A MINI-PROJECT REPORT

ON

“Movie Recommendation System”

Submitted by

Shubham Dharmesh Kothari (A-38)

Aniket Babaji Kurkute (A-40)

Atul Nagnath Gadekar (A-23)

Supervisor:

Mrs. Varsha Wangikar

Department of Computer Engineering

K.C. College of Engineering and Management Studies And Research, Thane (E)

University of Mumbai

2020-21

CERTIFICATE

Certified that the mini-project work entitled **“Movie Recommendation System”** is a bonafide work carried out by

Shubham Dharmesh Kothari (A-38)
Aniket Babaji Kurkute (A-40)
Atul Nagnath Gadekar (A-23)

The report has been approved as it satisfies the academic requirements in respect of mini-project work prescribed for the course.

…………………………	……………………………
Mrs. Varsha Wangikar	**Mr. Mandar Ganjapurkar**
Supervisor	Head of the Department

……………………….
Dr.Vilas Nitnaware
Principal

Department of Computer Engineering

K.C. College of Engineering and Management Studies And Research, Thane (E)
University of Mumbai

2020-21

DECLARATION

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name of Students & Roll Nos	Signature
Shubham Dharmesh Kothari (A-38)	
Aniket Babaji Kurkute (A-40)	
Atul Nagnath Gadekar (A-23)	

Date:

ACKNOWLEDGEMENT

No project is ever complete without the guidance of those expert who have already traded this past before and hence become master of it and as a result, our leader. So, we would like to take this opportunity to take all those individuals how have helped us in visualizing this project.

We would take this opportunity to thank our mini project co-ordinator **Mrs. Varsha Wangikar** and other staff for their guidance in selecting this project and also for providing timely assistant to our query and guidance of this project.

We are very grateful to our Head of the Department **Mr. Mandar Ganjapurkar** for extending his help directly and indirectly through various channels in our project work.

We extend our sincerity appreciation to our Principal **Dr. Vilas Nitnaware** for providing us the opportunity to implement our project.

We are really thankful to all our Professors from K.C. College of Engineering and Management Studies and Research for their valuable inside and tip during the designing of the project. Their contributions have been valuable in so many ways that we find it difficult to acknowledge of them individual.

Thanking You.

Index

List of Figure

List of Table

ABSTRACT

The main aim of project “**M**ovie **R**ecommendation **S**ystem” is to help people to decide which movie to watch next. In a busy world watching trailers and reviews before going to cinema to watch a particular movie is really old fashioned by wasting time on 100 of reviews to read the movie recommender system gives everyone exact idea about the next movie they would like to watch

With this the movie recommendation system also keep track of what kind of movie a user like so that recommended movies are best to watch for that user. In lots of variety of movies and web series to find what exactly someone will like is very tedious job and this is where the Machine learning helps the movie recommender to get best out of previous data about user.

1. Introduction

1.1 Introduction

Recommendation system is a subclass of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item.

Recommender systems are used in a variety of areas, with commonly recognized examples taking the form of playlist generators for video and music services, product recommenders for online stores, or content recommenders for social media platforms and open web content recommenders.

The most popular recommendations engine includes movie recommendations are being used by the people since 1996

1.2Purpose and Scope

Document Purpose

This document is intended to provide the report of the Movie Recommender System project. The document includes the project perspective, data model and constraints of the overall system.

This document provides detailed information about the functionalities of the platform.

It explains the usage i.e. by whom and how can it be used. Implementation i.e. uses cases and its implementation on various levels. Future scope i.e. advancements of the platform.

Product Scope

The movie recommender system give recommendation to users based on their previous seen movies also it recommends top rated movies so that no cold start problem happen for the user

This platform is beneficial for all movie enthusiasts, movie makers, and other people involved in a production of movies. Movie enthusiasts can see which movie to b seen next, movie makers and other people involved in making a movie see the top rated movies and try to make movies of similar tastes for the audience.

1.3 Problem Statement

The world of Entertainment is changing rapidly with the entry of OTT (over the top) frameworks users get a Wide range of movies and web series to see. In this kind of huge movie and web series library's finding best movie to see next is very tedious job this is where recommendation system help us .Based on user old preferences the recommendation system tries to find best movies which the user will like

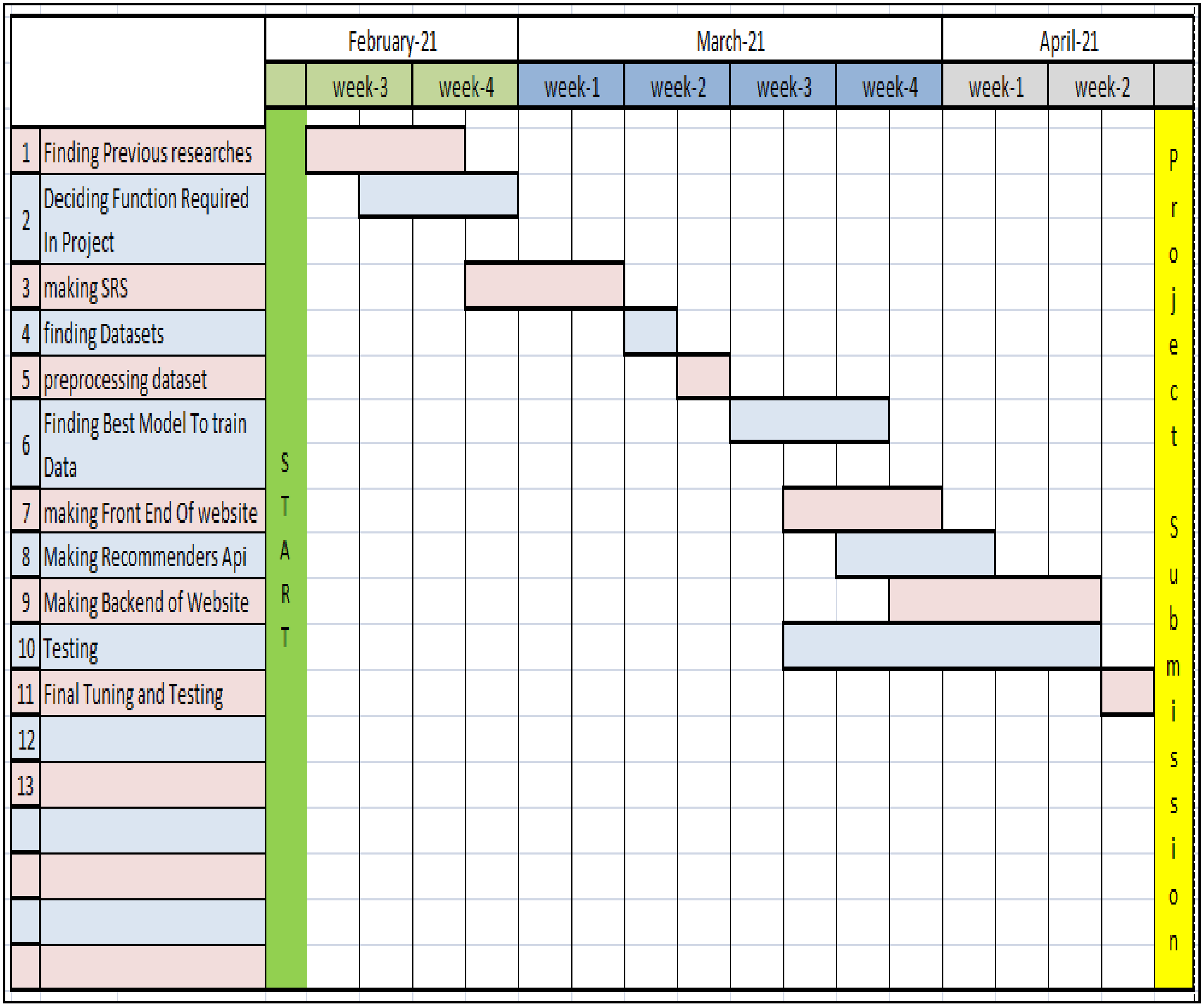
The recommendation system we try to make is build using collaborative clustering where in even the new user gets recommendation depending on most popular movies at that time with this method we try to solve cold start problem in recommendation system. Movies depending on era the user like actors the user like genres the user like will be recommended using the Ml algorithm. Using the data of past movies watched by user the recommended system try to give new suggestions of same emotion.

