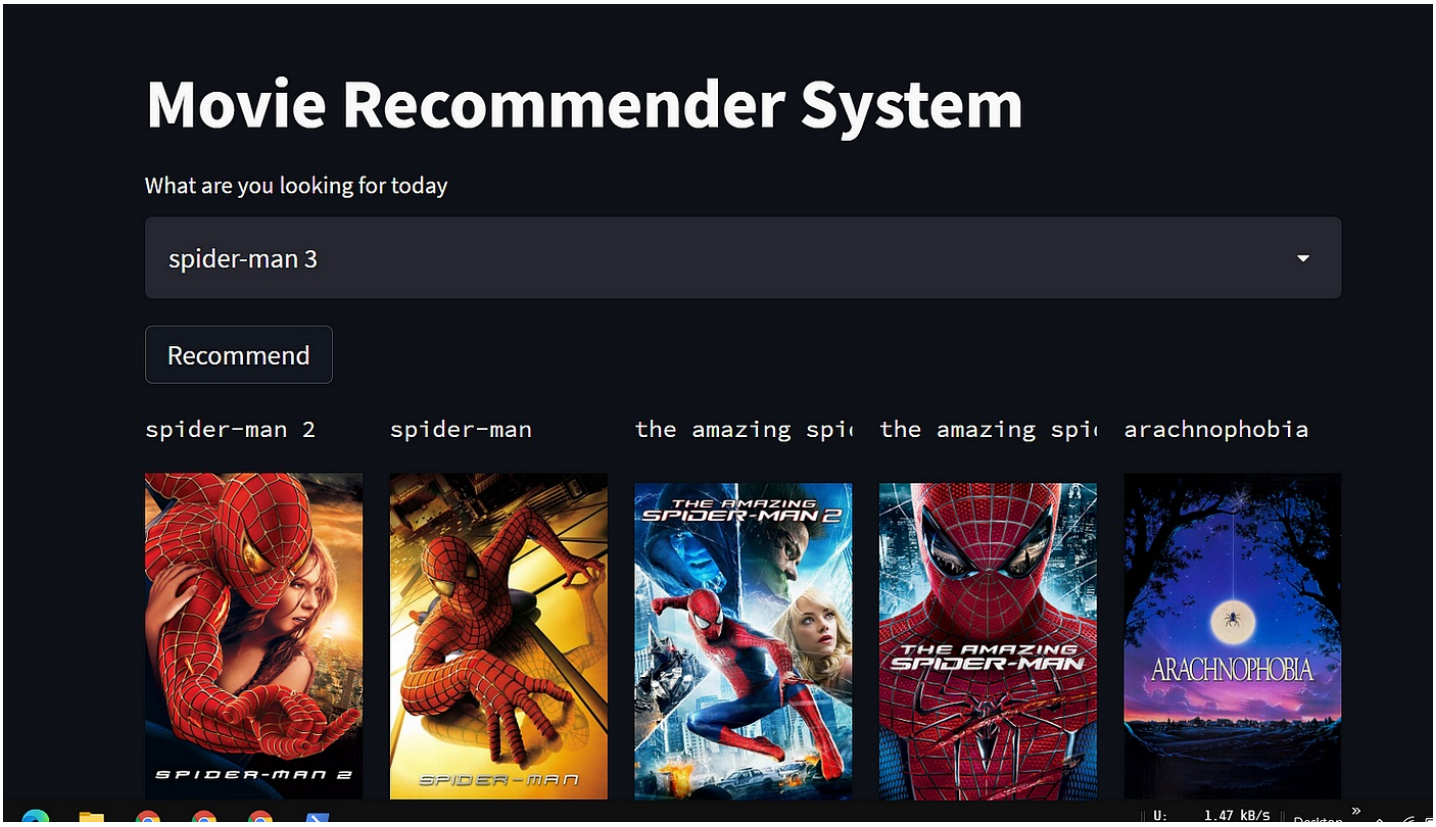
Today's Project: Movie Recommender System by Sachin Kumar.



