved in our project are:

1. Collection of datasets.
2. Data cleaning and preprocessing for training and testing.
3. Training the model.
4. Repeating steps 2 and 3 for different ratios of training and testing data to maximize accuracy.
5. Frontend creation
6. Linking both frontend and backend.

The output of our project consists of recommendations of similar movies to the users based on their preferences and also if the user wants popular recommendations then the popular movies will also be displayed for the user.

5.2.1 Importing of Libraries :

The first step is to import the libraries required to preprocess and cleaning of data and other libraries are required for constructing the model outputs. We used streamlit framework for the output.

```
import pandas as pd

import numpy as np

import nltk

from nltk.stem.porter import PorterStemmer

#import sklearn library

from sklearn.metrics.pairwise import cosine_similarity

import pickle

import streamlit as st

from PIL import Image
```

5.2.2 Collection of Datasets :

Using the pandas datareader library, we will upload the datasets required for the prediction of books to the different users based on their preferences.

5.3. MODULE DESCRIPTION

5.3.1 Streamlit Framework

Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. Streamlit allows you to write an app the same way you write a python code. Streamlit makes it seamless to work on the interactive loop of coding and viewing results in the web app. It can deploy any machine learning model and any python project with ease and without worrying about the frontend. It provide tools, libraries and technologies that allow you to build a web app. A few of the advantages of using Streamlit tools like Dash and Flask. It embraces Python scripting, No HTML, CSS, JavaScript knowledge is needed. Less code is needed to create a beautiful application. No callbacks are needed since widgets are treated as variables. Data caching simplifies and speeds up computation pipelines.

5.3.2 MODEL

MACHINE LEARNING

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. Over the next couple of decades, the technological developments around storage and processing power will enable some innovative products that we know and love today, such as Netflix's recommendation engine or self-driving cars.

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

Types of machine learning:

1. Supervised machine learning
2. Unsupervised machine learning
3. Semi-supervised learning

1.Supervised machine learning

Supervised learning, also known as supervised machine learning, is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately. This occurs as part of the cross validation process to ensure that the model avoids overfitting or underfitting. Supervised learning helps organizations solve for a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox. Some methods used in supervised learning include neural networks, naïve bayes, linear regression, logistic regression, random forest, support vector machine (SVM), and more.

2.Unsupervised machine learning

Here, the machine learning algorithm studies data to identify patterns. There is no answer key or human operator to provide instruction. Instead, the machine determines the correlations and relationships by analysing available data. In an unsupervised learning process, the machine learning algorithm is left to interpret large data sets and address that data accordingly. The algorithm tries to organise that data in some way to describe its structure. This might mean grouping the data into clusters or arranging it in a way that looks more organised. As it assesses more data, its ability to make decisions on that data gradually improves and becomes more refined.

3.Semi-supervised learning

Semi-supervised learning is similar to supervised learning, but instead uses both labelled and unlabelled data. Labelled data is essentially information that has meaningful tags so that the algorithm can understand the data, whilst unlabelled data lacks that information. By using this combination, machine learning algorithms can learn to label unlabelled data.

5.3.2.1 Recommender systems in machine learning

A recommendation engine is a type of data filtering tool using machine learning algorithms to recommend the most relevant items to a particular user or customer. It operates on the principle of finding patterns in consumer behavior data, which can be collected implicitly or explicitly. The recommender system deals with a large volume of information present by filtering the most important information based on the data provided by a user and other factors that take care of the user’s preference and interest. It finds out the match between user and item and imputes the similarities between users and items for recommendation.

Types of recommender systems

1. Popularity-Based Recommender System
2. Content-Based Recommendation System
3. Collaborative Filtering
   a) User-based nearest-neighbor collaborative filtering
   b) Item-based nearest-neighbor collaborative filtering

In our project we have used popularity based recommender system and content based recommender system which we have extended by providing additional attributes or choices to the user.

1. Popularity-Based Recommender System

It is a type of recommendation system which works on the principle of popularity and or anything which is in trend. These systems check about the product or movie which are in trend or are most popular among the users and directly recommend those.

For example, if a movie is watched by most of the people then the system will get to know that movie is the most popular, so for every new user who just signed it, the system will recommend that movie to that new user also and chances becomes high that the new user will also watch that movie.

Merits of popularity based recommender system

- It does not suffer from cold start problems which means on day 1 of the business also it can recommend products on various different filters.
- There is no need for the user's historical data.

Demerits of popularity based recommender system

- Not personalized
- The system would recommend the same sort of products/movies which are solely based upon popularity to every other user.

Example

- Google News: News filtered by trending and most popular news.
- YouTube: Trending videos.

Popularity recommender system model:

Popularity model is a straightforward model based on the popularity and rating of an item given to it by the user

A popularity based model does not suffer from cold start problems which means on day 1 of the business also it can recommend products on various different filters. There is no need for the user’s historical data.

The popularity index used for our books dataset was weighted rating.

$$WR = [(v * R)/(v + m)] + [(m * c)/(v + m)]$$

where,

v is the number of votes for the movies;

m is the minimum votes required to be listed in the chart;

R is the average rating of the movie; and

C is the mean vote across the whole report.

The Popularity Based Recommender provides a general chart of recommended movies to all the users. They are not sensitive to the interests and tastes of a particular user. It is not personalised and the system would recommend the same sort of products/movies which are solely based upon popularity to every other user.

2. Content-Based Recommender System

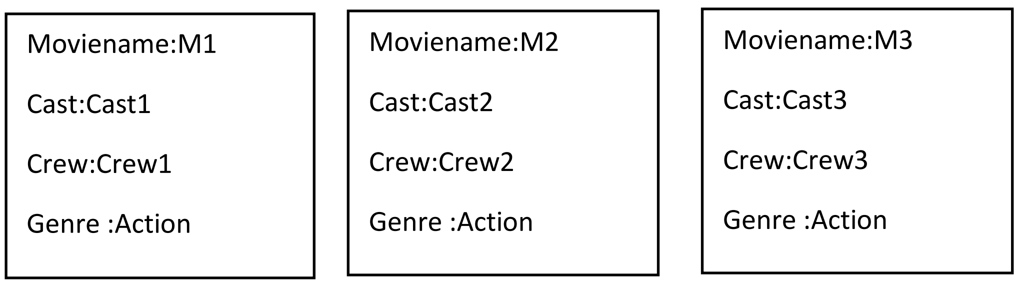
It is another type of recommender system which works on the principle of similar content. If a user is watching a movie, then the system will check about other movies of similar content or the same genre of the movie the user is watching. There are various fundamentals attributes that are used to compute the similarity while checking about similar content.

To explain more about how exactly the system works, an example is stated below:

Configuration of different models of different movies of same genre for recommendation system example.

Figure1: Different models of movies.

Content based recommender system model :



Moviename:M1
Cast:Cast1
Crew:Crew1
Genre :Action

Moviename:M2
Cast:Cast2
Crew:Crew2
Genre :Action

Moviename:M3
Cast:Cast3
Crew:Crew3
Genre :Action

Figure 1 image shows the different models of movies. If a person is watching for M1 then, M2 and M3 are recommended to the user based on similarity between the Movie name or cast or crew or genre. We used nearest neighbors algorithm to find the similarity between the movies.

Nearest Neighbours (Supervised Learning)

The K-Nearest-Neighbour algorithm estimates how likely a data point is to be a member of one group or another. It essentially looks at the data points around a single data point to determine what group it is actually in. For example, if one point is on a grid and the algorithm is trying to determine what group that data point is in (Group A or Group B, for example) it would look at the data points near it to see what group the majority of the points are in.

In the classification setting, the K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given "unseen" observation. Similarity is defined according to a distance metric between two data points. A popular one is the cosine similarity method.

Cosine Similarity:

Cosine of the angle between the two vectors of the item, vectors of A and B is calculated for imputing similarity. If the vectors are closer, then small will be the angle and large will be the cosine.

Cosine similarity formula is

$$\mathrm{Cos}(x, y) = x \,.\, y \,/\, ||x|| * ||y||$$

where,

x . y = product (dot) of the vectors 'x' and 'y'.

||x|| and ||y|| = length of the two vectors 'x' and 'y'.

||x|| * ||y|| = cross product of the two vectors 'x' and 'y'.

Merits

- There is no requirement for much of the user's data.
- We just need item data that enable us to start giving recommendations to users.
- A content-based recommender engine does not depend on the user's data, so even if a new user comes in, we can recommend the user as long as we have the user data to build his profile.
- It does not suffer from a cold start.

Demerits

- Items data should be in good volume.
- Features should be available to compute the similarity.

6. SYSTEM DESIGN

6.1 INTRODUCTION TO UML:

Unified Modeling Language **(UML)** is a general-purpose modeling language. The main aim of UML is to define a standard way to **visualize** the way a system has been designed. It is quite like blueprints used in other fields of engineering. UML is not a programming language, it is rather a visual language. We use UML diagrams to portray the behaviour and structure of a system. UML helps software engineers, businessmen and system architects with modeling, design and analysis. The Object Management Group (OMG) adopted Unified Modelling Language as a standard in 1997. It's been managed by OMG ever since. International Organization for Standardization (ISO) published UML as an approved standard in 2005. UML has been revised over the years and is reviewed periodically.