2. Project Analysis

2.1 Review of Literature

SR No	Year Published	Paper	Conclusion
1	2017	An effective collaborative movie recommender system with cuckoo search	Using Cuckoo algorithm to find recommendation
2	2017	Recommendation system for cold start items	Use of popular movies of database to show at the start of recommendation
3	2017	Enhanced Movie Content Similarity Based on Textual, Auditory and Visual Information	Use of cosine similarity for recommendation of different movies
4	2019	Trends in content-based recommendation	Popular algorithms such as knn cosine similarity are best for finding recommendations
5	2018	Content Based Movie Recommendation System Using Genre Correlation	Use of cosine similarity for finding similar movies

2.2 Project Timeline



			February-21				March-21								April-21				
			week-3		week-4		week-1		week-2		week-3		week-4		week-1		week-2		
1	Finding Previous researches	START	■	■	■														Project Submission
2	Deciding Function Required In Project			■	■	■													
3	making SRS					■	■	■											
4	finding Datasets								■										
5	preprocessing dataset									■									
6	Finding Best Model To train Data										■	■	■						
7	making Front End Of website											■	■	■					
8	Making Recommenders Api												■	■	■				
9	Making Backend of Website													■	■	■	■		
10	Testing											■	■	■	■	■	■		
11	Final Tuning and Testing																	■	
12																			
13																			

2.3 Dataset details

DATA SET link - https://www.kaggle.com/shubhammehta21/movie-lens-small-latest-dataset?select=ratings.csv

	Movie recommendation model	Sentiments analysis model
Algorithm	KNN	Multinomial Naive Bayes
Train test split		80 -20
datasets	Movielens small latest (contains movie.csv and rating.csv)	review.csv
Rows X columns	Movie.csv (10000 rows X 4 columns (movieId, title, genres, tmdb_id)) Ratings.csv (1 lac rows X 4 columns (userId, movieId , rating, timestamp)	7000 rows X 2 columns (Reviews ,Comments)
Accuracy		97 % on dataset may reduce by 10-15 % on real world data

2.4 Methodology Used

KNN- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique. K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm. K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems. K-NN is a **non-parametric algorithm**, which means it does not make any assumption on underlying data. It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

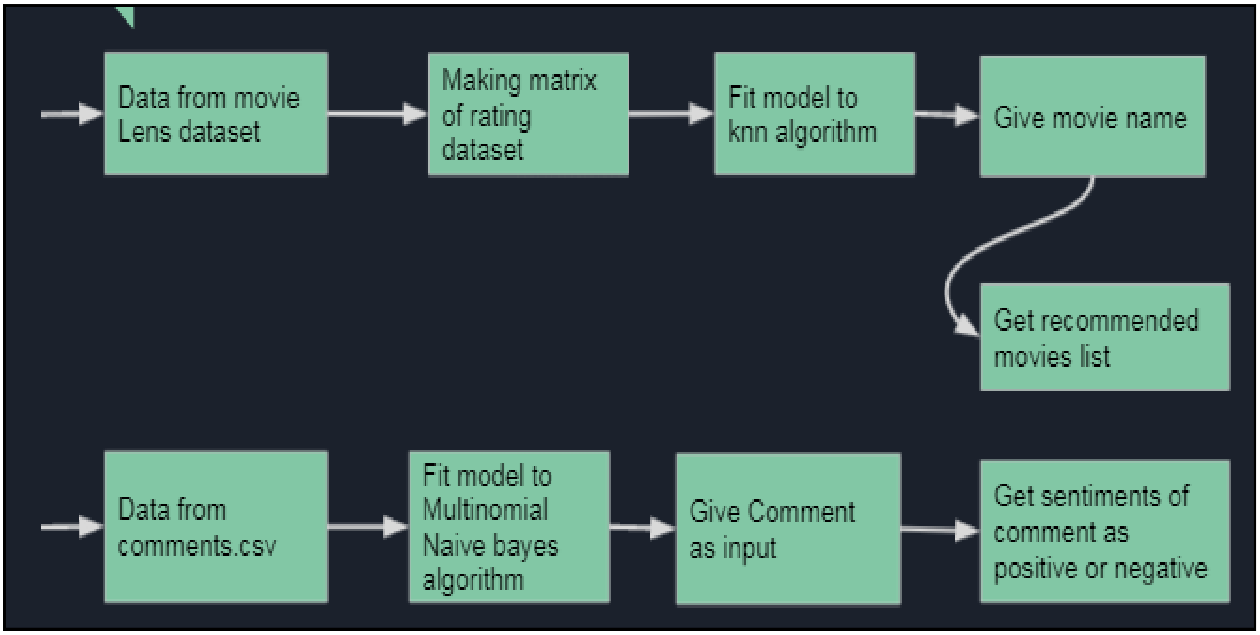
Multinomial Naive Bayes – MNB algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.

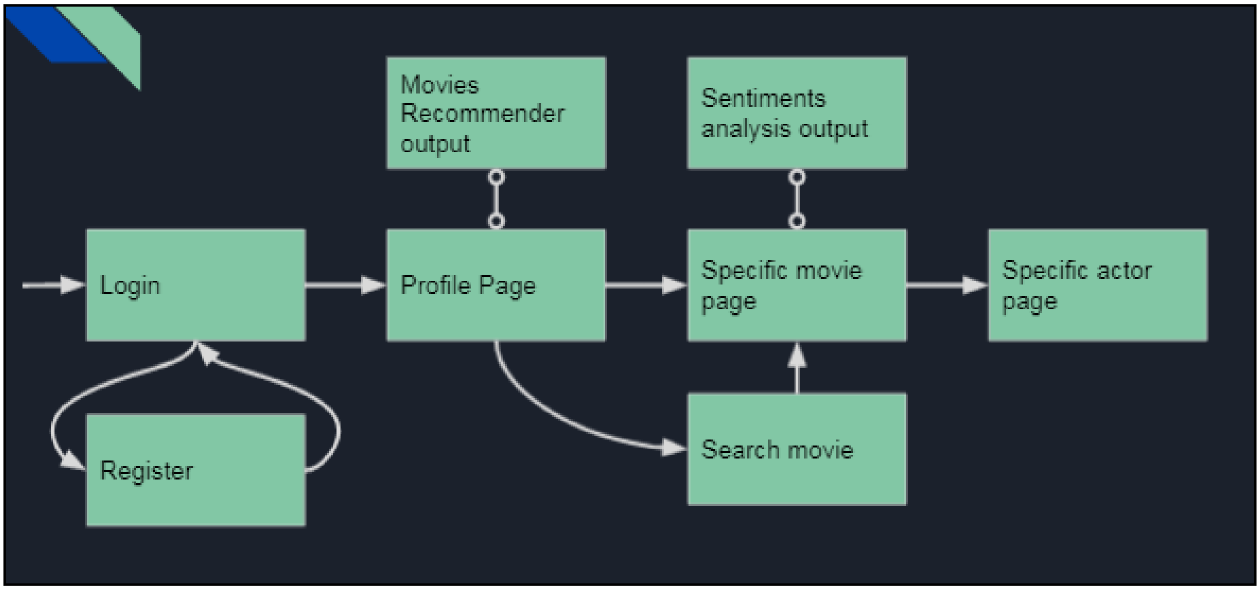
Naive Bayes classifier is a collection of many algorithms where all the algorithms share one common principle, and that is each feature being classified is not related to any other feature. The presence or absence of a feature does not affect the presence or absence of the other feature.

- We have used KNN and Multinomial Naïve Bayes in our movie recommender system
- Knn helps us to make movie recommendation and Naïve Bayes in comments sentiments analysis
- In Naïve Bayes a split of 80:20 data is been done for traing and testing after testing it gives accuracy of almost 97 %