In [1]:

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

In [2]:

```
credits = pd.read_csv('tmdb_5000_credits.csv')
movies = pd.read_csv('tmdb_5000_movies.csv')
```

In [3]:

```
credits.head(4)
```

Out[3]:

	movie_id	title	cast	crew
0	19995	Avatar	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	206647	Spectre	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	49026	The Dark Knight Rises	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...

In [4]:

```
movies.head(1)
```

Out[4]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popular

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popular
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.4375

In [163]:

```
# ['cast'] accesses the specific column named 'cast' in the DataFrame
# Each entry in the 'cast' column appears to contain JSON-like data
# .values converts the data in the 'cast' column from a pandas Series to a NumPy array

 # credits.head(1)['cast'].values
```

In [6]:

```
movies.shape
```

Out[6]:

```
(4803, 20)
```

In [7]:

```
credits.shape
```

Out[7]:

```
(4803, 4)
```

In [8]:

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

In [9]:

```
movies.shape
```

Out[9]:

```
(4809, 23)
```

In [10]:

```
movies.head(1)
```

Out[10]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popular
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.4375

1 rows × 23 columns

In [12]:

```
# fetch only wanted columsn

# genres
# id
# keywords
# title
# overview
```

```
# cast
# crew


# subset is typically used as the input for further processing in the project

movies = movies[['genres','movie_id','keywords','title','overview','cast','crew']]
```

In [14]:

```
movies.head()
```

Out[14]:

	genres	movie_id	keywords	title	overview	cast	crew
0	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	Spectre	A cryptic message from Bond’s past sends him o...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	[{"id": 28, "name": "Action"}, {"id": 80, "nam...	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	The Dark Knight Rises	Following the death of District Attorney Harve...	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	49529	[{"id": 818, "name": "based on novel"}, {"id":...	John Carter	John Carter is a war-weary, former military ca...	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

In [15]:

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

Out[15]:

```
genres      0
movie_id    0
keywords    0
title       0
overview    3
cast        0
crew        0
dtype: int64
```

In [16]:

```
# dropna() is a pandas DataFrame method used to remove rows or columns containing NaN (No
t a Number) values


movies.dropna(inplace=True)
```

In [17]:

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

Out[17]:

```
genres      0
movie_id    0
```

```
keywords    0
title       0
overview    0
cast        0
crew        0
dtype: int64
```

In [18]:

```
#Duplicate rows can skew the recommendations if they artificially increase the importance of certain movies

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

Out[18]:

```
0
```

In [19]:

```
# .iloc[] is used for integer-location based indexing to select rows & columns by their numerical position

movies.iloc[0]
```

Out[19]:

```
genres      [{"id": 28, "name": "Action"}, {"id": 12, "nam...
movie_id                                                19995
keywords    [{"id": 1463, "name": "culture clash"}, {"id":...
title                                                  Avatar
overview    In the 22nd century, a paraplegic Marine is di...
cast        [{"cast_id": 242, "character": "Jake Sully", "...
crew        [{"credit_id": "52fe48009251416c750aca23", "de...
Name: 0, dtype: object
```

In [20]:

```
movies.iloc[0].genres
```

Out[20]:

```
'[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]'
```

In [21]:

```
# '[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"},
#  {"id": 878, "name": "Science Fiction"}]'

# Convert above line into ['Action',''Adventure','Fantasy','Science Fiction'] so we make Helper function for this

# Helper Functions

#This function takes an input parameter obj, which is expected to be list of dictionaries,
# similar to the given JSON-like structure.
# Initialize an Empty List
# This loop iterates over each dictionary in the input list obj
# For each dictionary i, value corresponding to key 'name' is accessed using i['name'].
# extracted genre name is then appended to list L.