Why we need UML

- Complex applications need collaboration and planning from multiple teams and hence require a clear and concise way to communicate amongst them.
- Businessmen do not understand code. So, UML becomes essential to communicate with non-programmers' essential requirements, functionalities and processes of the system.
- A lot of time is saved down the line when teams can visualize processes, user interactions and static structure of the system.

UML is linked with **object-oriented** design and analysis. UML makes the use of elements and forms associations between them to form diagrams. Diagrams in UML can be broadly classified as:

- ➢ **Structural Diagrams** – Capture static aspects or structure of a system. Structural Diagrams include Component Diagrams, Object Diagrams, Class Diagrams and Deployment Diagrams.

➢ **Behaviour Diagrams** – Capture dynamic aspects or behaviour of the system. Behaviour diagrams include Use Case Diagrams, State Diagrams, Activity Diagrams and Interaction Diagrams.

Building Blocks of the UML Building Blocks of the UML Building Blocks of the UML

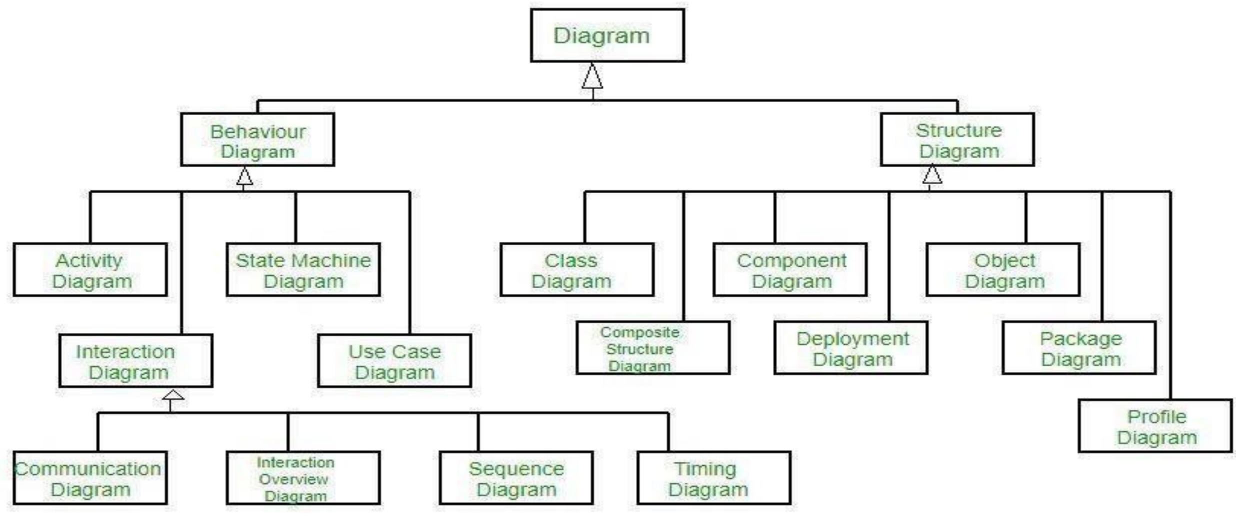


FIGURE 6.1.1 BUILDING BLOCKS IN UML

6.2 Building Blocks of the UML

The vocabulary of the UML encompasses three kinds of building blocks:

- Things
- Relationships
- Diagrams

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

Things in the UML

There are four kinds of things in the UML:

- ➢ Structural things
- ➢ Behavioural things
- ➢ Grouping things
- ➢ Annotational things

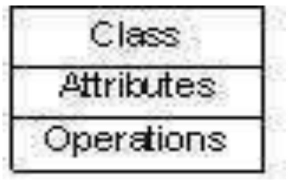
These things are the basic object-oriented building blocks of the UML. You use them to write well-formed models.

Structural Things

Structural things are the nouns of UML models. These are the mostly static parts of a model, representing elements that are either conceptual or physical. Collectively, the structural things are called classifiers.

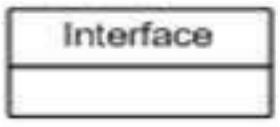
A class is a description of a set of objects that share the same attributes, operations, relationships, and semantics. A class implements one or more interfaces. Graphically, a class is rendered as a rectangle, usually including its name, attributes, and operations

Class - A Class is a set of identical things that outlines the functionality and properties of an object. It also represents the abstract class whose functionalities are not defined. Its notation is as follows

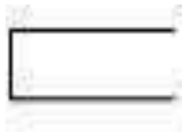


Interface - A collection of functions that specify a service of a class or component, i.e.

Externally visible behavior of that class.

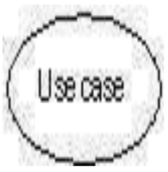


Collaboration - A larger pattern of behaviors and actions. Example: All classes and behaviors that create the modeling of a moving tank in asimulation.

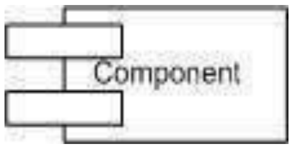


Use Case - A sequence of actions that a system performs that yields an observable result.

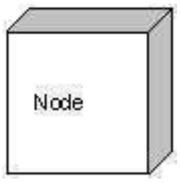
Used to structure behavior in a model. Is realized by collaboration.



Component - A physical and replaceable part of a system that implements a number of interfaces. Example: a set of classes, interfaces, and collaborations.



Node - A physical element existing at run time and represents are source.

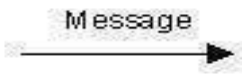


Behavioral Things

Behavioral things are the dynamic parts of UML models. These are the verbs of a model, representing behavior over time and space. In all, there are three primary kinds of behavioral things

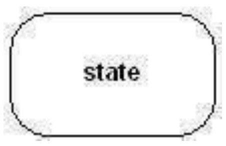
- Interaction
- State machine **Interaction**

It is a behavior that comprises a set of messages exchanged among a set of objects or roles within a particular context to accomplish a specific purpose. The behavior of a society of objects or of an individual operation may be specified with an interaction. An interaction involves a number of other elements, including messages, actions, and connectors (the connection between objects). Graphically, a message is rendered as a directed line, almost always including the name of its operation.



State machine

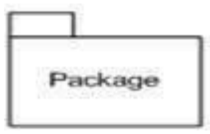
State machine is a behaviour that specifies the sequences of states an object or an interaction goes through during its lifetime in response to events, together with its responses to those events. The behaviour of an individual class or a collaboration of classes may be specified with a state machine. A state machine involves a number of other elements, including states, transitions (the flow from state to state), events (things that trigger a transition), and activities (the response to a transition). Graphically, a state is rendered as a rounded rectangle, usually including its name and its substates.



Grouping Things

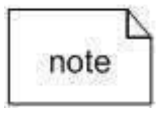
Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available.

Package − Package is the only one grouping thing available for gathering structural and behavioural things.



Annotational Things

Annotational things are the explanatory parts of UML models. These are the comments you may apply to describe, illuminate, and remark about any element in a model. There is one primary kind of annotational thing, called a note. A note is simply a symbol for rendering constraints and comments attached to an element or a collection of elements.



Relationships in the UML

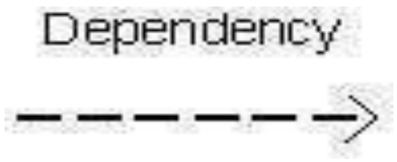
Relationship is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships in the UML:

- Dependency
- Association
- Generalization
- Realization

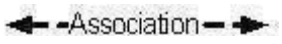
Dependency

It is an element (the independent one) that may affect the semantics of the other element (the dependent one). Graphically, a dependency is rendered as a dashed line, possibly directed, and occasionally including a label.



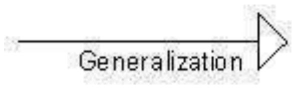
Association

Association is basically a set of links that connects the elements of a UML model. It
also describes how many objects are taking part in that relationship.



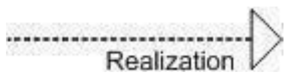
Generalization

It is a specialization/generalization relationship in which the specialized element (the child) builds on the specification of the generalized element (the parent). The child shares the structure and the behavior of the parent. Graphically, a generalization relationship is rendered as a solid line with a hollow arrowhead pointing to the parent



Realization

Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility, which is not implemented and the other one implements them. This relationship exists in case of interfaces.



6.3 UML DIAGRAMS

UML is a modern approach to modeling and documenting software. It is based on **diagrammatic representations** of software components. It is the final output, and the diagram represents the system.