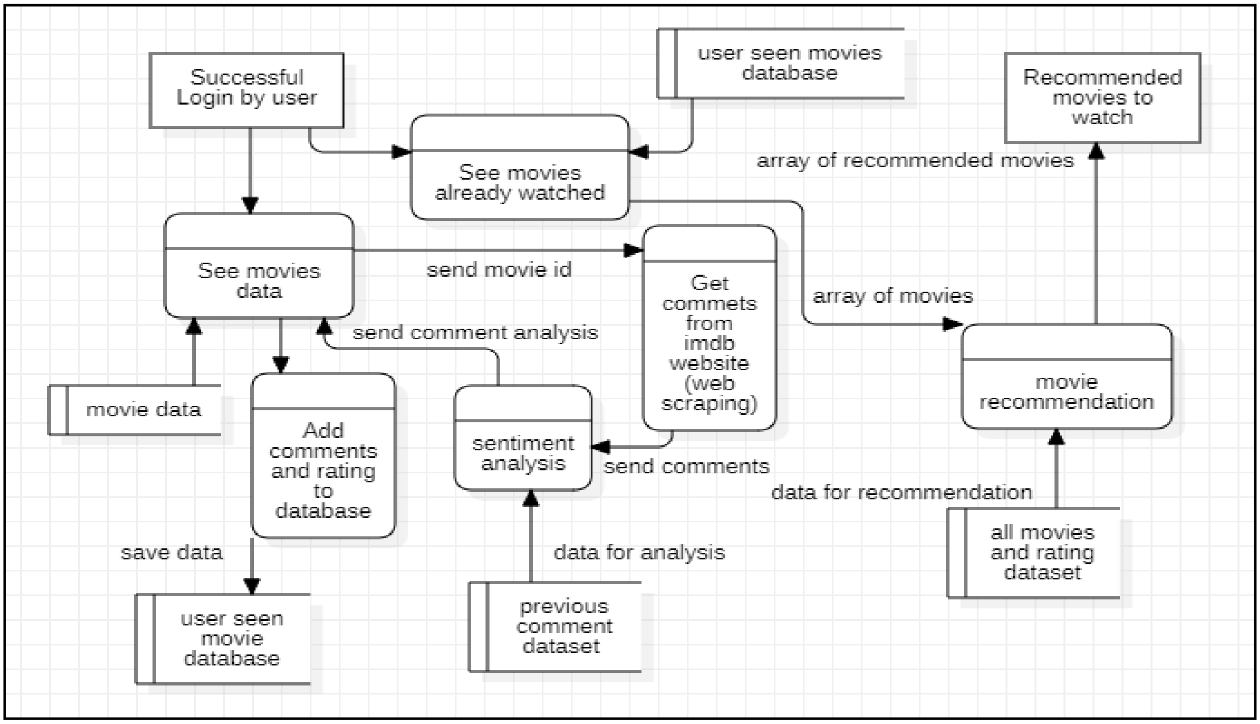
3. Project Design

3.1 Block Diagram

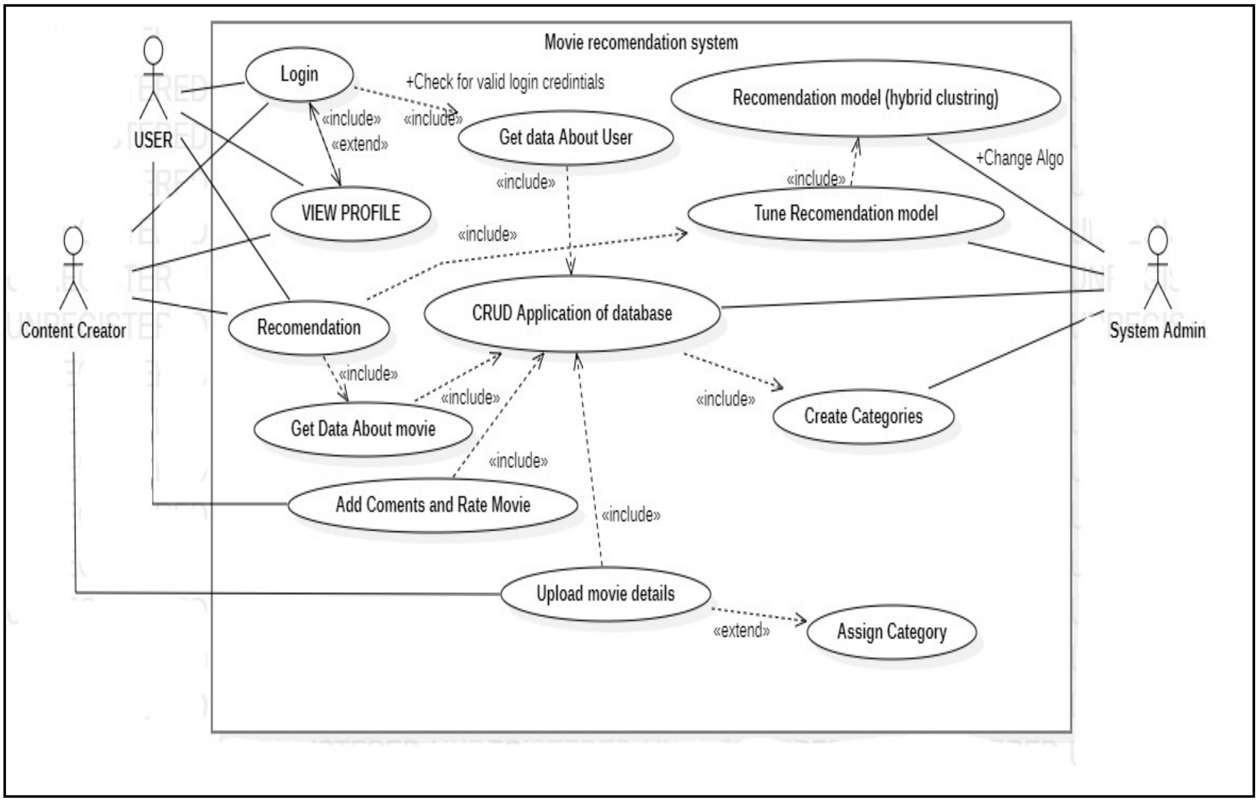

Data from movie Lens dataset
Making matrix of rating dataset
Fit model to knn algorithm
Give movie name
Get recommended movies list
Data from comments.csv
Fit model to Multinomial Naive bayes algorithm
Give Comment as input
Get sentiments of comment as positive or negative



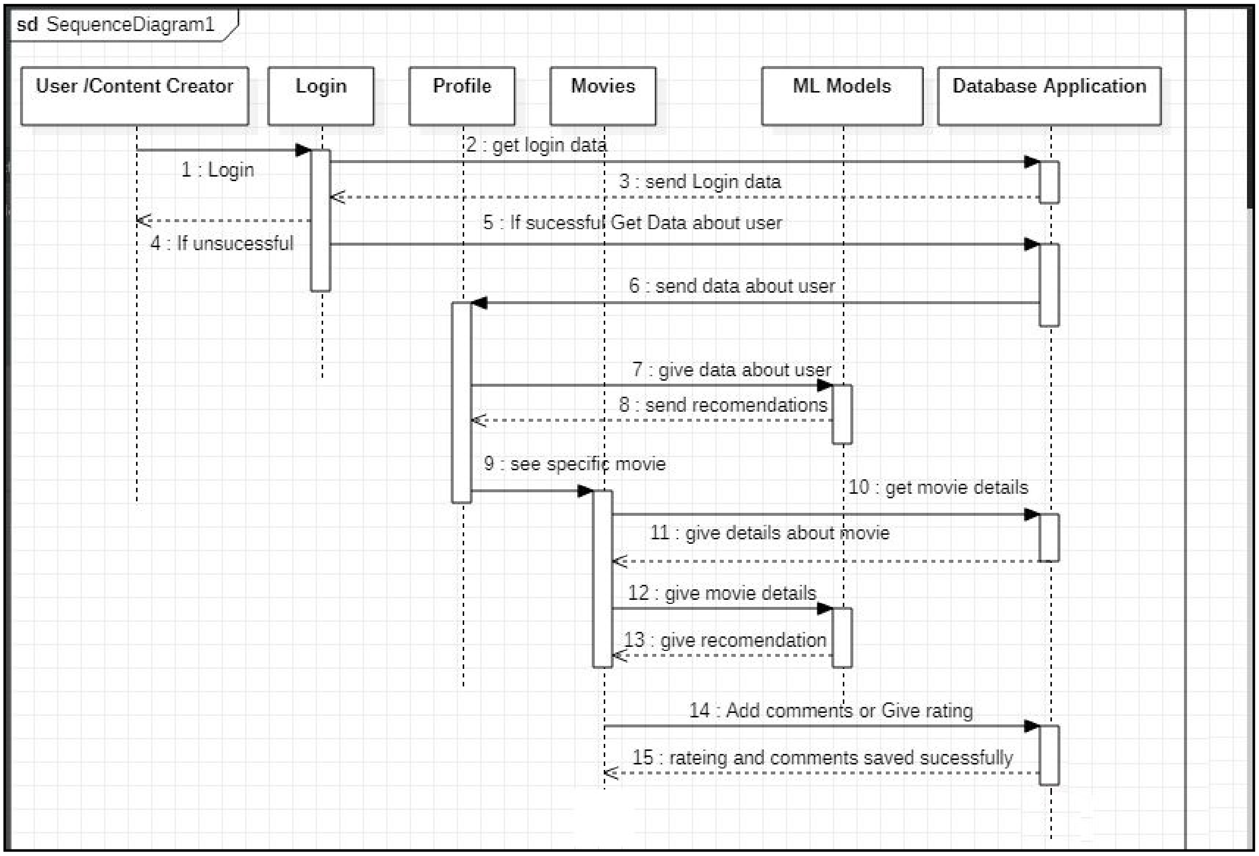
3.2 Data Flow Diagram



3.3 Use Case Diagram



3.4 Sequence Diagram



4. Implementation

4.1 Project Implementation Technology

Hardware Requirement

1. A computer with Minimum of 16 GB RAM and 200 GB ROM

Software Requirement

1. Python V3 3.6>
2. VS Code
3. Python Packages Such as Flask, Urllib, Requests, Sklearn, Pandas, Numpy etc
4. DB Browser (database visualizer)
5. Any one browser Chrome, Firefox, Edge, Opera etc

