**CHAPTER 1**

**INTRODUCTION**

* 1. **Relevance of the Project**

A recommendation system or recommendation engine is a model used for information filtering where it tries to predict the preferences of a user and provide suggests based on these preferences. These systems have become increasingly popular nowadays and are widely used today in areas such as movies, music, books, videos, clothing, restaurants, food, places and other utilities. These systems collect information about a user's preferences and behaviour, and then use this information to improve their suggestions in the future.

Movies are a part and parcel of life. There are different types of movies like some for entertainment, some for educational purposes, some are animated movies for children, and some are horror movies or action films. Movies can be easily differentiated through their genres like comedy, thriller, animation, action etc. Other way to distinguish among movies can be either by releasing year, language, director etc. Watching movies online, there are a number of movies to search in our most liked movies. Movie Recommendation Systems helps us to search our preferred movies among all of these different types of movies and hence reduce the trouble of spending a lot of time searching our favourable movies. So, it requires that the movie recommendation system should be very reliable and should provide us with the recommendation of movies which are exactly same or most matched with our preferences.

A large number of companies are making use of recommendation systems to increase user interaction and enrich a user's shopping experience. Recommendation systems have several benefits, the most important being customer satisfaction and revenue. Movie Recommendation system is very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffer with poor recommendation quality and scalability issues.

* 1. **Problem Statement:**

The goal of the project is to recommend a movie to the user. Providing related content out of relevant and irrelevant collection of items to users of online service providers.

* 1. **Objective of the Projects**

• Improving the Accuracy of the recommendation system

• Improve the Quality of the movie Recommendation system

• Improving the Scalability.

• Enhancing the user experience.

* 1. **Scope of the Project**

The objective of this project is to provide accurate movie recommendations to users. The goal of the project is to improve the quality of movie recommendation system, such as accuracy, quality and scalability of system than the pure approaches. This is done using a simple approach by content based to eradicate the overload of the data, recommendation system is used as information filtering tool in social networking sites.

Hence, there is a huge scope of exploration in this field for improving scalability, accuracy and quality of movie recommendation systems Movie Recommendation system is very powerful and important system. But, due to the problems associated with pure collaborative approach, movie recommendation systems also suffer with poor recommendation quality and scalability issues.

* 1. **Methodology for Movie Recommendation**

The approach proposed an integrative method by using Cosine Similarity of KNN and genetic algorithm based weighted similarity measure to construct a movie recommendation system. The proposed movie recommendation system gives finer similarity metrics and quality than the existing Movie recommendation system but the computation time which is taken by the proposed recommendation system is more than the existing recommendation system.

This problem can be fixed by taking the clustered data points as an input dataset the proposed approach is for improving the scalability and quality of the movie recommendation system. We use a simple approach, by unifying Content-Based Filtering, so that the approach can be profited. For computing similarity between the different movies in the given dataset efficiently and in least time and to reduce computation time of the movie recommender engine we used cosine similarity measure.

* 1. **Agile Methodology:**

**1.collecting the data sets:** Collecting all the required data set from Kaggle web site.in this project we require tmdb\_5000\_movies.csv, tmdb\_5000\_credits.csv.

**2.Data Analysis:** Make sure that that the collected data sets are correct and analysing the data in the csv files. i.e. checking whether all the column Felds are present in the data sets.

**3.Algorithms:** In our project we have only one algorithm i.e. cosine similarity used to build the machine learning recommendation model.

**4.Training and Testing the model:** Once the implementation of algorithm is completed. we have to train the model to get the result. We have tested it several times the model is recommend different set of movies to different users. Then we built the web-based application in ‘streamlit’ framework of python.

**5.Improvements in the project:** In the later stage we can implement different algorithms and methods for better recommendation.

**CHAPTER 2**

**LITERATURE SURVEY**

Over the years, many recommendation systems have been developed using either collaborative, content based or hybrid filtering methods. These systems have been implemented using various big data and machine learning algorithms.

**2.1 Movie Recommendation System by K-Nearest Neighbour:**

A recommendation system collect data about the user’s preferences either implicitly or explicitly on different items like movies. An implicit acquisition in the development of movie recommendation system uses the user’s behaviour while watching the movies. On the other hand, an explicit acquisition in the development of movie recommendation system uses the user’s previous ratings or history. The other supporting technique that are used in the development of recommendation system is clustering. Clustering is a process to group a set of objects in such a way that objects in the same clusters are more similar to each other than to those in other clusters.

K-Nearest Neighbour is implemented on the movie lens dataset in order to obtain the best-optimized result. In existing technique, the data is scattered which results in a high number of clusters while in the proposed technique data is gathered and results in a low number of clusters. The process of recommendation of a movie is optimized in the proposed scheme. The proposed recommender system predicts the user’s preference of a movie on the basis of different parameters. The recommender system works on the concept that people are having common preference or choice. These users will influence on each other’s opinions.

**2.2 Movie Recommendation System Using Content-based Filtering:**

It uses attributes such as genre, director, description, actors, etc. for movies, to make suggestions for the users. The intuition behind this sort of recommendation system is that if a user liked a particular movie or show, he/she might like a movie or a show similar to it.

**CHAPTER 3**

**SYSTEM REQUIREMENTS SPECIFICATION**

This chapter involves both the hardware and software requirements needed for the project and detailed explanation of the specifications.