UML includes the following

- ➢ Class diagram
- ➢ Object diagram
- ➢ Component diagram
- ➢ Composite structure diagram
- ➢ Use case diagram
- ➢ Sequence diagram
- ➢ Communication diagram
- ➢ State diagram
- ➢ Activity diagram

SYSTEM ARCHITECTURE

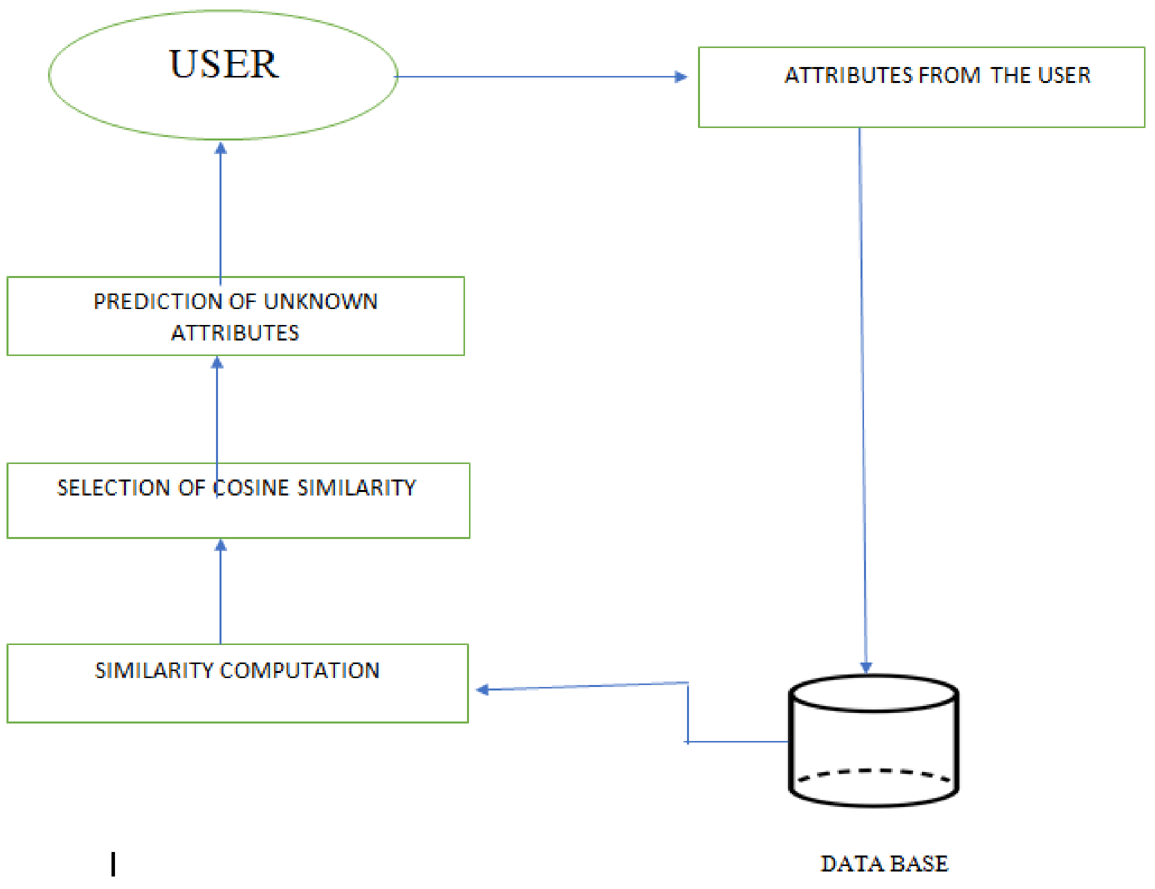

USER
ATTRIBUTES FROM THE USER
PREDICTION OF UNKNOWN ATTRIBUTES
SELECTION OF COSINE SIMILARITY
SIMILARITY COMPUTATION
DATA BASE

Figure 6.3.1 System Architecture

The Figure 6.1 describes the architecture of the online bookstore with recommendation and searching. The user request resources in the web browser in the client side, the system processes them using the system modules and database and responses with new view.

USE CASE DIAGRAM

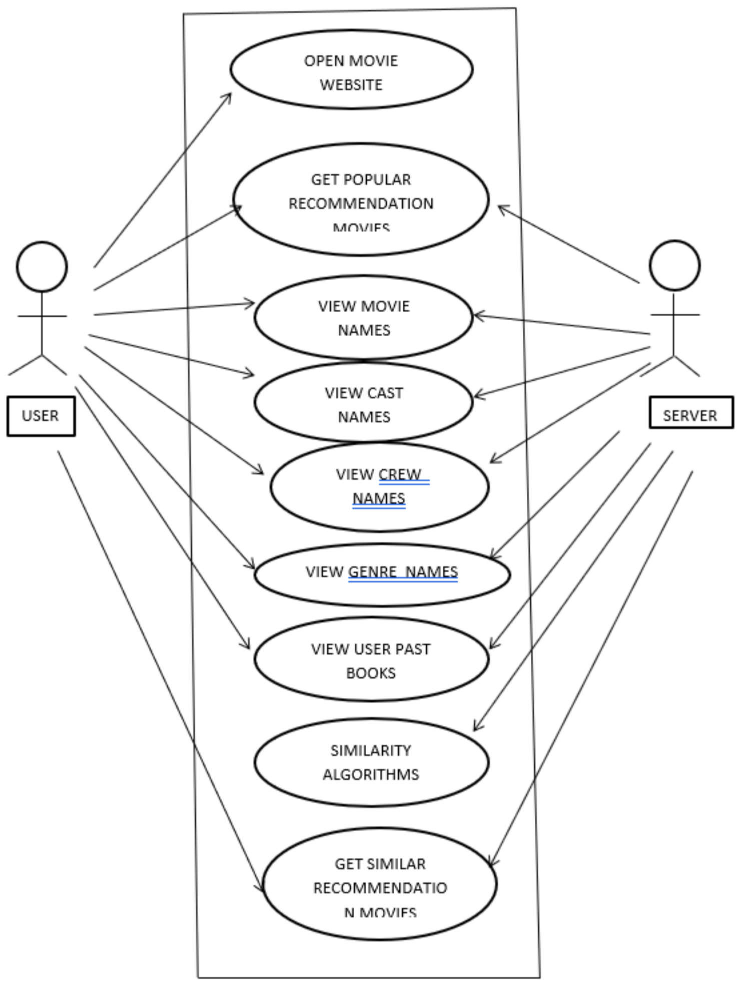
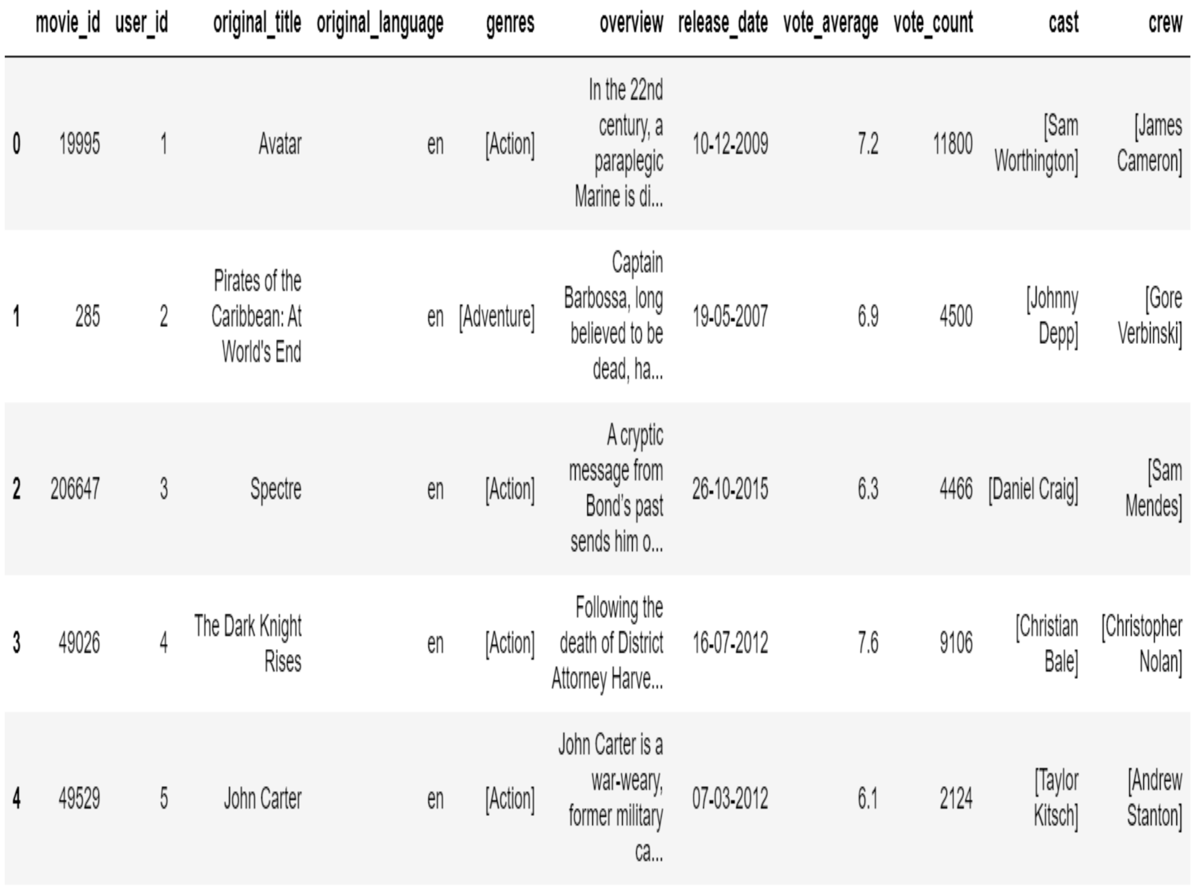