4.2 Experimental Setup

Preprocessing of data:-

1. The datasets movies.csv and ratrings.csv doesn’t require pre-processing
2. For comments.csv we use stop words to find all important words in the comments of the user then the list of important words are categorised based on their appearance in the positive comment and negative comment
3. Then this list of categories are been used for analysis

Api From Tmdb :-

1. Tmdb gives a huge amount of data about movies and for getting info about a movie tmdb api key is must

4.3 Coding

```
import hashlib
import json
import random
import sqlite3
import string
import requests
from flask import Flask, flash, redirect, render_template, request,
url_for,jsonify
from tmdbv3api import TMDb,Movie,Person

tmdb = TMDb()
tmdb.api_key = 'bf048b0e274f5a9ee17fa19066dfc3ed'
tmdb.language = 'en'
print("https://colab.research.google.com/drive/1346EhztureV4khTB7gb7JgEGwG
Z1J0TK?authuser=1#scrollTo=p3BYV3pDmalR")
sk_api=input("enter SK api=> ")
API_key='bf048b0e274f5a9ee17fa19066dfc3ed'
def rand_str():
    digits = string.digits
    letter_digit_list = list(string.digits + string.ascii_letters)
    random.shuffle(letter_digit_list)
    sample_str = ''.join((random.choice(digits) for i in range(6)))
    sample_str  +=  ''.join((random.choice(letter_digit_list)  for  i  in
range(10)))
    aList = list(sample_str)
    random.shuffle(aList)
    final_str = ''.join(aList)
    return final_str

def get_stats(data):
    def conv_dict_prec(dict_input):
        ret_array=[]
        s = sum(dict_input.values())
        for k, v in dict_input.items():
            pct = round(v * 100.0 / s,2)
            ret_array.append({"y":pct, "label":str(k)})
        return ret_array


gen_origin_dict={"genres":[{"id":28,"name":"Action"},{"id":12,"name":"Adve
nture"},{"id":16,"name":"Animation"},{"id":35,"name":"Comedy"},{"id":80,"n
ame":"Crime"},{"id":99,"name":"Documentary"},{"id":18,"name":"Drama"},{"id
":10751,"name":"Family"},{"id":14,"name":"Fantasy"},{"id":36,"name":"Histo
ry"},{"id":27,"name":"Horror"},{"id":10402,"name":"Music"},{"id":9648,"nam
e":"Mystery"},{"id":10749,"name":"Romance"},{"id":878,"name":"Science
Fiction"},{"id":10770,"name":"TV
Movie"},{"id":53,"name":"Thriller"},{"id":10752,"name":"War"},{"id":37,"na
me":"Western"}]}
    gen_dict,rate_dict=dict(),{'0':0,'1':0,'2':0,'3':0,'4':0,'5':0}
    ret=[]
    if data==[]:
        return []
    for i in data:
        #print(i[3])
        # 3 for genere if get data changed please change it too
        gen=i['genres'].split(';')
        for j in gen:
            if j!='':
                if j not in gen_dict:
                    gen_dict[j] = 1
                else:
                    gen_dict[j] += 1

        # 9 for user rating if get data changed please change it too
        rate=i['user_review']
        if rate=='':
            rate_dict[0]=rate_dict[0]+1
        else:
            rate_dict[rate]=rate_dict[rate]+1

    ret=[conv_dict_prec(gen_dict),conv_dict_prec(rate_dict)]
    #print(ret)
    return ret

def get_actor(x):
    movie = Movie()
    #print(dir(movie))
    m ,ret_lst= movie.details(int(x)),[]
    z=m.casts
    if len(z['cast'])>0:
        #print(z['cast'][0])
        if len(z['cast'])>0:
            top_cast = [0,1,2,3,4,5,6,7,8,9]
        else:
            top_cast = [0,1,2,3,4]
        for i in top_cast:
            #https://image.tmdb.org/t/p/original
            temp_lst={}
            for j in ['name','character','profile_path','id']:
                if j=='profile_path':
                    if z['cast'][i][j]!=None:

temp_lst[j]='https://image.tmdb.org/t/p/original'+z['cast'][i][j]
                    else:

temp_lst[j]='https://webstockreview.net/images/human-clipart-human-symbol-
19.png'
                    continue
                temp_lst[j]=z['cast'][i][j]
            #print(temp_lst)
            ret_lst.append(temp_lst)
        return ret_lst


def get_popular_movies():
    movie = Movie()
    popular = movie.popular()
    lst=[]
    j=0
    for p in popular:
        ret_lst={}
        ret_lst['title']=p.title
        ret_lst['id']=p.id
        if p.poster_path==None:

ret_lst['poster_path']='https://www.sundialhome.com/assets/images/noImage.
jpg'
        else:

ret_lst['poster_path']='https://image.tmdb.org/t/p/original'+p.poster_path
        ret_lst['popularity']=p.popularity
        j=j+1
        lst.append(ret_lst)
        if j>9:
            break
    return lst

def get_data_from_API(API_key, Movie_IDs): # get movie data from api
    text=[]
    data=[]
    user_review=[]
    posterpath='https://image.tmdb.org/t/p/original'
    for i in Movie_IDs:
        #print(i)
        query                                                                     =
'https://api.themoviedb.org/3/movie/'+str(i[0])+'?api_key='+API_key+'&lang
uage=en-US'
        response =  requests.get(query)
        if response.status_code==200:
            array = response.json()
            text.append(array)
            user_review.append(str(i[1]))
    for i in range(len(text)):
        #if change in service_req please change stat to

services=['title','id','imdb_id','genres','tagline','poster_path','popular
ity','release_date','vote_average','vote_count','overview']
        lst={}
        for req_service in services:
            if text[i][req_service] is None:
                if req_service=='poster_path':

lst['poster_path']='https://www.sundialhome.com/assets/images/noImage.jpg'
                else:
                    lst['poster_path']=' '
            elif req_service=='genres':
                gen=''
                for gen_name in text[i]['genres']:
                    gen=gen_name['name']+';'+gen
                lst['genres']=gen
            elif req_service=='poster_path':
                lst['poster_path']=str(posterpath+text[i]['poster_path'])
            else:
                lst[req_service]=text[i][req_service]
        lst['user_review']=user_review[i]
        data.append(lst)

    return data

def get_about_actors(x):
    person_ret={}
    people = Person()
    p=people.details(int(x))
    person_ret["name"]=p.name
    person_ret["birthday"]=p.birthday
    person_ret["place_of_birth"]=p.place_of_birth

person_ret["profile_path"]='https://image.tmdb.org/t/p/original'+str(p.pro
file_path)
    person_ret["popularity"]=p.popularity
    person_ret["biography"]=p.biography
    #person_ret["movies"]=[]
    p=people.movie_credits(int(x))
    a,temp1=0,[]
    for i in p['cast']:
        temp={}
        if a<20:
            temp['id']=i['id']

temp['poster_path']='https://image.tmdb.org/t/p/original'+str(i['poster_pa
th'])
```

```
            temp['title']=i['title']
            temp['character']=i['character']
            temp1.append(temp)
        else:
            break
        a+=1
    person_ret["movies"]=temp1
    return person_ret


def get_movie_recom(movie_id_lst,seen_movie):
    global sk_api
    seen=[]
    for i in seen_movie:
        seen.append(i[0])
    #print(seen)
    ret_dict,temp_dict=[],{}
    for i in movie_id_lst:
        con = sqlite3.connect("app.db")
        cur = con.cursor()
        z=cur.execute("select title from movies where movieId
=?",[i[0]]).fetchall()
        nta,a=0,''
        for j in z[0][0]:
            if j=='(':
                nta=1
                continue
            if j==')':
                nta=0
                continue
            if nta==0:
                a=a+j
        moviename=a.strip()
        query = sk_api+'/movieanalysis?moviename='+str(moviename)
        #print(query)
        response =  requests.get(query)
        if response.status_code==200:
            array = response.json()
            for k in array:
                if k not in ret_dict:
                    temp_dict[k] = 1
                else:
                    temp_dict[k] = temp_dict[k]+1