**3.1 Hardware Requirements**

• A PC with Windows/Linux OS

• Processor with 1.7-2.4gHz speed

• Minimum of 8gb RAM

• 2gb Graphic card

**3.2 Software Specification**

• Text Editor (VS-code/Jupyter Notebook)

• Anaconda distribution package

• Python libraries

**3.3 Software Requirements**

**3.3.1 Anaconda distribution:**

Anaconda is a free and open-source distribution of the Python programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management system and deployment. Package versions are managed by the package management system conda. The anaconda distribution includes data-science packages suitable for Windows, Linux and MacOS.3

**3.3.2 Python libraries:**

For the computation and analysis, we need certain python libraries which are used to perform analytics. Packages such as SKlearn, Numpy, pandas, Streamlit framework, etc are needed.

**SKlearn:** It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

**NumPy:** NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Pandas: Pandas is one of the most widely used python libraries in data science. It provides high-performance, easy to use structures and data analysis tools. Unlike NumPy library which provides objects for multi-dimensional arrays, Pandas provides in-memory 2d table object called Data frame.

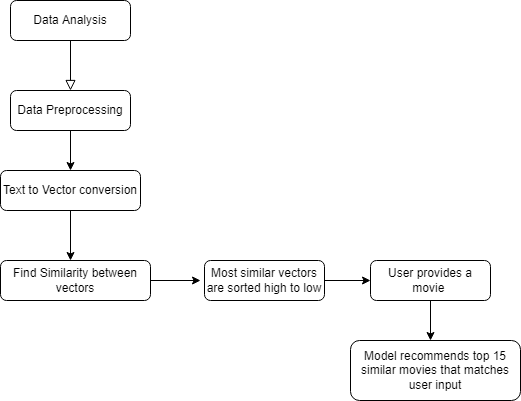
**Pandas:** Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package named [NumPy](https://www.activestate.com/products/python/python-packages/), which provides support for multi-dimensional arrays. As one of the most popular data wrangling packages, Pandas works well with many other [data science](https://www.activestate.com/products/python/python-data-science/) modules inside the Python ecosystem, and is typically included in every Python distribution, from those that come with your operating system to commercial vendor distributions like ActiveState’s [ActivePython](https://platform.activestate.com/featured-projects).

**Streamlit:** Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. We can instantly develop web apps and deploy them easily using Streamlit. 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.

**CHAPTER 4**

**SYSTEM ANALYSIS AND DESIGN**

**4.1 System Architecture of Proposed System:**

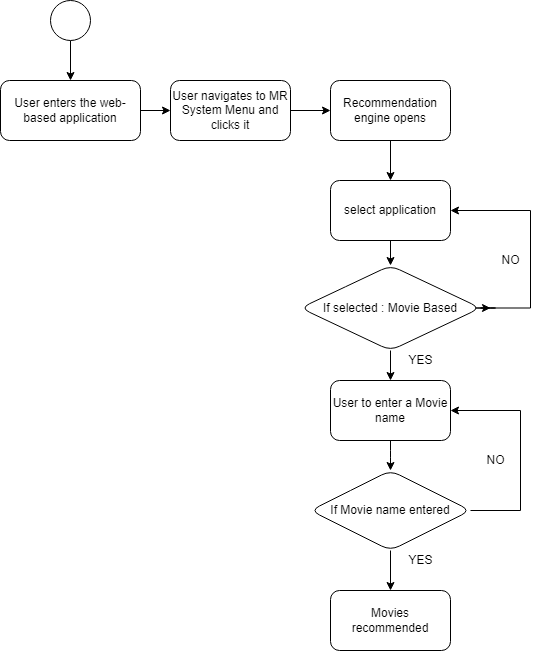
****

**Fig.1 System Architecture**

The ‘TMDB 5000 Movie Dataset’ is taken into consideration for movie recommendation purpose in this research work. This dataset is available on kaggle.com. The dataset is composed of 2 CSV files - ‘tmdb\_5000\_movies.csv’ and ‘tmdb\_5000\_credits.csv’.

For each different individual use different list of movies are recommended, as user provides a movie name to the recommendation system, it recommends the list of movies with its poster which are most similar to the movie name given as input. It finds the similarity with the rest 4999 movies from the csv file and generates the output.

**4.2 Activity Diagram:**

****

**Fig- 2. User activity**

User enters the web-based application and selects the application and provides a name of a movie and the recommendation engine recommends top 15 similar movies based on cast, director and genres.

**4.3 Control Flow Diagram:**

****

**Fig-3. Control Flow Diagram**

Initially load the data sets that are required to build a model the data set that are required in this project are ‘tmdb\_5000\_movies.csv’ and ‘tmdb\_5000\_credits.csv’, all the data sets are available in the Kaggle.com. Basically, a model is built in this project content based produce a list of movies to a user by finding the similarity between rest 4,999 movies from the csv file.

**CHAPTER 5**

**IMPLEMENTATION**

The proposed system makes use of different modules, algorithm and methods for the implementation of our approach towards our project.