Figure 6.3.2 Use case Diagram

7. DEVELOPMENT

7.1 DATASET USED

- A DATASET is a set or collection of data. This set is normally presented in a tabular pattern. Every column describes a particular variable. Each row corresponds to a given member of the data set, as per the given question. This is a part of DATA MANAGEMENT.
- Some types of Datasets are: Numerical, Bivariate, Multivariate, Correlational, Categorical.
- We have Collected MOVIE datasets from KAGGLE.COM. Our dataset contains nearly 200 movies. Sample of our dataset is shown below:



	movie_id	user_id	original_title	original_language	genres	overview	release_date	vote_average	vote_count	cast	crew
0	19995	1	Avatar	en	[Action]	In the 22nd century, a paraplegic Marine is di...	10-12-2009	7.2	11800	[Sam Worthington]	[James Cameron]
1	285	2	Pirates of the Caribbean: At World's End	en	[Adventure]	Captain Barbossa, long believed to be dead, ha...	19-05-2007	6.9	4500	[Johnny Depp]	[Gore Verbinski]
2	206647	3	Spectre	en	[Action]	A cryptic message from Bond's past sends him o...	26-10-2015	6.3	4466	[Daniel Craig]	[Sam Mendes]
3	49026	4	The Dark Knight Rises	en	[Action]	Following the death of District Attorney Harve...	16-07-2012	7.6	9106	[Christian Bale]	[Christopher Nolan]
4	49529	5	John Carter	en	[Action]	John Carter is a war-weary, former military ca...	07-03-2012	6.1	2124	[Taylor Kitsch]	[Andrew Stanton]

Figure 7.1.1 Sample datasets

7.2 SAMPLE CODE

#importing pandas and numpy

```
import numpy as np
import pandas as pd
```

#load the movies.csv and credits.csv in movies and credits dataframe

```
movies = pd.read_csv('tmdb_200_movies.csv')
credits = pd.read_csv('tmdb_200_credits.csv')
```

#view the view the movies and credits dataframe

```
movies.head()
credits.head()
```

#merging the movies and credits

```
movies = movies.merge(credits,on='title’)
```

#taking only some attributes into the table or dataset

```
movies=movies[['movie_id','user_id','original_title','original_language','genres','overview','rel
ease_date','vote_average','vote_count','cast','crew’]]
movies.info()
```

#checking if any null values are in my movies.csv dataset

```
movies.isnull().sum()
movies.duplicated().sum()
```

#importing abstract module

```
movies.iloc[0].genres
import ast
ast.literal_eval('[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14,
"name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]’)
```

#extracting the genres from genres feature

```
def convert(obj):
   L=[]
   counter = 0
   for i in ast.literal_eval(obj):
      if counter != 1:
         L.append(i['name'])
         counter+=1
      else:
         break
   return L
movies['genres'].apply(convert)
movies['genres']=movies['genres'].apply(convert)
```

#extracting the cast(actor) from cast feature

```
def convert3(obj):
   L=[]
   counter = 0
   for i in ast.literal_eval(obj):
      if counter != 1:
         L.append(i['name'])
         counter+=1
      else:
         break
   return L
movies['cast'].apply(convert3)
movies['cast']=movies['cast'].apply(convert3)
```

#extracting the crew(director) from the crew feature

```
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'].apply(fetch_director)
movies['crew']=movies['crew'].apply(fetch_director
```

#overview is converted into list

```
movies['overview']=movies['overview'].apply(lambda x:str(x).split())
```

#new attribute tags is created in movies.csv by combing overview, genres, cast, crew

```
movies['tags']=movies['overview']+movies['genres']+movies['cast']+movies['crew’]
```

#cast, crew, genres, overview, tags are converted from list to string

```
movies['cast']=movies['cast'].apply(lambda x:" ".join(x))
movies['crew']=movies['crew'].apply(lambda x:" ".join(x))
movies['genres']=movies['genres'].apply(lambda x:" ".join(x))
movies['overview']=movies['overview'].apply(lambda x:" ".join(x))
movies['tags']=movies['tags'].apply(lambda x:" ".join(x))
movies.head()
```

#create new dataframe and view it

```
movies['tags']=movies['tags'].apply(lambda x:x.lower())
new_df=movies[['movie_id','user_id','original_title','original_language','genres','overview','rel
ease_date','vote_average','cast','crew','tags']]
new_df
```

#importing countvectorizer from scikit-learn

```
from sklearn.feature_extraction.text import CountVectorizer
```

#removing stopwords and converting the tags into vectors

```
cv=CountVectorizer(max_features=200,stop_words='english')
cv.fit_transform(new_df['tags']).toarray()
cv.fit_transform(new_df['tags']).toarray().shape
vectors=cv.fit_transform(new_df['tags']).toarray()
vectors
len(cv.get_feature_names())
cv.get_feature_names()
```

#importing nltk

```
import nltk
```

#importing PorterStemmer from nltk

```
from nltk.stem.porter import PorterStemmer
```

#Stemming

```
ps=PorterStemmer()
def stem(text):
   y=[]
   for i in text.split():
       y.append(ps.stem(i))
   return " ".join(y)
ps.stem('loved’)
new_df['tags'].apply(stem)
new_df['tags']=new_df['tags'].apply(stem)


from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=200,stop_words='english')
cv.fit_transform(new_df['tags']).toarray()
vectors=cv.fit_transform(new_df['tags']).toarray()
vectors
cv.get_feature_names()
```

#importing cosine similarity from scikit-learn

```
from sklearn.metrics.pairwise import cosine_similarity
```

#Finding similarity between the vectors

```
cosine_similarity(vectors)
similarity=cosine_similarity(vectors)
similarity[0]
 new_df['original_title'] == 'Avatar’
new_df[new_df['original_title'] == 'Avatar']
new_df[new_df['original_title'] == 'Avatar'].index
similarity[0]
```

#sorting and enumerating the similarity

```
sorted(similarity[0])
sorted(similarity[0])[-10:-1]
sorted(similarity[0],reverse=True)
enumerate(similarity[0])
list(enumerate(similarity[0]))
sorted(list(enumerate(similarity[0])),reverse=True)
```

#popularity based recommender system

```
C= movies['vote_average'].mean()
m= movies['vote_count'].quantile(0.5)
q_movies = movies.loc[movies['vote_count'] >= m]
q_movies.shape
```

```
def weighted_rating(x, m=m, C=C):
    v = x['vote_count']
    R = x['vote_average']
    return (v/(v+m) * R) + (m/(m+v) * C)
q_movies['score'] = q_movies.apply(weighted_rating, axis=1)
```

#Sorting movies based on score calculated above

```
q_movies = q_movies.sort_values('score', ascending=False)
```

#Printing the top 10 movies

```
q_movies[['original_title', 'vote_count', 'vote_average', 'score']].reset_index(drop=True).head(10)
q_movies.original_title
```

#recommending the movies based on the original_title

```
def recommend(movie):
    movie_index=new_df[new_df['original_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]].original_title)
recommend('Spider-Man’)
```

#recommending the movies based on the cast

```
def recommend(movie):
        recommended_movie_names = []
        dex = []
        for i, s in enumerate(movies['cast']):
            if movie in s:
                cast_index = i
                m = (movies.iloc[i].original_title)
                dex.append(m)
        n = len(dex[0:5])
        idex = '\n'.join(dex[0:5])
        distances = similarity[cast_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6 -
n]
        for i in movies_list:
            idex = idex + '\n' + (movies.iloc[i[0]].original_title)
        recommended_movie_names = idex.split('\n')
        return recommended_movie_names
recommend('Johnny Depp')
```

#recommending the movies based on the crew

```
def recommend(movie):
        recommended_movie_names = []
        dex = []
        for i, s in enumerate(movies['crew']):
            if movie in s:
                crew_index = i
                m = (movies.iloc[i].original_title)
                dex.append(m)
        n = len(dex[0:5])
        idex = '\n'.join(dex[0:5])
        distances = similarity[crew_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6 -
n]
        for i in movies_list:
            idex = idex + '\n' + (movies.iloc[i[0]].original_title)
        recommended_movie_names = idex.split('\n')
        return recommended_movie_names
recommend('James Cameron')
def icst(the_list, substring):
    for i, s in enumerate(the_list):
        if substring in s:
              return i
    return -1
```

#recommending the movies based on the user_id

```
def recommend(user_id):
    user_index=icst(new_df['user_id'],user_id)
    distances=similarity[user_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]].original_title
def recommend(movie):
        recommended_movie_names = []
        dex = []
        for i, s in enumerate(movies['genres']):
            if movie in s:
                genres_index = i
                m = (movies.iloc[i].original_title)
                dex.append(m)
        n = len(dex[0:5])
        idex = '\n'.join(dex[0:5])
        distances = similarity[genres_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6 -
n]
        for i in movies_list:
            idex = idex + '\n' + (movies.iloc[i[0]].original_title)
        recommended_movie_names = idex.split('\n')
        return recommended_movie_names
recommend('Action')
```

#importing pickle

```
import pickle
```

#dump the new_df, similarity, q_movies into movies.pkl, similarity.pkl, popular.pkl

```
pickle.dump(new_df,open('movies.pkl','wb'))
pickle.dump(similarity,open('similarity.pkl','wb'))
pickle.dump(q_movies,open('popular.pkl','wb'))
```

STREAMLIT CODE(INTERFACE CODE)