    temp_dict=sorted(temp_dict, key=temp_dict.get, reverse=True)
    a=0
    for i in temp_dict:
        if a<10:
            z=cur.execute("select tmdb_id from movies where title like
?",[i]).fetchall()[0][0]
            if int(z) not in seen:
                #print(z,seen)
                ret_dict.append((z,-1))
                a=a+1
    #print(ret_dict)
    ret_dict=get_data_from_API(API_key, ret_dict)
    return ret_dict
    #return ret_dict

def get_comments(Movie_ID): #get individual movie comments from imdb
    global sk_api
    text=[]
    #must be changed according to ngrok values
    query = sk_api+'/sentimentanalyisis?id='+str(Movie_ID)
    response =  requests.get(query)
    if response.status_code==200:
        array = response.json()
        return array
    else:
        return []


app = Flask(__name__)
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'

@app.route('/', methods=['GET','POST'])
@app.route('/login', methods=['GET','POST'])
def login():
    if request.method=='POST':
        email=request.form['email']
        password=request.form['password']
        if len(email)>1 and len(password)>1:
            result = hashlib.sha256(password.encode())
            result = result.hexdigest()
            con = sqlite3.connect("app.db")
            cur = con.cursor()
            z=cur.execute("select user_id,password from user where
email=?",[email]).fetchall()
            #rows = z.fetchall()
            try:
                if z[0][1]==result:
                    sess_str=rand_str()
                    cur.execute("update user set sess=? where
user_id=?",[sess_str,z[0][0]])
                    con.commit()
                    return redirect(url_for('profile',sess=sess_str))
                else :
                    flash(u'Invalid Crdentials ! ','alert alert-danger
alert-dismissible')
            except:
                flash(u'Invalid Crdentials ! ','alert alert-danger alert-
dismissible')

    return render_template('login.html')
    #return 'Hello, World!'

@app.route('/register', methods=['GET','POST'])
def register():
    if request.method=='POST':
        email = request.form['email']
        password = request.form['password']
        name = request.form['name']
        if len(email)>1 and len(password)>1 and len(name)>1:
            result = hashlib.sha256(password.encode())
            result = result.hexdigest()
            con = sqlite3.connect("app.db")
            cur = con.cursor()
            cur.execute("insert into user (name,email,password) values (
?, ?, ?)",[str(name),str(email),str(result)])
            con.commit()
            flash(u'Added Account Successfully','alert alert-success
alert-dismissible')
            return redirect(url_for('login'))

    return render_template('register.html')

@app.route('/profile', methods=['GET','POST'])
def profile():
    if request.method=='GET':
        user_data=[]
        sess=request.args.get('sess')
        con = sqlite3.connect("app.db")
        cur = con.cursor()
        seen_movie=cur.execute("select movie_id,rateing from user_movie
where user_id = (select user_id from user where sess= ?) order by
time_stamp DESC",[sess]).fetchall()
        print(seen_movie)
        user_data.append(len(seen_movie))
        seen_movie_data=get_data_from_API(API_key, seen_movie)
        z=cur.execute("select name,email from user where user_id = (select
user_id from user where sess=?)",[sess]).fetchall()
        user_data.append(z[0][0])
        user_data.append(z[0][1])
        stats=get_stats(seen_movie_data)
        popular_movie=get_popular_movies()
        z=cur.execute("select movieId from movies where tmdb_id in (select
movie_id from user_movie where user_id=(select user_id from user where
sess= ?))",[sess]).fetchall()
        recomended_movie=get_movie_recom(z,seen_movie)
        con.commit()
        if stats==[]:
            return
render_template('profile.html',sess_str=sess,recomended_movie=recomended_m
ovie,data=seen_movie_data,user_data=user_data,popular_movie=popular_movie)
        #print(stats[0])
        else:
            return
render_template('profile.html',sess_str=sess,recomended_movie=recomended_m
ovie,data=seen_movie_data,popular_movie=popular_movie,user_data=user_data,
stats_gen=json.dumps(stats[0]),stats_rating=json.dumps(stats[1]))

@app.route('/movie', methods=['GET','POST'])
def movie():
    if request.method=='POST':
        sess=request.form['sess']
        comment=request.form['comment']
        rating=request.form['rating']
        add_to_watch=request.form['add_to_watch']
        #print(sess,comment,rating,add_to_watch)
        con = sqlite3.connect("app.db")
        cur = con.cursor()
        z=cur.execute("select user_id,movie_id from user_movie where
user_id= (select user_id from user where sess=?) and movie_id =
?",[sess,add_to_watch]).fetchall()
        if z==[]:
            cur.execute("insert into user_movie
(user_id,movie_id,rateing,comments) values ((select user_id from user
where sess=?),?,?,?)",[sess,add_to_watch,rating,comment])
        else:
            cur.execute("update user_movie set rateing=? , comments=?
where user_id = (select user_id from user where sess=?) and movie_id =
?",[rating,comment,sess,add_to_watch])
        con.commit()
        return redirect(url_for('searchmovie',sess=sess))
    if request.method=='GET':
        user_data=[]
        sess=request.args.get('sess')
        movie_tmbd_id=request.args.get('movie_id')
        con = sqlite3.connect("app.db")
        cur = con.cursor()
        z=cur.execute("select rateing,comments from user_movie where
user_id = (select user_id from user where sess= ?) and movie_id =
?",[sess,movie_tmbd_id]).fetchall()
        if z==[]:
            z.append([-1])
        seen_movie_data=get_data_from_API(API_key,
[[movie_tmbd_id,z[0][0]]])
        actor=get_actor(movie_tmbd_id)
        #print(seen_movie_data[0]['imdb_id'])
        comments=get_comments(seen_movie_data[0]['imdb_id'])
        #print(comments)
        #print(seen_movie_data)
        con.commit()
        return
render_template('movie.html',sess=sess,seen_movie_data=seen_movie_data[0],
actor=actor,comments=comments,user_rate_comm=z)

@app.route('/actor', methods=['GET','POST'])
def actor():
    if request.method=='GET':
        sess=request.args.get('sess')
        actor_id=request.args.get('actor_id')
        actor_data=get_about_actors(actor_id)
        return
render_template('actor.html',sess=sess,actor_data=actor_data)

@app.route('/autocomplete', methods=['GET'])
def autocomplete():
    if request.method=='GET':
        search = request.args.get('q')
        con = sqlite3.connect("app.db")
        cur = con.cursor()
        query=cur.execute("select title from movies where title like
?",['%'+str(search)+'%'])
        con.commit()
        results = [mv[0] for mv in query.fetchall()]
        return jsonify(matching_results=results)
```

```
@app.route('/searchmovie', methods=['GET','POST'])
def searchmovie():
    if request.method=='GET':
        if request.args.get('q')!=None:
            search = request.args.get('q')
            sess=request.args.get('sess')
            con = sqlite3.connect("app.db")
            cur = con.cursor()
            z=[]
            query=cur.execute("select tmdb_id from movies where title like 
?",['%'+str(search)+'%']).fetchall()
            for i in query:
                z.append((i[0],-1))
            movie_data=get_data_from_API(API_key, z)
            con.commit()
            print(query)
            return 
render_template('searchMovie.html',sess=sess,movie_data=movie_data)
        else:
            sess=request.args.get('sess')
            return render_template('searchMovie.html',sess=sess)

if __name__=='__main__':
    app.run(debug=True,port=5000,use_reloader=False)
```