**5.1 Cosine Similarity:**

Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

**Formula:**

**5.2 Experimental Requirements:**

**Code: Front-end (Streamlit)**

In this project we have used popular front-end web framework of python (streamlit) to build an interactive user interface as web-based application.

import numpy as np

import streamlit as st

from streamlit\_option\_menu import option\_menu

from streamlit\_lottie import st\_lottie

import streamlit.components.v1 as html

from PIL import Image

import numpy as np

import pandas as pd

from st\_aggrid import AgGrid

import plotly.express as px

import pickle

import requests

*#---Configuring the web apps's title name and its favicon*

st.set\_page\_config(page\_title =" Movie Recommendation System",page\_icon =":computer:",layout="wide")

*#----Making the navigation menu*

with st.sidebar:

    choose =option\_menu(

        "MR System",["Home","MR System","About us","Contact us"],

        icons = ["house","cpu","people","envelope"],

        menu\_icon = "app-indicator",

        default\_index = 0,

        styles={

"container": {"padding": "5!important","background-color": "#040C6D"},

        "icon": {"color": "#FFFFFF ", "font-size": "20px"},

        "nav-link": {"font-size": "20px", "text-align": "left","color": "#FFFFFF ", "margin":"0px", "--hover-color": "#373A5B "},

        "nav-link-selected": {"background-color": "#010318"},)

def load\_lottieurl(url):

    r = requests.get(url)

    if r.status\_code != 200:

        return None

    return r.json()

*#---Loading all the iamges used here*

lottie\_coding=load\_lottieurl("https://assets3.lottiefiles.com/private\_files/lf30\_cbemdbsc.json")

lottie\_coding2=load\_lottieurl("https://assets7.lottiefiles.com/packages/lf20\_knvn3kk2.json")

logo = Image.open(r'logo2.png')

kishor = Image.open(r'kishork.png')

dhruv = Image.open(r'dhruv.png')

subhajit = Image.open(r'subhajit.png')

rimi = Image.open(r'rimi.png')

ankush = Image.open(r'ankush.png')

anna = Image.open(r'anna.png')

*#--- Making the decision if-else to navigate through diffrent menus selected*

*#--if "Home " is selected --- its operational code*

if choose == "Home":

    col1, col2 = st.columns( [0.8, 0.2])

    with col1:

*#---To display the header text using css style*

        st.markdown(""" <style> .font { font-size:45px ; font-family: 'Cooper Black'; color: #009edc;} </style> """, unsafe\_allow\_html=True)

        st.markdown('<p class="font">Movie Recommendation System</p>', unsafe\_allow\_html=True)

    with col2:

*#---To display brand logo*

        st.image(logo, width=90 )

    col1, col2 = st.columns( [0.7, 0.3])

    with col1:

        st.subheader('What is Recommendation system ?')

        st.write("A recommendation system is a subclass of Information filtering Systems that seeks to predict the rating or the preference a user might give to an item. In simple words, it is an algorithm that suggests relevant items to users. Eg: In the case of Netflix which movie to watch, In the case of e-commerce which product to buy, or In the case of kindle which book to read, etc.")

        st.subheader('Use-Cases Of Recommendation System')

        st.write("""There are many use-cases of it. Some are

A. Personalized Content:  Helps to Improve the on-site experience by creating dynamic recommendations for different kinds of audiences like Netflix does.

B. Better Product search experience:  Helps to categories the product based on their features. Eg: Material, Season, etc.""")

    with col2:

      st\_lottie(lottie\_coding, height = 350, key =" coding")

    st.header('TYPES OF RECOMMENDATION SYSTEM')

    st.subheader('1. Content-Based Filtering')

    st.write("In this type of recommendation system, relevant items are shown using the content of the previously searched items by the users. Here content refers to the attribute/tag of the product that the user like. In this type of system, products are tagged using certain keywords, then the system tries to understand what the user wants and it looks in its database and finally tries to recommend different products that the user wants.")

    st.subheader('2. Collaborative Based Filtering')

    st.write("""Recommending the new items to users based on the interest and preference of other similar users is basically collaborative-based filtering.

                This overcomes the disadvantage of content-based filtering as it will use the user Interaction instead of content from the items used by the users. For this, it only needs the historical performance of the users. Based on the historical data, with the assumption that user who has agreed in past tends to also agree in future.""")

    st.header("CONCLUSION")

    st.write("This small article covered many topics related to recommendation engines like What are it and its use-cases. Apart from this different type of recommendation systems like content-based filtering and collaborative based filtering and in collaborative filtering also user-based as well as item-based along with its examples, advantages and disadvantages, and finally the evaluation metrics to evaluate the model.")

*#--if "MR System " is selected --- its operational code[the main system of project]*

elif choose == "MR System":

    def fetch\_poster(movie\_id):

*#--- Function made to fetch the poster of movies*

     url="https://api.themoviedb.org/3/movie/{}?api\_key=8265bd1679663a7ea12ac168da84d2e8&language=en-US".format(movie\_id)

        data = requests.get(url)

        data = data.json()

        poster\_path = data['poster\_path']

        full\_path = "https://image.tmdb.org/t/p/w500/" + poster\_path

        return full\_path

    movies = pickle.load(open('movie\_dict.pkl', 'rb'))

*#--- loading the pickle format data from jupyter notebook*

    similarity = pickle.load(open('Similarity.pkl', 'rb'))

    def recommend(movie):

*#--- Function that implements KNN using cosine similarity.*

        index = movies[movies['title'] == movie].index[0]

        distances = sorted(list(enumerate(similarity[index])), reverse=True, key=lambda x: x[1])

        movie\_list = sorted(list(enumerate(distances)),

                            reverse=True, key=lambda x: x[1])[1:16]

        recommended\_movie\_names = []

        recommended\_movie\_posters = []

        for i in distances[1:16]:

*# fetch the movie poster*

            movie\_id = movies.iloc[i[0]].movie\_id

            recommended\_movie\_posters.append(fetch\_poster(movie\_id))

            recommended\_movie\_names.append(movies.iloc[i[0]].title)

        return recommended\_movie\_names, recommended\_movie\_posters

    movie\_list = movies['title'].values

*#--- Designing the header part of our function*

    col1, col2 = st.columns( [0.8, 0.2])

    with col1:               *#---To display the header text using css style*

            st.markdown("""  <style> .font { font-size:45px ; font-family: 'Cooper Black'; color: #009edc;} </style> """, unsafe\_allow\_html=True)

            st.markdown('<p class="font">Movie Recommendation System</p>', unsafe\_allow\_html=True)

    with col2:               *#---To display brand logo*

        st.image(logo, width=90 )

    apps = ['--Select--', 'Movie based']

    app\_options = st.selectbox('Select application:', apps)

    if app\_options == 'Movie based':

        selected\_movie =st.selectbox('Select movie:', movie\_list)

*#--- Recommending top 15 movies:*

    if st.button('Show Recommendation'):

        recommended\_movie\_names, recommended\_movie\_posters = recommend(selected\_movie)

        col1, col2, col3, col4, col5 = st.columns(5)

        with col1:

            st.image(recommended\_movie\_posters[0])

            st.text(recommended\_movie\_names[0])

        with col2:

            st.image(recommended\_movie\_posters[1])

            st.text(recommended\_movie\_names[1])

        with col3:

            st.image(recommended\_movie\_posters[2])

            st.text(recommended\_movie\_names[2])

        with col4:

            st.image(recommended\_movie\_posters[3])

st.text(recommended\_movie\_names[3])

        with col5:

            st.image(recommended\_movie\_posters[4])

            st.text(recommended\_movie\_names[4])

        col6, col7, col8, col9, col10 = st.columns(5)

        with col6:

            st.image(recommended\_movie\_posters[5])

            st.text(recommended\_movie\_names[5])

        with col7:

            st.image(recommended\_movie\_posters[6])

            st.text(recommended\_movie\_names[6])

        with col8:

            st.image(recommended\_movie\_posters[7])

            st.text(recommended\_movie\_names[7])

        with col9:

            st.image(recommended\_movie\_posters[8])

            st.text(recommended\_movie\_names[8])

        with col10:

            st.image(recommended\_movie\_posters[9])

            st.text(recommended\_movie\_names[9])

        col10, col11, col12, col13, col14 = st.columns(5)

        with col10:

            st.image(recommended\_movie\_posters[10])

            st.text(recommended\_movie\_names[10])

        with col11:

            st.image(recommended\_movie\_posters[11])

            st.text(recommended\_movie\_names[11])

        with col12:

            st.image(recommended\_movie\_posters[12])

            st.text(recommended\_movie\_names[12])

        with col13:

            st.image(recommended\_movie\_posters[13])

            st.text(recommended\_movie\_names[13])

        with col14:

            st.image(recommended\_movie\_posters[14])

            st.text(recommended\_movie\_names[14])

elif choose == "About us":

    with st.container():

        st.markdown(""" <style> .font { font-size:45px ; font-family: 'Cooper Black'; color: #009edc;} </style> """, unsafe\_allow\_html=True)

        st.markdown('<p class="font">About us and our project.</p>', unsafe\_allow\_html=True)

        st.write("##")

        col1, col2 = st.columns([0.7,0.3])

        with col1:

            st.write("I am Kishor Kumar and along with my team-mates, we have developed this project 'Movie Recommendation System'. It is a machine learning project developed in python language using a web application framework called 'Streamlit'.")

            st.write("#")

            st.write("We have provided all the details below:")

            st.write("NAME :  Movie Recommendation System")

            st.write("TECHNOLOGY :  Machine Learning")

            st.write("LANGUAGE USED :  Python, HTML and CSS")

            st.write("IDE :  Jupyter Notebook and VS Code")

            st.write("FRAMEWORK :  Streamlit [Python's Web-based application development framework] & Elements of Bootstrap")

            st.write("##")

        with col2:

            st\_lottie(lottie\_coding2, height = 300, key =" coding")

        st.write("---")

        st.subheader("TEAM MEMBERS")

        st.write("---")

        imageclm , textclm = st.columns([0.3,0.7])

        with imageclm:

            st.image(kishor, width = 150)

        with textclm:

            st.subheader("KISHOR KUMAR")

            st.write("Coding & UI Designing")

            st.write("##")

        st.write("#")

        st.write("---")

        imageclm , textclm = st.columns([0.3,0.7])

        with imageclm:

            st.image(dhruv, width = 150)

        with textclm:

            st.subheader("DHRUBAJIT GOPE")

            st.write("Coding & Problem Solving")

            st.write("##")

        st.write("#")

        st.write("---")

        imageclm , textclm = st.columns([0.3,0.7])

        with textclm:

            st.subheader("RIMI MONDAL")

            st.write("Documentation")

            st.write("#")

        with imageclm:

            st.image(rimi, width = 150)

        st.write("#")

        st.write("---")

        imageclm , textclm = st.columns([0.3,0.7])

        with textclm:

            st.subheader("ANANYA MUKHERJEE")

            st.write("Documentation")

            st.write("#")

        with imageclm:

            st.image(anna, width = 150)

        st.write("#")

        st.write("---")

        imageclm , textclm = st.columns([0.3,0.7])

        with textclm:

            st.subheader("ANKUSH PAUL")

            st.write("Analysis & Testing")

            st.write("#")

        with imageclm:

            st.image(ankush, width = 150)

        st.write("#")

        st.write("---")

        imageclm , textclm = st.columns([0.3,0.7])

        with textclm:

            st.subheader("SUBAJIT CHAKRABORTY ")

            st.write("Analysis & Testing")

            st.write("##")

        with imageclm:

            st.image(subhajit, width = 150)