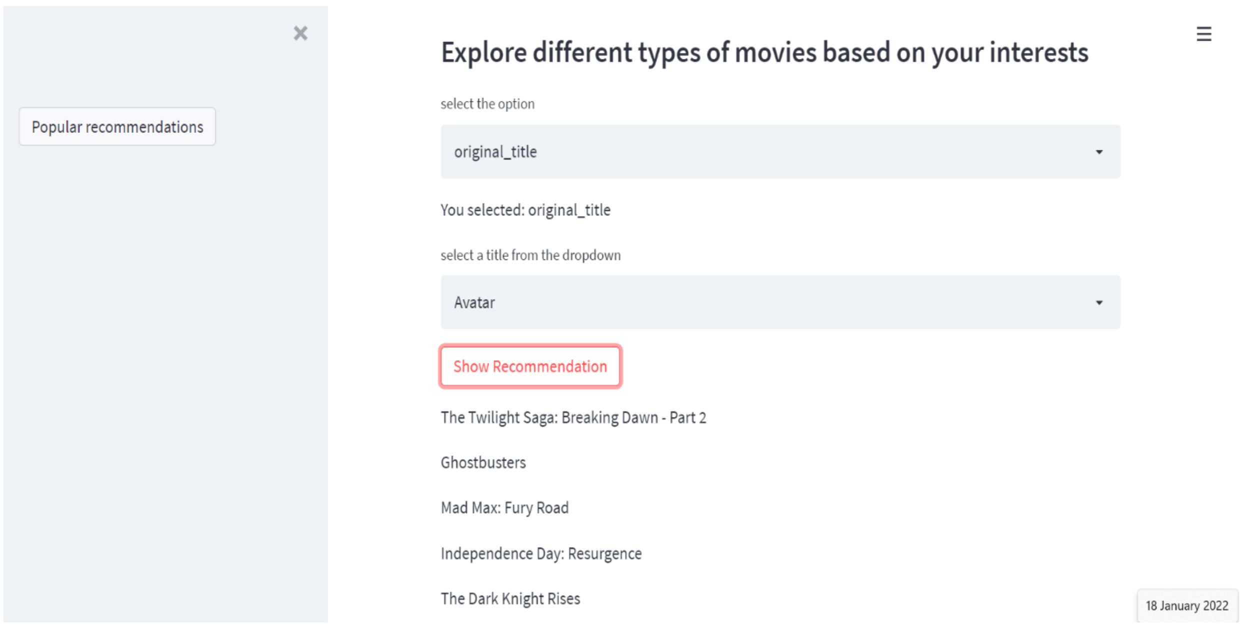
```
import streamlit as st
import pickle
import pandas as pd
import numpy as np

st.title("Movie Recommender system")
st.subheader("Your choices matters ")
st.write("""
     ABOUT US\n
       You are at the right place!Here you can fill all your movie watching dreams!!!\n
         Online movie recommendations offers many types of movies together\n
         to fulfill readers requirements based on their interests.\n
       This system also provide popular recommendations so user can get right popular choices.
    """)

from PIL import Image

image=Image.open("img_1.png")
st.image(image,caption="All types of movies are available")

movies_list=pickle.load(open('movies.pkl','rb'))
movies=pd.DataFrame(movies_list)
similarity=pickle.load(open('similarity.pkl','rb'))

title_list=movies_list['original_title'].values
cast_list=movies_list['cast'].values
crew_list=movies_list['crew'].values
user_list=movies_list['user_id'].values
genre_list=movies_list['genres'].values

st.subheader("""  Explore different types of movies based on your interests  """)
option=st.selectbox('select the option',('original_title','cast','crew','user_id','genres'))

def unique(genre_list):
  unique_list=[]
  for x in genre_list:
    if x not in unique_list:
      unique_list.append(x)
  return unique_list

def icst(the_list, substring):
  for i, s in enumerate(the_list):
    if substring in s:
      return i
  return -1
```

```
movies['user_id']=movies['user_id'].astype(str)

recommend(movie):
    if option == 'cast':
        recommended_movie_names = []
        dex = []        for i, s in enumerate(movies['cast']):
            if movie in s:
                cast_index = i
                m = (movies.iloc[i].original_title)
                dex.append(m)
        n = len(dex[0:5])
        idex = '\n'.join(dex[0:5])
        distances = similarity[cast_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6 -
n]
        for i in movies_list:
            idex = idex + '\n' + (movies.iloc[i[0]].original_title)
        recommended_movie_names = idex.split('\n')
        return recommended_movie_names

elif option=='crew':
        recommended_movie_names = []
        dex = []
        for i, s in enumerate(movies['crew']):
            if movie in s:
                crew_index = i
                m = (movies.iloc[i].original_title)
                dex.append(m)
        n = len(dex[0:5])
        idex = '\n'.join(dex[0:5])
        distances = similarity[crew_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6 -
n]
        for i in movies_list:
            idex = idex + '\n' + (movies.iloc[i[0]].original_title)
        recommended_movie_names = idex.split('\n')
        return recommended_movie_names

  elif option == 'original_title':
        movie_index = movies[movies['original_title'] == movie].index[0]
        distances = similarity[movie_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6]
        recommended_movies= []
        for i in movies_list:
            recommended_movies.append(movies.iloc[i[0]].original_title)
        return recommended_movies
```

```
elif option=='user_id':
        user_index = icst(movies['user_id'], movie)
        distances = similarity[user_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6]
        recommended_movies = []
        for i in movies_list:
            recommended_movies.append(movies.iloc[i[0]].original_title)
        return recommended_movies

 elif option=='genres':
        recommended_movie_names = []
        dex = []
        for i, s in enumerate(movies['genres']):
            if movie in s:
                genres_index = i
                m = (movies.iloc[i].original_title)
                dex.append(m)
        n = len(dex[0:5])
        idex = '\n'.join(dex[0:5])
        distances = similarity[genres_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6 -
n]
        for i in movies_list:
            idex = idex + '\n' + (movies.iloc[i[0]].original_title)
        recommended_movie_names = idex.split('\n')
        return recommended_movie_names

st.write('You selected:', option)
if(option=='original_title'):
    selected_choice = st.selectbox("select a title from the dropdown",title_list)
if(option=='cast'):
    selected_choice = st.selectbox("select a cast from the dropdown",cast_list)
if(option=='crew'):
    selected_choice = st.selectbox("select a crew from the dropdown",crew_list)
if(option=='user_id'):
    selected_choice=st.selectbox("select a user_id from the dropdown",user_list)
if(option=='genres'):
    selected_choice=st.selectbox("select a user_id from the dropdown",unique(genre_list))
if st.button('Show Recommendation'):
    recommended_movie_names = recommend(selected_choice)
    for i in recommended_movie_names:
        st.write(i)

if st.sidebar.button("Popular recommendations "):
    st.sidebar.write("Popular recommendations for the movies ")
    popular_movies_list = pickle.load(open('popular.pkl', 'rb'))
    popular_movies = pd.DataFrame(popular_movies_list)
    x = popular_movies_list["original_title"].values
    st.sidebar.write("Popular recommendations")
    st.sidebar.write(x)
```

7.3 RESULTS

ORIGINAL_TITLE:


Popular recommendations

Explore different types of movies based on your interests

select the option

original_title

You selected: original_title

select a title from the dropdown

Avatar

Show Recommendation

The Twilight Saga: Breaking Dawn - Part 2

Ghostbusters

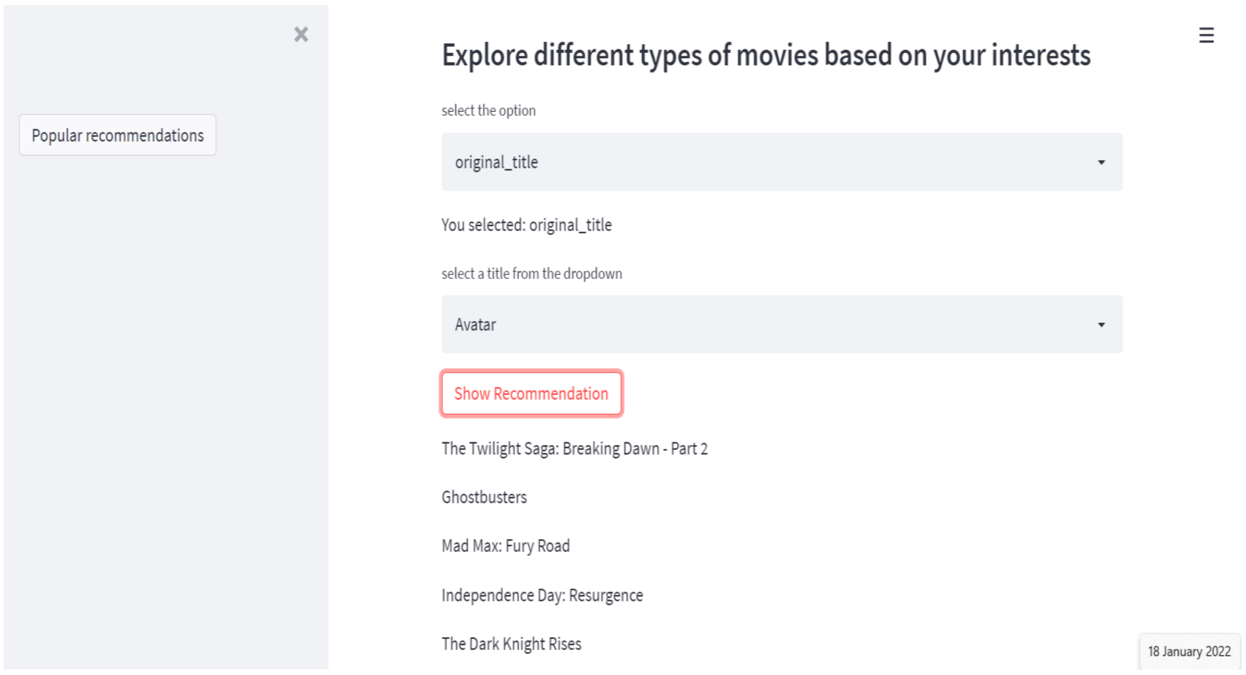
Mad Max: Fury Road

Independence Day: Resurgence

The Dark Knight Rises

18 January 2022

CAST:


Popular recommendations

Explore different types of movies based on your interests

select the option

original_title

You selected: original_title

select a title from the dropdown

Avatar

Show Recommendation

The Twilight Saga: Breaking Dawn - Part 2

Ghostbusters

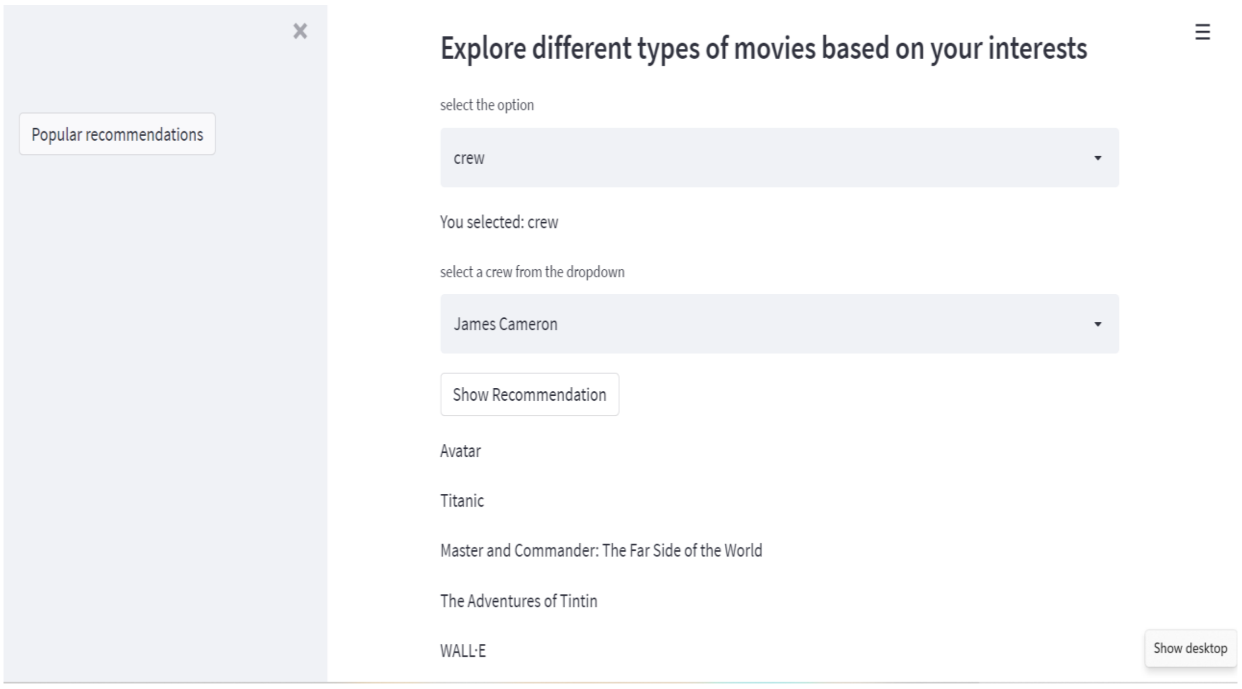
Mad Max: Fury Road

Independence Day: Resurgence

The Dark Knight Rises

18 January 2022

CREW:



Explore different types of movies based on your interests

select the option

crew

You selected: crew

select a crew from the dropdown

James Cameron

Show Recommendation

Avatar

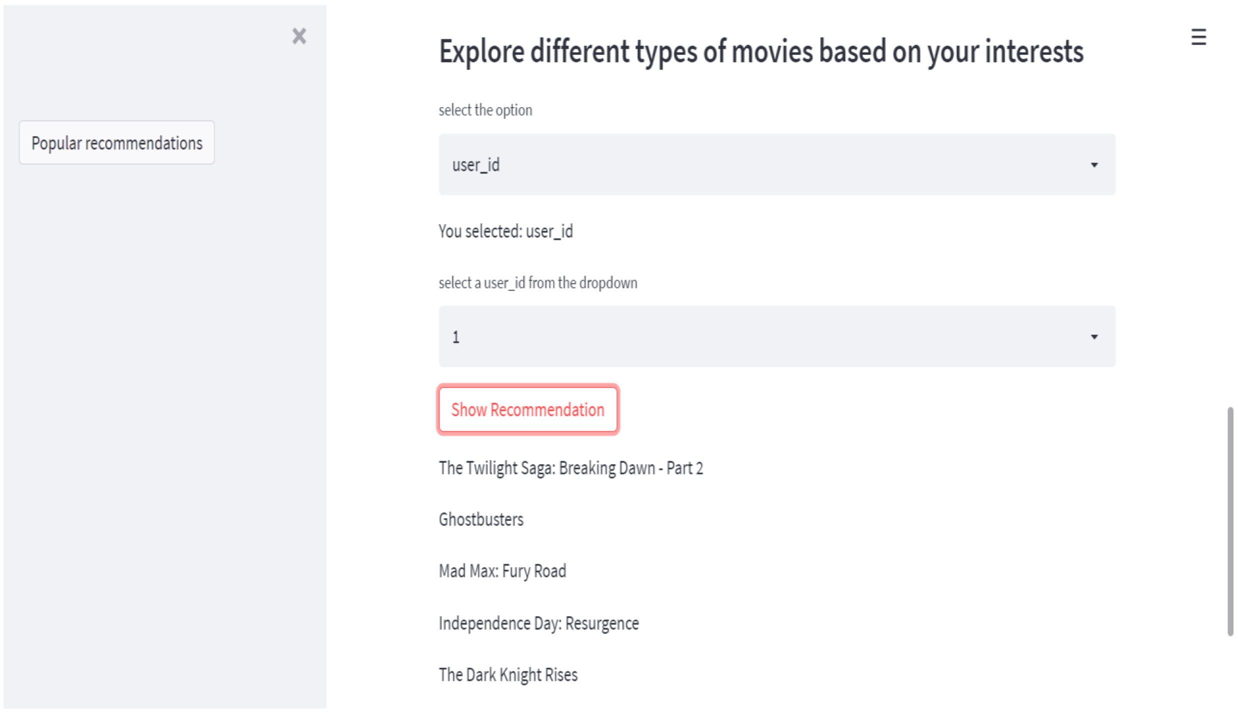
Titanic

Master and Commander: The Far Side of the World

The Adventures of Tintin

WALL·E

USER_ID:



Explore different types of movies based on your interests

select the option

user_id

You selected: user_id

select a user_id from the dropdown

1

Show Recommendation

The Twilight Saga: Breaking Dawn - Part 2

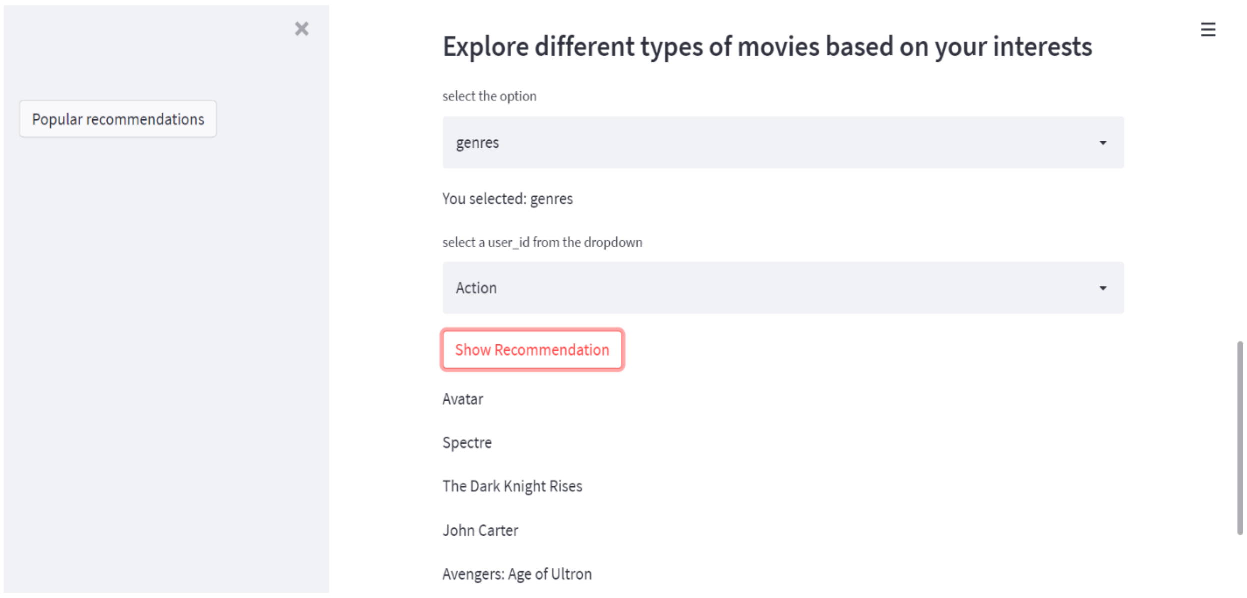
Ghostbusters

Mad Max: Fury Road

Independence Day: Resurgence

The Dark Knight Rises

GENRES:



Explore different types of movies based on your interests

select the option

genres

You selected: genres

select a user_id from the dropdown

Action

Avatar

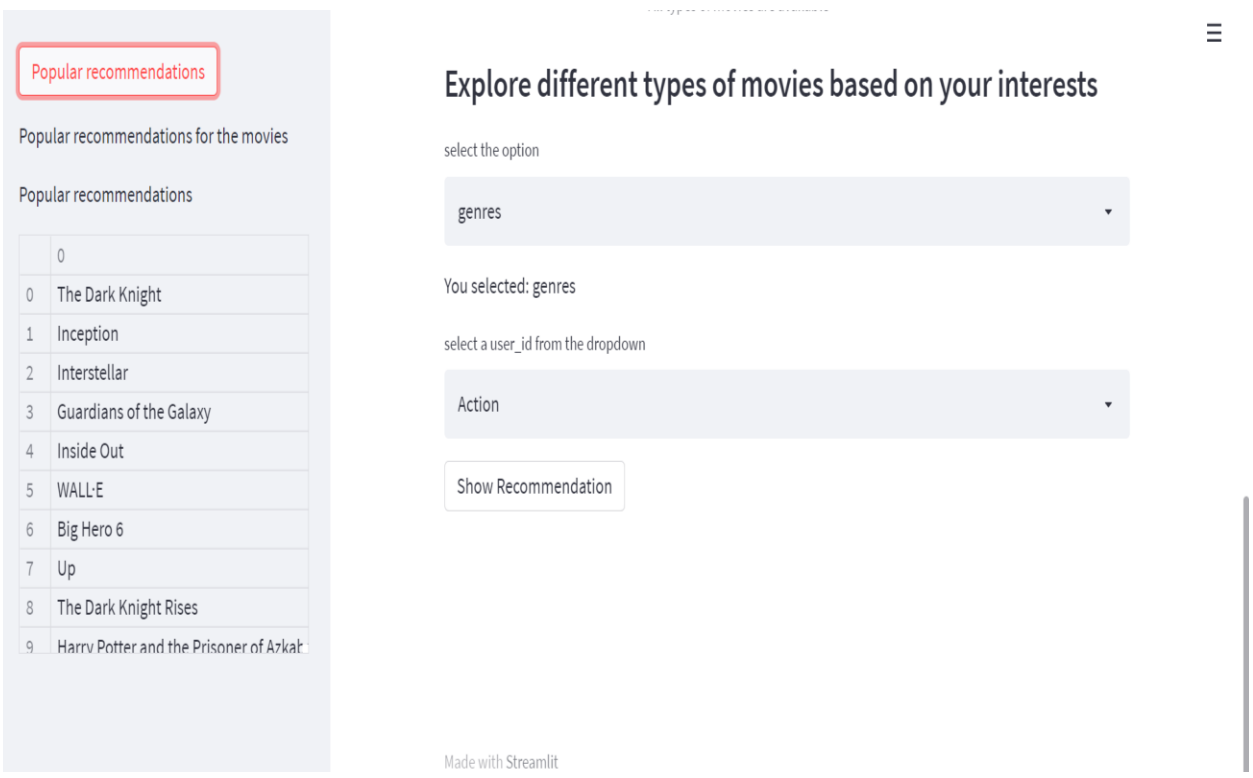
Spectre

The Dark Knight Rises

John Carter

Avengers: Age of Ultron

POPULAR RECOMMENDATIONS:



Popular recommendations for the movies

Popular recommendations

	0
0	The Dark Knight
1	Inception
2	Interstellar
3	Guardians of the Galaxy
4	Inside Out
5	WALL·E
6	Big Hero 6
7	Up
8	The Dark Knight Rises
9	Harry Potter and the Prisoner of Azkab

Explore different types of movies based on your interests

select the option

genres

You selected: genres

select a user_id from the dropdown

Action

8.TESTING

8.1 INTRODUCTION OF TESTING

SOFTWARE TESTING is defined as an activity to check whether the actual results match the expected results and to ensure that the software system is Defect free. It involves the execution of a software component or system component to evaluate one or more properties of interest. It is required for evaluating the system .This phase is the critical phase of software quality assurance and presents the ultimate view of coding.

Importance of Testing

The importance of software testing is imperative. A lot of times this process is skipped, therefore, the product and business might suffer. To understand the importance of testing, here are some key points to explain

- ➢ Software Testing saves money
- ➢ Provides Security
- ➢ Improves Product Quality
- ➢ Customer satisfaction

Testing is of different ways The main idea behind the testing is to reduce the errors and do it with a minimum time and effort.

Benefits of Testing

- **Cost-Effective:** It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
- **Security:** It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.
- **Product quality:** It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.
- **Customer Satisfaction:** The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

Different types of Testing

Unit Testing: Unit tests are very low level, close to the source of your application. They consist in testing individual methods and functions of the classes, components or modules used by your software. Unit tests are in general quite cheap to automate and can be run very quickly by a Continuous integration server.

Integration Testing: Integration tests verify that different modules or services used by your application work well together. For example, it can be testing the interaction with the database or making sure that microservices work together as expected. These types of tests are more expensive to run as they require multiple parts of the application to be up and running.

Functional Tests: Functional tests focus on the business requirements of an application. They only verify the output of an action and do not check the intermediate states of the system when performing that action.

There is sometimes a confusion between integration tests and functional tests as they both require multiple components to interact with each other. The difference is that an integration test may simply verify that you can query the database while a functional test would expect to get a specific value from the database as defined by the product requirements.

Regression Testing: Regression testing is a crucial stage for the product & very useful for the developers to identify the stability of the product with the changing requirements. Regression testing is a testing that is done to verify that a code change in the software does not impact the existing functionality of the product.

System Testing: System testing of software or hardware is testing conducted on a complete integrated system to evaluate the system's compliance with its specified requirements. System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system.

Performance Testing: It checks the speed, response time, reliability, resource usage, scalability of a software program under their expected workload. The purpose of Performance Testing is not to find functional defects but to eliminate performance bottlenecks in the software or device.

Alpha Testing: This is a form of internal acceptance testing performed mainly by the in-house software QA and testing teams. Alpha testing is the last testing done by the test teams at the development site after the acceptance testing and before releasing the software for the beta test. It can also be done by the potential users or customers of the application.
But still, this is a form of in-house acceptance testing.

Beta Testing: This is a testing stage followed by the internal full alpha test cycle. This is the final testing phase where the companies release the software to a few external user groups outside the company test teams or employees. This initial software version is known as the beta version. Most companies gather user feedback in this release.

Black Box Testing: It is also known as Behavioural Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is **not** known to the tester. These tests can be functional or non-functional, though usually functional.

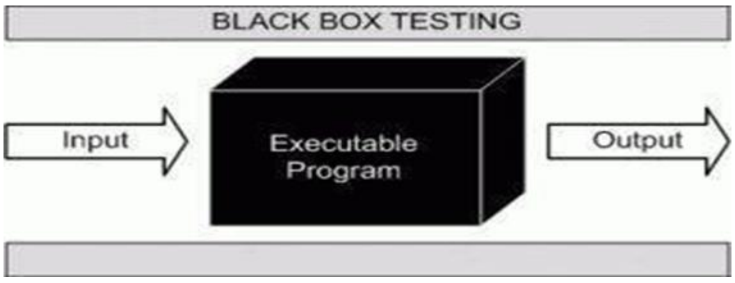


Figure 8.1.1 Black Box testing

This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

➢ Incorrect or missing functions
➢ Interface errors
➢ Errors in data structures or external database access
➢ Behaviour or performance errors ➢ Initialization and termination errors

White Box Testing: White box testing (also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing) is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the nitty-gritty of a system. This method is named so because the software program, in the eyes of the tester, is like a white/transparent box; inside which one clearly sees.