4.4 Testing

1. The testing of a recommender system can only be done with reviews from real users
2. While doing testing we found that when we gave a single movie to the recommender the recommendation which we got were the movies we have already seen and we loved to watch
3. So according to us the Accuracy of the model is 80% and can be increased by adding more movies and rating data to the system
4. The testing score of sentiments analysis model is 97 % and it could easily distinguish the different keywords in the comment

5. Result

Snapshot of Result

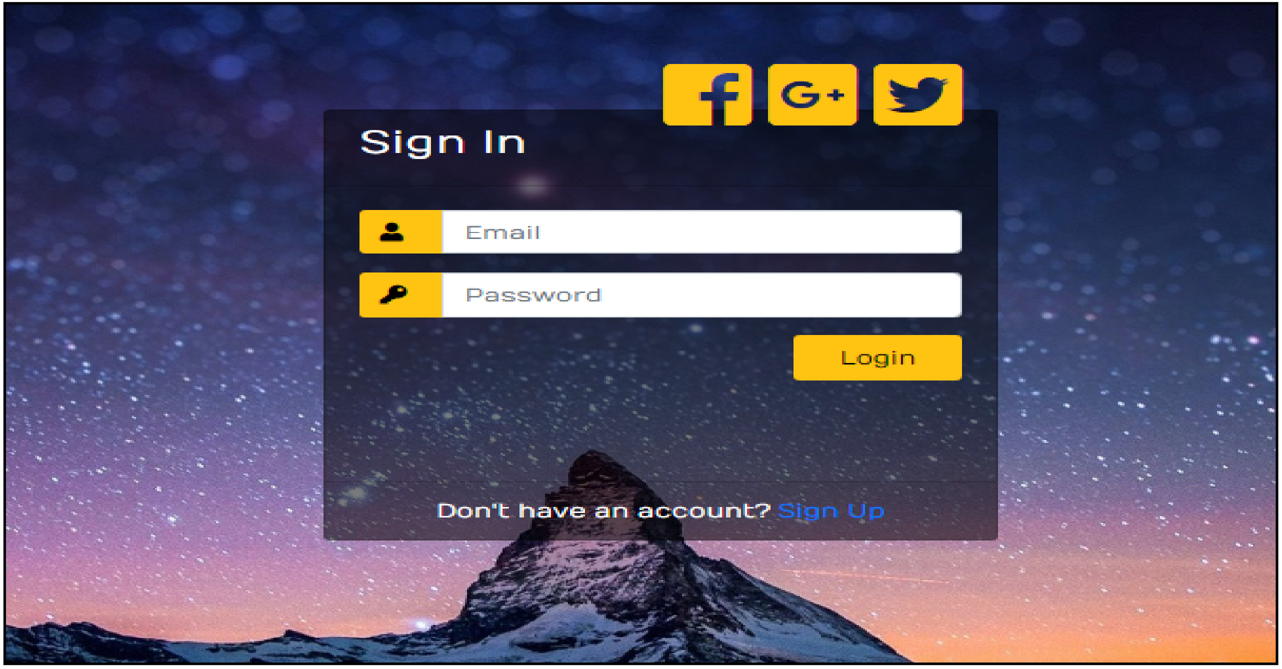
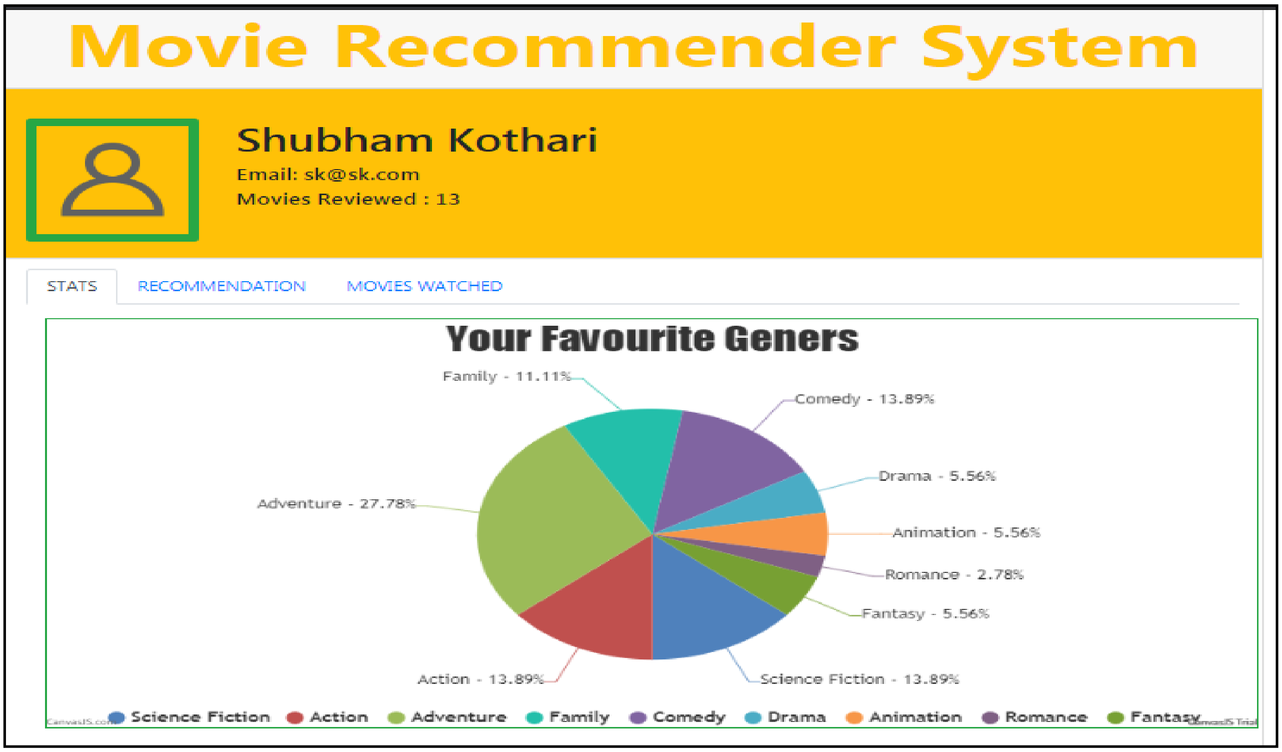


Figure 1: Login Page

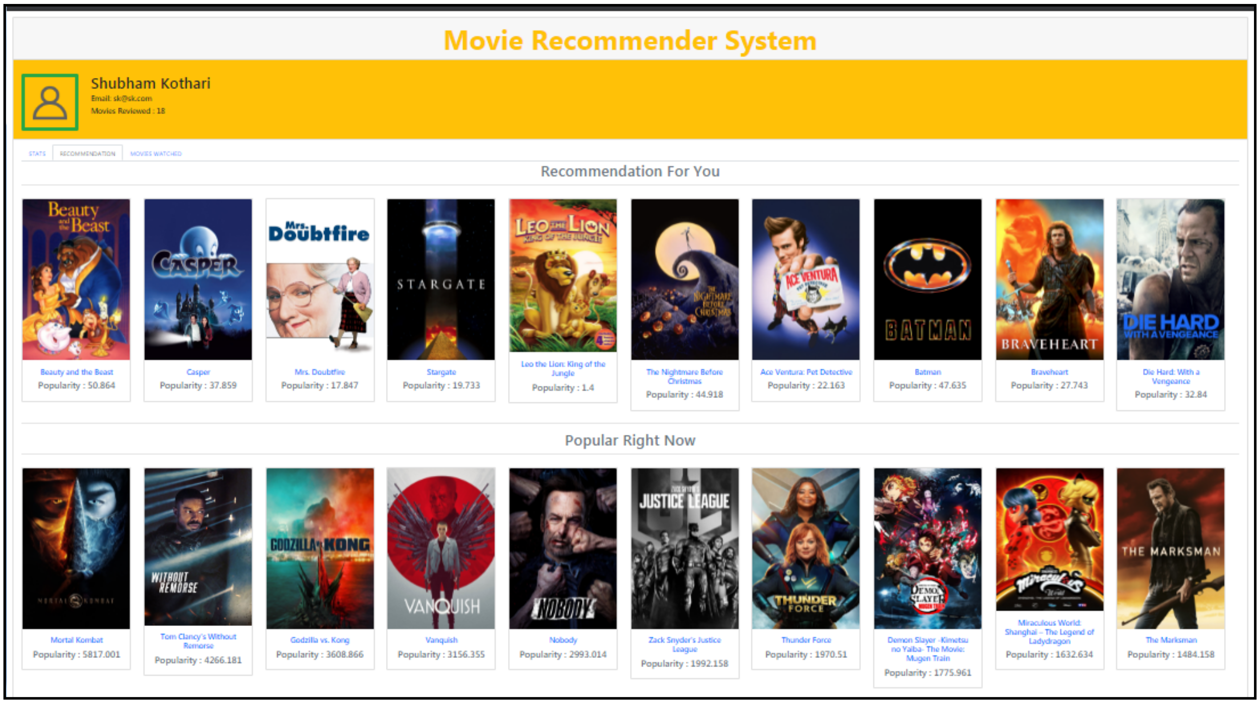

Movie Recommender System

Shubham Kothari

Email: sk@sk.com

Movies Reviewed : 13

Figure 2: Profile Page (STATS)