*#---If "Contact us" selected, it will redirect to contact form page.*

elif choose == "Contact us":

    st.markdown(""" <style> .font {

    font-size:35px ; font-family: 'Cooper Black'; color: #009edc;}

    </style> """, unsafe\_allow\_html=True)

    st.markdown('<p class="font">Contact Form</p>', unsafe\_allow\_html=True)

    st.subheader(":mailbox: Get in touch with us!")

    with st.form(key='columns\_in\_form2',clear\_on\_submit=True): *#set clear\_on\_submit=True so that the form will be reset/cleared once it's submitted*

*#st.write('Please help us improve!')*

        Name=st.text\_input(label='Please Enter Your Name') *#Collect user feedback*

        Email=st.text\_input(label='Please Enter Email') *#Collect user feedback*

        Message=st.text\_input(label='Please Enter Your Message') *#Collect user feedback*

        submitted = st.form\_submit\_button('Submit')

        if submitted:

            st.write('Thanks for your contacting us. We will respond to your questions or inquiries as soon as possible!')

**VS CODE Snippet**

In Visual Studio code, we have developed our front-end part using python web-based framework **Streamlit.** This helped us to design a better UI for the user. We imported the pickle file from our jupyter notebook and used here as a raw data. We used TMDB API Key to fetch posters of the similar movies that are going to be recommended. We designed our project as a website which consist mainly four web

pages, **Home, MR System, About Us** and **Contact Us.** Whenever the user will click on any of the web page it will redirect to that certain web page. MR System consist our actual project i.e. Recommendation System.

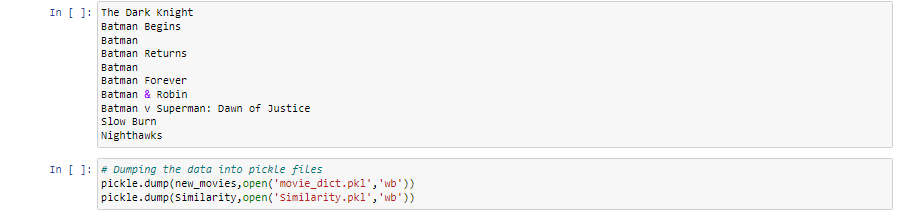
**Code: Back-end (Anaconda Jupyter Notebook)**

For backend we have use anaconda Jupyter Notebook for data cleaning, data pre-processing, analysis and model building.









**Fig- 4 Jupyter Code**

**CHAPTER 6**

**RESULT AND DISCUSSION**

Since our project is movie recommendation system one can develop a movie recommendation system by using either content based or collaborative filtering or combining both.

In our project we have developed a simple approach i.e. content filtering. The approache has advantages and dis-advantages. In content-based filtering the it based on the user ratings or user likes only such kind of movie will recommended to the user.

**Advantages:** it is easy to design and it takes less time to compute

**Disadvantages:** the model can only make recommendations based on existing interests of the user. In other words, the model has limited ability to expand on the users' existing interests.

In Collaborative filtering the recommendation is comparison of similar users.

**Advantages:** No need domain knowledge because the embeddings are automatically learned. The model can help users discover new interests. In isolation, the ML system may not know the user is interested in a given item, but the model might still recommend it because similar users are interested in that item.

**Disadvantages:** The prediction of the model for a given (user, item) pair is the dot product of the corresponding embeddings. So, if an item is not seen during training, the system can't create an embedding for it and can't query the model with this item. This issue is often called the cold-start problem.