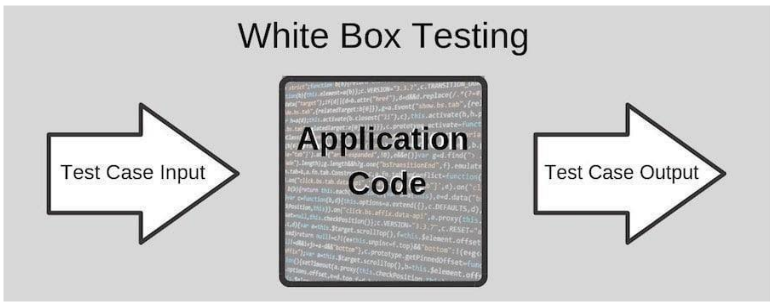


Figure 2 White Box testing

8.2 TEST CODE:

```
import streamlit as st
import pickle
import pandas as pd
import numpy as np

st.title("Movie Recommender system")
st.subheader("Your choices matters ")
st.write("""
        ABOUT US\n
            You are at the right place!Here you can fill all your movie watching dreams!!!\n
                Online movie recommendations offers many types of movies together\n
                to fulfill readers requirements based on their interests.\n
            This system also provide popular recommendations so user can get right popular
choices.
      """)

from PIL import Image

image=Image.open("img_1.png")
st.image(image,caption="All types of movies are available")

movies_list=pickle.load(open('movies.pkl','rb'))
movies=pd.DataFrame(movies_list)
similarity=pickle.load(open('similarity.pkl','rb'))

title_list=movies_list['original_title'].values
cast_list=movies_list['cast'].values
crew_list=movies_list['crew'].values
user_list=movies_list['user_id'].values
genre_list=movies_list['genres'].values

st.subheader("""  Explore different types of movies based on your interests  """)
option=st.selectbox('select the option',('original_title','cast','crew','user_id','genres'))

def unique(genre_list):
    unique_list=[]
    for x in genre_list:
        if x not in unique_list:
            unique_list.append(x)
    return unique_list

def icst(the_list, substring):
    for i, s in enumerate(the_list):
        if substring in s:
            return i
    return -1

movies['user_id']=movies['user_id'].astype(str)
```

```
recommend(movie):
    if option == 'cast':
        recommended_movie_names = []
        dex = []
        for i, s in enumerate(movies['cast']):
            if movie in s:
                cast_index = i
                m = (movies.iloc[i].original_title)
                dex.append(m)
        n = len(dex[0:5])
        idex = '\n'.join(dex[0:5])
        distances = similarity[cast_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6 -
n]
        for i in movies_list:
            idex = idex + '\n' + (movies.iloc[i[0]].original_title)
        recommended_movie_names = idex.split('\n')
        return recommended_movie_names

elif option=='crew':
        recommended_movie_names = []
        dex = []
        for i, s in enumerate(movies['crew']):
            if movie in s:
                crew_index = i
                m = (movies.iloc[i].original_title)
                dex.append(m)
        n = len(dex[0:5])
        idex = '\n'.join(dex[0:5])
        distances = similarity[crew_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6 -
n]
        for i in movies_list:
            idex = idex + '\n' + (movies.iloc[i[0]].original_title)
        recommended_movie_names = idex.split('\n')
        return recommended_movie_names

  elif option == 'original_title':
        movie_index = movies[movies['original_title'] == movie].index[0]
        distances = similarity[movie_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6]
        recommended_movies= []
        for i in movies_list:
            recommended_movies.append(movies.iloc[i[0]].original_title)
        return recommended_movies
```

```
elif option=='user_id':
        user_index = icst(movies['user_id'], movie)
        distances = similarity[user_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6]
        recommended_movies = []
        for i in movies_list:
            recommended_movies.append(movies.iloc[i[0]].original_title)
        return recommended_movies

 elif option=='genres':
        recommended_movie_names = []
        dex = []
        for i, s in enumerate(movies['genres']):
            if movie in s:
                genres_index = i
                m = (movies.iloc[i].original_title)
                dex.append(m)
        n = len(dex[0:5])
        idex = '\n'.join(dex[0:5])
        distances = similarity[genres_index]
        movies_list = sorted(list(enumerate(distances)), reverse=True, key=lambda x: x[1])[1:6 -
n]
        for i in movies_list:
            idex = idex + '\n' + (movies.iloc[i[0]].original_title)
        recommended_movie_names = idex.split('\n')
        return recommended_movie_names

st.write('You selected:', option)
if(option=='original_title'):
    selected_choice = st.selectbox("select a title from the dropdown",title_list)
if(option=='cast'):
    selected_choice = st.selectbox("select a cast from the dropdown",cast_list)
if(option=='crew'):
    selected_choice = st.selectbox("select a crew from the dropdown",crew_list)
if(option=='user_id'):
    selected_choice=st.selectbox("select a user_id from the dropdown",user_list)
if(option=='genres'):
    selected_choice=st.selectbox("select a user_id from the dropdown",unique(genre_list))
if st.button('Show Recommendation'):
    recommended_movie_names = recommend(selected_choice)
    for i in recommended_movie_names:
        st.write(i)

if st.sidebar.button("Popular recommendations "):
    st.sidebar.write("Popular recommendations for the movies ")
    popular_movies_list = pickle.load(open('popular.pkl', 'rb'))
    popular_movies = pd.DataFrame(popular_movies_list)
    x = popular_movies_list["original_title"].values
    st.sidebar.write("Popular recommendations")
    st.sidebar.write(x)
```

8.3 TEST CASES:

Input-1	Input-2	Predicted Output	Actual Output
Original_title	Avatar	1.The Twilight Saga: Breaking Dawn - Part 2 2.Ghostbusters 3.Mad Max: Fury Road 4.Independence Day: Resurgence 5.The Dark Knight Rises	1.The Twilight Saga: Breaking Dawn - Part 2 2.Ghostbusters 3.Mad Max: Fury Road 4.Independence Day: Resurgence 5.The Dark Knight Rises
Cast	Johnny Depp	1.Pirates of the Caribbean: At World's End 2.Pirates of the Caribbean: Dead Man's Chest 3.The Lone Ranger 4. Pirates of the Caribbean: On Stranger Tides 5.Alice Through the Looking Glass	1.Pirates of the Caribbean: At World's End 2.Pirates of the Caribbean: Dead Man's Chest 3.The Lone Ranger 4. Pirates of the Caribbean: On Stranger Tides 5.Alice Through the Looking Glass

Crew	James Cameron	1.Avatar 2.Titanic 3.Master and Commander: The Far Side of the World 4.The Adventures of Tintin 5.WALL·E	1.Avatar 2.Titanic 3.Master and Commander: The Far Side of the World 4.The Adventures of Tintin 5.WALL·E
User_id	1	1.The Twilight Saga: Breaking Dawn - Part 2 2.Ghostbusters 3.Mad Max: Fury Road 4.Independence Day: Resurgence 5.The Dark Knight Rises	1.The Twilight Saga: Breaking Dawn - Part 2 2.Ghostbusters 3.Mad Max: Fury Road 4.Independence Day: Resurgence 5.The Dark Knight Rises
genres	Action	1.Avatar 2.Spectre 3.The Dark Knight Rises 4.John Carter 5.Avengers: Age of Ultron	1.Avatar 2.Spectre 3.The Dark Knight Rises 4.John Carter 5.Avengers: Age of Ultron

Popular movies	--------	1.The Dark Knight 2.Inception 3.Interstellar 4.Guardians of the Galaxy 5.Inside Out 6.WALL·E 7. Big Hero 6 8.Up 9. The Dark Knight Rises 10. Harry Potter and the Prisoner of Azkaban	1.The Dark Knight 2.Inception 3.Interstellar 4.Guardians of the Galaxy 5.Inside Out 6.WALL·E 7. Big Hero 6 8.Up 9. The Dark Knight Rises 10. Harry Potter and the Prisoner of Azkaban

9.CONCLUSION

Recommender system has become more and more important because of the information overload. Content-based movie recommender systems which are existing have poor efficiency due which movies are suggested to the users only based on the users past or previous history. We have developed content-based movie recommender system in which the user can get the recommendations based on his/her past or previous history, favourite cast, favourite crew, favourite genres. We have also recommended the top 10 popular/trending movies to the users by using popular-based recommender system.

In this project, to recommend the movies to the users based on the inputs given by the user, we have used cosine similarity algorithm.

In this project, we have developed a movie recommender system, which is a combination of both content-based and popular-based, with good accuracy and efficiency.

10. FUTURE SCOPE

- Future scope of this project will involve adding more attributes and factors of the both user profile and the item profile etc. The more the parameters are taken into account more will be the accuracy.
- The use of traditional recommender algorithms and data mining techniques can also help to predict the movie to the user.
- In the future, we plan to add more parameters and create a web application. So that user may register into it and get recommendations of the movies.

11. REFERENCES

[1] Choi, Sang-Min, Sang-Ki Ko, and Yo-Sub Han. "A movie recommendation algorithm based on genre correlations." Expert Systems with Applications 39.9 (2012): 8079-8085.

[2] Lekakos, George, and Petros Caravelas. "A hybrid approach for movie recommendation." Multimedia tools and applications 36.1 (2008): 55-70

[3] Das, Debashis, Laxman Sahoo, and Sujoy Datta. "A survey on recommendation system." International Journal of Computer Applications 160.7 (2017).

[4] Zhang, Jiang, et al. "Personalized real-time movie recommendation system: Practical prototype and evaluation." Tsinghua Science and Technology 25.2 (2019): 180-191.

[5] Rajarajeswari, S., et al. "Movie Recommendation System." Emerging Research in Computing, Information, Communication and Applications. Springer, Singapore, 2019. 329-340.

[6] Arora, Gaurav, et al. "Movie recommendation system based on users' similarity." International Journal of Computer Science and Mobile Computing 3.4 (2014): 765-770.

[7] R. Lavanya, U. Singh and V. Tyagi, "A Comprehensive Survey on Movie Recommendation Systems," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021, pp. 532-536, doi:10.1109/ICAIS50930.2021.9395759.

[8] Subramaniyaswamy, V., et al. "A personalised movie recommendation system based on collaborative filtering." International Journal of High Performance Computing and Networking 10.1-2 (2017): 54-63.

[9] Harper, F. Maxwell, and Joseph A. Konstan. "The movielens datasets: History and context." Acm transactions on interactive intelligent systems (tiis) 5.4 (2015): 1-19.