Movie Recommender System

Shubham Kothari
Email: sk@sk.com
Movies Reviewed : 18

Recommendation For You

Beauty and the Beast
Popularity : 50.864

Casper
Popularity : 37.859

Mrs. Doubtfire
Popularity : 17.847

Stargate
Popularity : 19.733

Leo the Lion: King of the Jungle
Popularity : 1.4

The Nightmare Before Christmas
Popularity : 44.918

Ace Ventura: Pet Detective
Popularity : 22.163

Batman
Popularity : 47.635

Braveheart
Popularity : 27.743

Die Hard: With a Vengeance
Popularity : 32.84

Popular Right Now

Mortal Kombat
Popularity : 5817.001

Tom Clancy's Without Remorse
Popularity : 4266.181

Godzilla vs. Kong
Popularity : 3608.866

Vanquish
Popularity : 3156.355

Nobody
Popularity : 2993.014

Zack Snyder's Justice League
Popularity : 1992.158

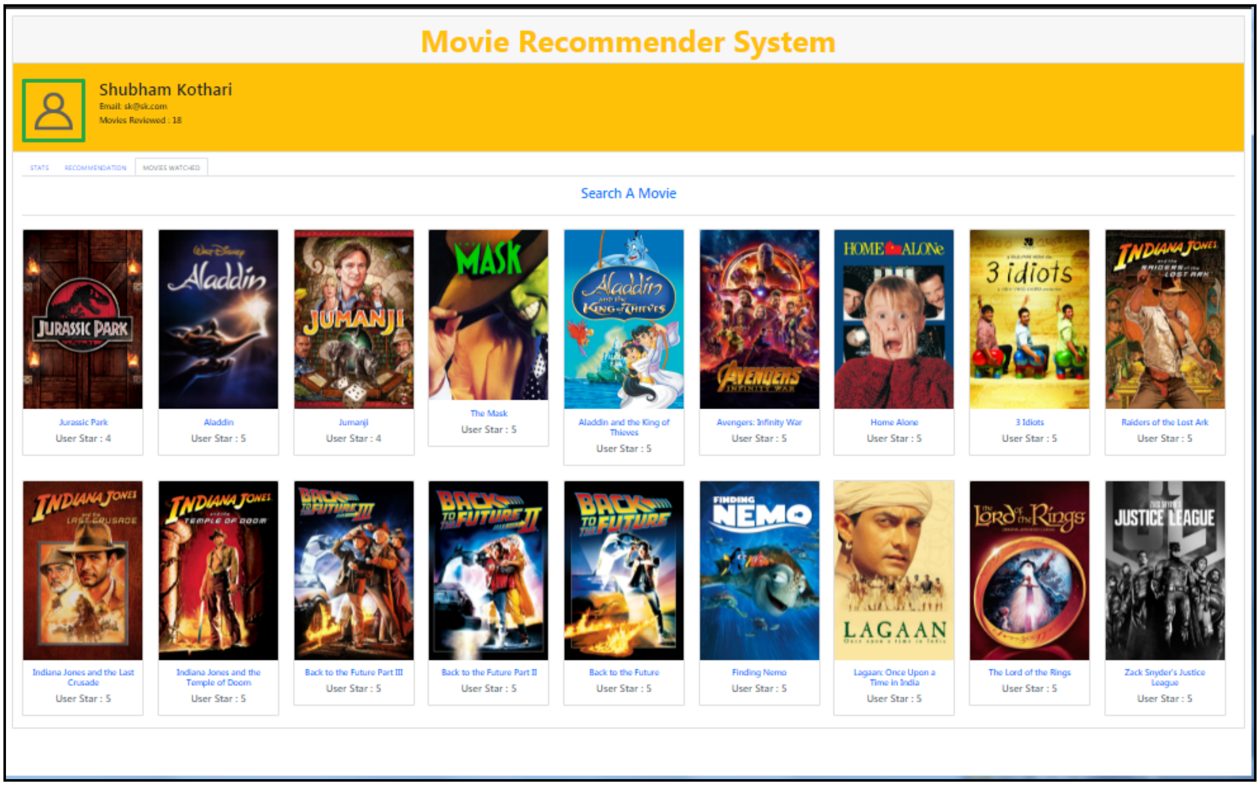
Thunder Force
Popularity : 1970.51

Demon Slayer -Kimetsu no Yaiba- The Movie: Mugen Train
Popularity : 1775.961

Miraculous World: Shanghai – The Legend of Ladydragon
Popularity : 1632.634

The Marksman
Popularity : 1484.158

Figure 3: Profile Page (Recommended Movie)



Movie Recommender System

Shubham Kothari
Email: sk@sk.com
Movies Reviewed : 18

Search A Movie

Jurassic Park
User Star : 4

Aladdin
User Star : 5

Jumanji
User Star : 4

The Mask
User Star : 5

Aladdin and the King of Thieves
User Star : 5

Avengers: Infinity War
User Star : 5

Home Alone
User Star : 5

3 Idiots
User Star : 5

Raiders of the Lost Ark
User Star : 5

Indiana Jones and the Last Crusade
User Star : 5

Indiana Jones and the Temple of Doom
User Star : 5

Back to the Future Part III
User Star : 5

Back to the Future Part II
User Star : 5

Back to the Future
User Star : 5

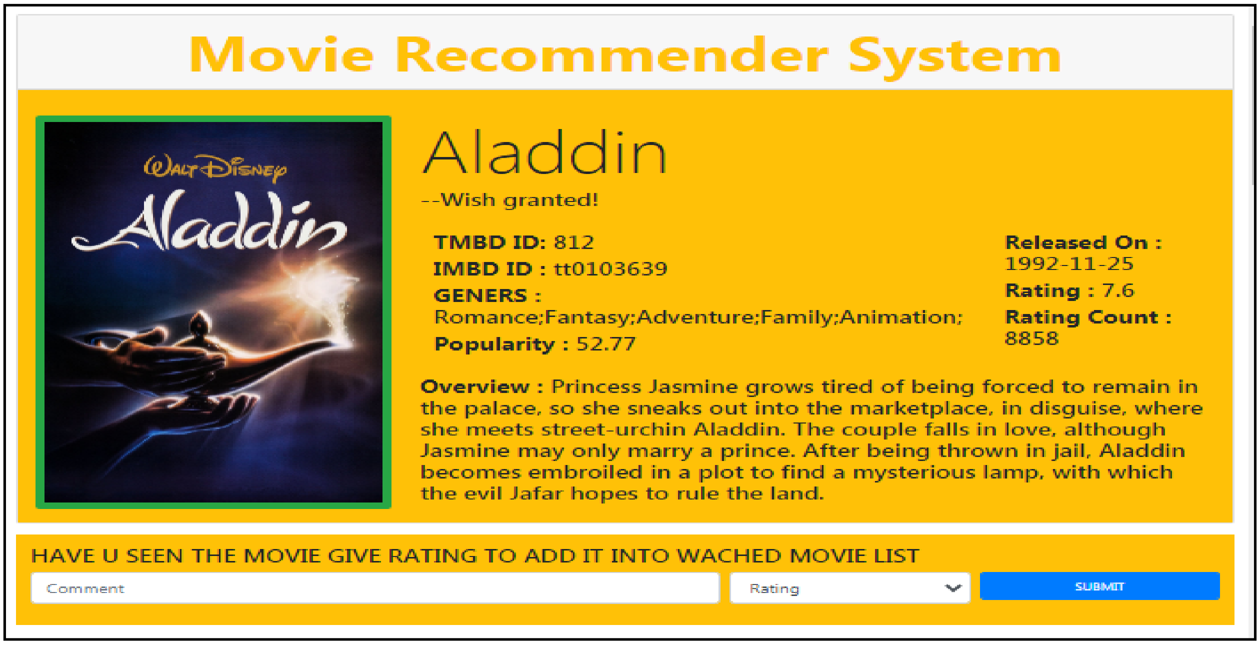
Finding Nemo
User Star : 5

Lagaan: Once Upon a Time in India
User Star : 5

The Lord of the Rings
User Star : 5

Zack Snyder's Justice League
User Star : 5

Figure 4: Profile Page (Previous Seen Movies)



Movie Recommender System

Aladdin

--Wish granted!