**Screenshot of the result:**

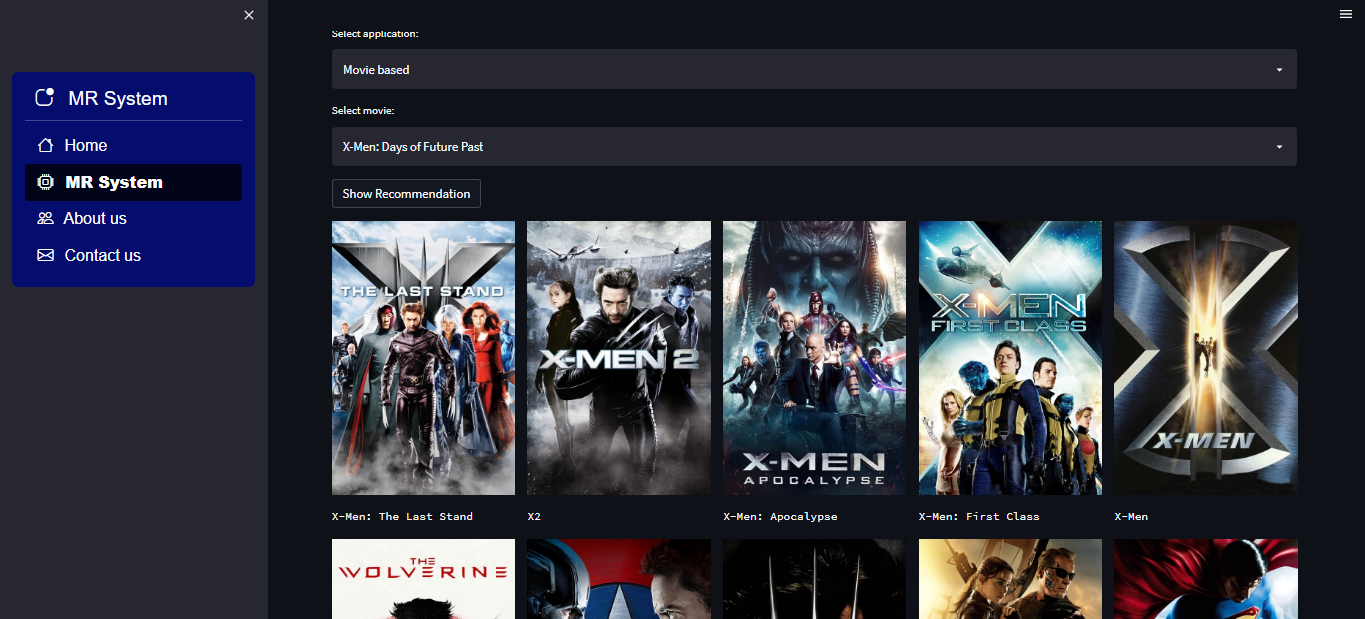
**Home Page**

****

**Fig – 5 Screenshot 1**

Some details of recommendation system, machine learning depicted in the home page of our web-based application.