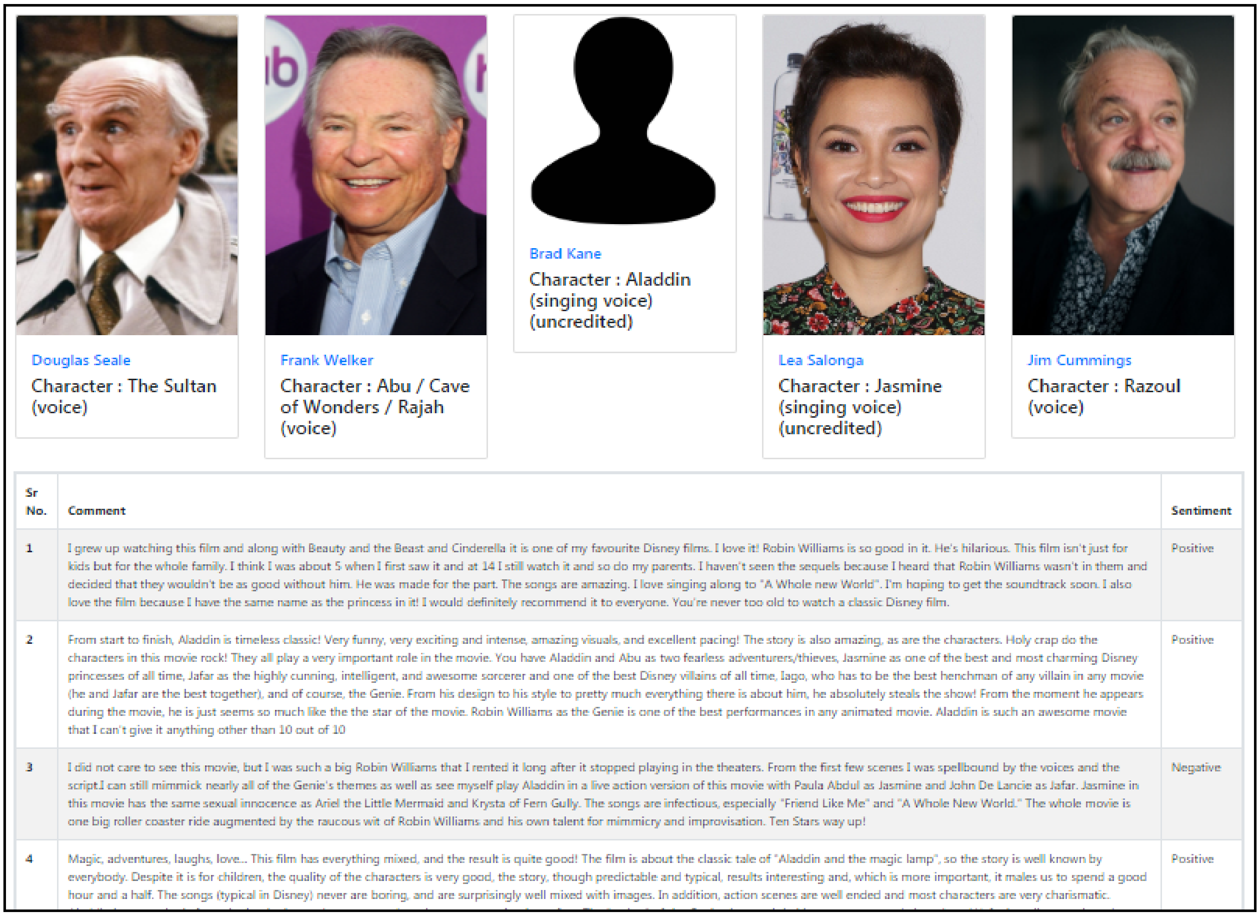
TMBD ID: 812
IMBD ID : tt0103639
GENERS : Romance;Fantasy;Adventure;Family;Animation;
Popularity : 52.77

Released On : 1992-11-25
Rating : 7.6
Rating Count : 8858

Overview : Princess Jasmine grows tired of being forced to remain in the palace, so she sneaks out into the marketplace, in disguise, where she meets street-urchin Aladdin. The couple falls in love, although Jasmine may only marry a prince. After being thrown in jail, Aladdin becomes embroiled in a plot to find a mysterious lamp, with which the evil Jafar hopes to rule the land.

HAVE U SEEN THE MOVIE GIVE RATING TO ADD IT INTO WACHED MOVIE LIST

Figure 5: Movies Page (Aladdin Movie)



Douglas Seale
Character : The Sultan (voice)

Frank Welker
Character : Abu / Cave of Wonders / Rajah (voice)

Brad Kane
Character : Aladdin (singing voice) (uncredited)

Lea Salonga
Character : Jasmine (singing voice) (uncredited)

Jim Cummings
Character : Razoul (voice)

Sr No.	Comment	Sentiment
1	I grew up watching this film and along with Beauty and the Beast and Cinderella it is one of my favourite Disney films. I love it! Robin Williams is so good in it. He's hilarious. This film isn't just for kids but for the whole family. I think I was about 5 when I first saw it and at 14 I still watch it and so do my parents. I haven't seen the sequels because I heard that Robin Williams wasn't in them and decided that they wouldn't be as good without him. He was made for the part. The songs are amazing. I love singing along to "A Whole new World". I'm hoping to get the soundtrack soon. I also love the film because I have the same name as the princess in it! I would definitely recommend it to everyone. You're never too old to watch a classic Disney film.	Positive
2	From start to finish, Aladdin is timeless classic! Very funny, very exciting and intense, amazing visuals, and excellent pacing! The story is also amazing, as are the characters. Holy crap do the characters in this movie rock! They all play a very important role in the movie. You have Aladdin and Abu as two fearless adventurers/thieves, Jasmine as one of the best and most charming Disney princesses of all time, Jafar as the highly cunning, intelligent, and awesome sorcerer and one of the best Disney villains of all time, Iago, who has to be the best henchman of any villain in any movie (he and Jafar are the best together), and of course, the Genie. From his design to his style to pretty much everything there is about him, he absolutely steals the show! From the moment he appears during the movie, he is just seems so much like the the star of the movie. Robin Williams as the Genie is one of the best performances in any animated movie. Aladdin is such an awesome movie that I can't give it anything other than 10 out of 10	Positive
3	I did not care to see this movie, but I was such a big Robin Williams that I rented it long after it stopped playing in the theaters. From the first few scenes I was spellbound by the voices and the script.I can still mimmick nearly all of the Genie's themes as well as see myself play Aladdin in a live action version of this movie with Paula Abdul as Jasmine and John De Lancie as Jafar. Jasmine in this movie has the same sexual innocence as Ariel the Little Mermaid and Krysta of Fern Gully. The songs are infectious, especially "Friend Like Me" and "A Whole New World." The whole movie is one big roller coaster ride augmented by the raucous wit of Robin Williams and his own talent for mimmicry and improvisation. Ten Stars way up!	Negative
4	Magic, adventures, laughs, love... This film has everything mixed, and the result is quite good! The film is about the classic tale of "Aladdin and the magic lamp", so the story is well known by everybody. Despite it is for children, the quality of the characters is very good, the story, though predictable and typical, results interesting and, which is more important, it males us to spend a good hour and a half. The songs (typical in Disney) never are boring, and are surprisingly well mixed with images. In addition, action scenes are well ended and most characters are very charismatic.	Positive

Figure 6: Movies Page (Comment's Sentiment analysis)

Analysis of result

1. The recommended movies are been seen on the website
2. The sentiments are assumed accurately by the model
3. Overall the movies recommendation system can be easily hosted as a website and can be used by real world users

Advantage and Disadvantages of Model

Advantages

1. Gives user stats about the movies they like
2. The website allows everyone to get their own set of recommended movies unlike Google’s recommender which takes only one movie for recommendation at a time
3. Everyone gets clear idea about movies by seeing sentiments of peoples comments

Disadvantages

1. Less movies in dataset :- contains only 9500 movies
2. All attributes are not taken into consideration for model even though they have partial effect on the model
3. Less optimized or slow code :- use of python has made a code little bit slow

Conclusion & Future Scope

Conclusion

Thus we have made movie recommender system using KNN algorithm movie lens dataset and Python Flask
We have learned and used Technologies Such as python Flask, KNN, HTML and CSS

Future Scope

1. Need of improvement of dataset :- only movies which are popular are being present in used dataset we need to increase the no of movies in the data set
2. Need to add more attributes for ml algorithm :- for now we only consider rating genre as parameter but more attributes which partially or fully support the model are needed to be added (e.g. -revenue, actors, time of release ,etc…)

3. Need to optimize the code and database for smooth execution of the application :- upgrade the database from sqlite3 to some good and faster database , optimizing python code using packages like pypy

References

1. An effective collaborative movie recommender system with cuckoo search - Egyptian Informatics Journal - Volume 18, Issue 2, July 2017, Pages 105-112 https://www.sciencedirect.com/science/article/pii/S1110866516300470

2. Recommendation system for cold start items (2017) https://discovery.dundee.ac.uk/ws/files/39285097/Recommendation_system_for_cold_start_items.pdf

3. Multimodal Trust Based Recommende_sasmita (2021) https://drsachinandan.com/publication/2021/3.MultimodalTrustBasedRecommende_sasmita.pdf

4. Use of Deep Learning in Modern Recommendation System: A Summary of Recent Works (2017) https://arxiv.org/ftp/arxiv/papers/1712/1712.07525.pdf

5. An Efficient movie recommendation algorithm based on improved k-clique (2018) https://hcis-journal.springeropen.com/track/pdf/10.1186/s13673-018-0161-6.pdf

6. Enhanced Movie Content Similarity Based on Textual, Auditory and Visual Information (2017) https://arxiv.org/pdf/1711.03889.pdf

7. Trends in content-based recommendation (2019) https://link.springer.com/content/pdf/10.1007/s11257-019-09231-w.pdf

8. https://flask.palletsprojects.com/en/1.1.x/

9. https://jinja.palletsprojects.com/en/2.11.x/

10. https://www.kaggle.com/shubhammehta21/movie-lens-small-latest-dataset?select=ratings.csv