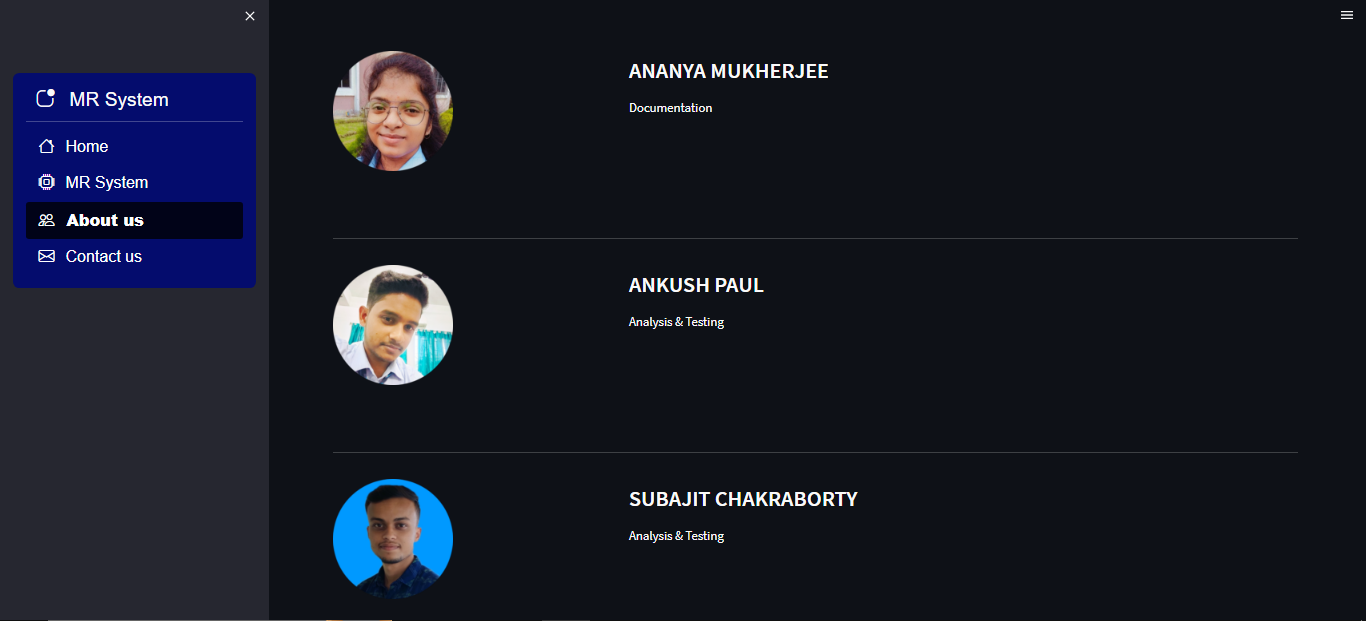
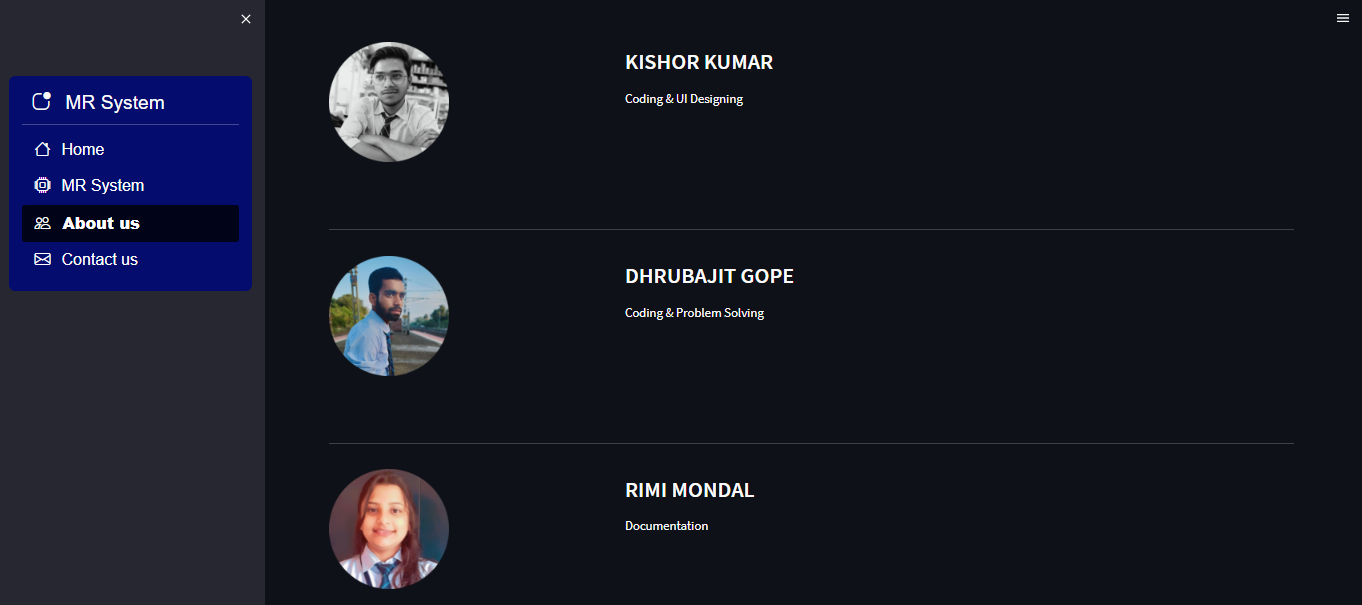
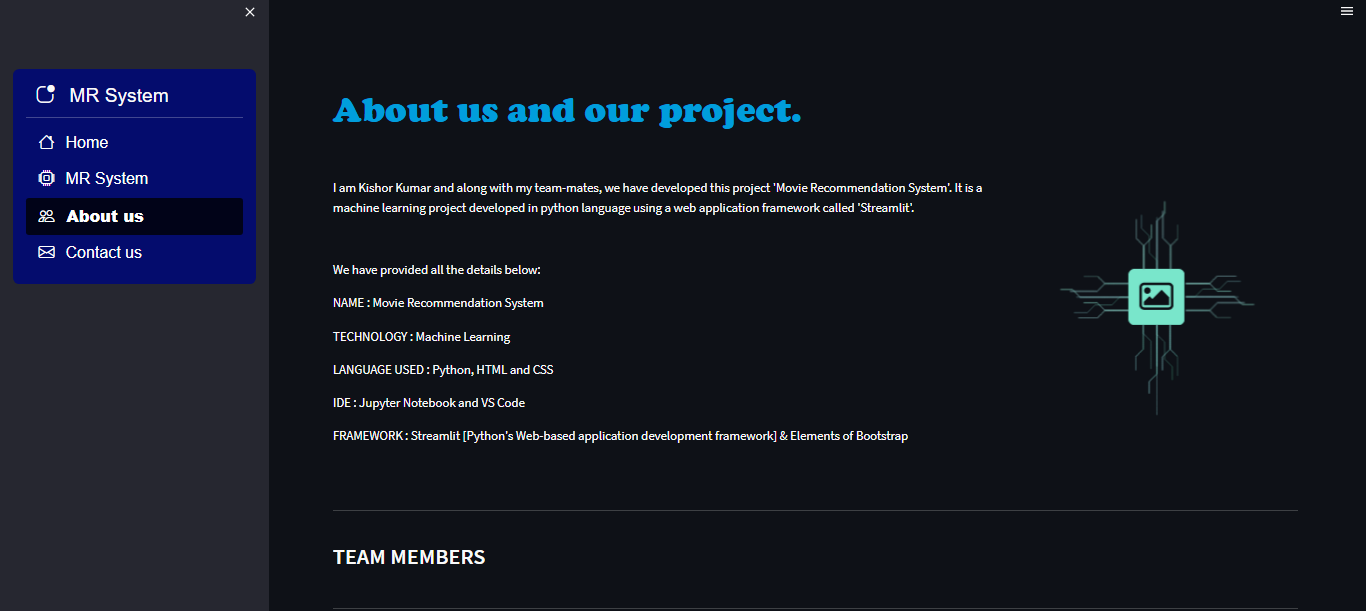
**MR System**

****

**Fig -6 Screenshot 2**

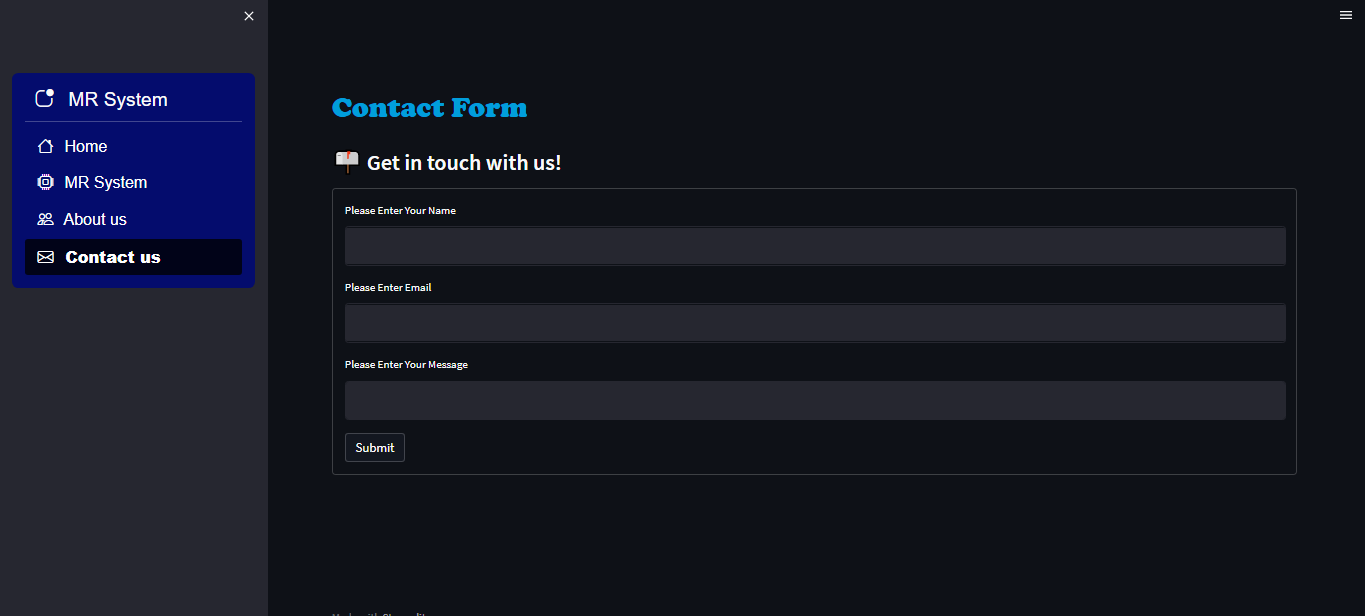
Main Recommendation System page, consist a form for user input. User provides a movie name and it shows top 15 similar movies.

**About US**

****

**Fig – 7 Screenshot 3**

**Contact US**

****

**Fig – 8 Screenshot 4**

A contact us form for user interaction with the developer team.

**CHAPTER 7**

**TESTING**

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all the system elements have been properly integrated and perform allocated functions. The testing process is actually carried out to make sure that the product exactly does the same thing what is supposed to do. In the testing stage following goals are tried to achieve: -

● To affirm the quality of the project.

● To find and eliminate any residual errors from previous stages.

● To validate the software as a solution to the original problem.

● To provide operational reliability of the system.

**7.1 Testing Methodologies**

There are many different types of testing methods or techniques used as part of the software testing methodology. Some of the important testing methodologies are:

**Unit Testing**

Unit testing is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a test-driven environment will typically write and run the tests prior to the software or feature being passed over to the test team. Unit testing can be conducted manually, but automating the process will speed up delivery cycles and expand test coverage. Unit testing will also make debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process. Test Left is a tool that allows advanced testers and developers to shift left with the fastest test automation tool embedded in any IDE.

**Integration Testing**

After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. These are then tested as group through integration testing to ensure whole segments of an application behave as expected (i.e, the interactions between units are seamless). These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests.

**System Testing**

System testing is a black box testing method used to evaluate the completed and integrated system, as a whole, to ensure it meets specified requirements. The functionality of the software is tested from end-to-end and is typically conducted by a separate testing team than the development team before the product is pushed into production.

**CHAPTER 8**

**CONCLUSION AND FUTURE SCOPE**

**8.1 Conclusion**

In this project, to improve the accuracy, quality and scalability of movie recommendation system, a Hybrid approach by unifying content-based filtering; COUNT VECTORIZER as a classifier and Cosine Similarity is presented in the proposed methodology. Existing pure approaches and proposed hybrid approach is implemented on three different Movie datasets and the results are compared among them. Comparative results depict that the proposed approach shows an improvement in the accuracy, quality and scalability of the movie recommendation system than the pure approaches. Also, computing time of the proposed approach is lesser.

**8.2 Future scope:**

In the proposed approach, it has considered Genres of movies but, in future we can also consider age of user as according to the age movie preferences also changes, like for example, during our childhood we like animated movies more as compared to other movies. There is a need to work on the memory requirements of the proposed approach in the future. The proposed approach has been implemented here on different movie datasets only. It can also be implemented on the Film Affinity and Netflix datasets and the performance can be computed in the future.

Using neural engine and algorithms of deep learning, we can calculate the accuracy, precision of a movie more accurately. This will improve the work load and execution time.

**REFERENCES**

[1] Hirdesh Shivhare, Anshul Gupta and Shalki Sharma (2015), “Recommender system using fuzzy c-means clustering and genetic algorithm based weighted similarity measure”, IEEE International Conference on Computer, Communication and Control.

[2] Manoj Kumar, D.K. Yadav, Ankur Singh and Vijay Kr. Gupta (2015), “A Movie Recommender System: MOVREC”, International Journal of Computer Applications (0975 – 8887) Volume 124 – No.3.

[3] RyuRi Kim, Ye Jeong Kwak, HyeonJeong Mo, Mucheol Kim, Seungmin Rho,Ka Lok Man, Woon Kian Chong (2015),“Trustworthy Movie Recommender System with Correct Assessment and Emotion Evaluation”, Proceedings of the International MultiConference of Engineers and Computer Scientists Vol II.

[4] Zan Wang, Xue Yu\*, Nan Feng, Zhenhua Wang (2014), “An Improved Collaborative Movie Recommendation System using Computational Intelligence”,Journal of Visual Languages & Computing,Volume 25, Issue 6.

[5] Debadrita Roy, Arnab Kundu, (2013), “Design of Movie Recommendation System by Means of Collaborative Filtering”, International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 4.