# def convert(obj):
 #   L = []
 #   for i in obj:
     #   L.append(i['name'])
  #  return L

# convert('[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "na
```

```
me": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]')

# TypeError: string indices must be integers, not 'str

#It ends up iterating over each character of the string obj (e.g., '[', '{', '"', etc.).
Then, when you try to access i['name'],
# Python throws the error because i is a single character from the string, not a dictiona
ry.
```

In [22]:

```
import json
```

In [23]:

```
def convert(obj):    #The convert function takes a single input, obj, which is expected t
o be a JSON string.
    obj = json.loads(obj)  # Convert JSON string into a Python object
    L = []
    for i in obj:
        L.append(i['name'])
    return L
```

In [24]:

```
# Input JSON string

data = '[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name"
: "Fantasy"}, {"id": 878, "name": "Science Fiction"}]'
```

In [25]:

```
convert(data)
```

Out[25]:

```
['Action', 'Adventure', 'Fantasy', 'Science Fiction']
```

In [26]:

```
movies['genres']
```

Out[26]:

```
0       [{"id": 28, "name": "Action"}, {"id": 12, "nam...
1       [{"id": 12, "name": "Adventure"}, {"id": 14, "...
2       [{"id": 28, "name": "Action"}, {"id": 12, "nam...
3       [{"id": 28, "name": "Action"}, {"id": 80, "nam...
4       [{"id": 28, "name": "Action"}, {"id": 12, "nam...
                              ...
4804    [{"id": 28, "name": "Action"}, {"id": 80, "nam...
4805    [{"id": 35, "name": "Comedy"}, {"id": 10749, "...
4806    [{"id": 35, "name": "Comedy"}, {"id": 18, "nam...
4807                                                   []
4808                  [{"id": 99, "name": "Documentary"}]
Name: genres, Length: 4806, dtype: object
```

In [27]:

```
#convert is applied to each element in the specified column of the movies

movies['genres'] = movies['genres'].apply(convert)
```

In [28]:

```
movies.head()
```

Out[28]:

	genres	movie_id	keywords	title	overview	cast	crew
	[Action		[{"id": 1463				

	genres	movie_id	keywords	title	overview	cast	crew
0	[Action, Adventure, Fantasy, Science Fiction]	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	[Adventure, Fantasy, Action]	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	[Action, Adventure, Crime]	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	Spectre	A cryptic message from Bond’s past sends him o...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	[Action, Crime, Drama, Thriller]	49026	[{"id": 849, "name": "dc comics"}, {"id": 853,...	The Dark Knight Rises	Following the death of District Attorney Harve...	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	[Action, Adventure, Science Fiction]	49529	[{"id": 818, "name": "based on novel"}, {"id":...	John Carter	John Carter is a war-weary, former military ca...	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

In [29]:

```
## Now convert same for keywords using Convert function

movies['keywords'] = movies['keywords'].apply(convert)
```

In [30]:

```
movies.head()
```

Out[30]:

	genres	movie_id	keywords	title	overview	cast	crew
0	[Action, Adventure, Fantasy, Science Fiction]	19995	[culture clash, future, space war, space colon...	Avatar	In the 22nd century, a paraplegic Marine is di...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	[Adventure, Fantasy, Action]	285	[ocean, drug abuse, exotic island, east india ...	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	[Action, Adventure, Crime]	206647	[spy, based on novel, secret agent, sequel, mi...	Spectre	A cryptic message from Bond’s past sends him o...	[{"cast_id": 1, "character": "James Bond", "cr...	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	[Action, Crime, Drama, Thriller]	49026	[dc comics, crime fighter, terrorist, secret i...	The Dark Knight Rises	Following the death of District Attorney Harve...	[{"cast_id": 2, "character": "Bruce Wayne / Ba...	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	[Action, Adventure, Science Fiction]	49529	[based on novel, mars, medallion, space travel...	John Carter	John Carter is a war-weary, former military ca...	[{"cast_id": 5, "character": "John Carter", "c...	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

In [164]:

```
## In Cast Columns we needs to fetch 3 star name from very long list {Importnat one}

#movies['cast'][0]
```

In [32]:

```
#A counter variable is initialized to keep track of how many movie names have been added
```

```
to the list.
# This ensures the function stops after processing the first 3 movies.
# The counter is incremented by 1 after each iteration to track how many movie names have
been added.

def convert3(obj):
    obj = json.loads(obj)
    L = []
    counter = 0
    for i in obj:
        if counter != 3:
            L.append(i['name'])
            counter+=1
        else:
            break
    return L
```

In [33]:

```
movies['cast'] = movies['cast'].apply(convert3)
```

In [34]:

```
movies.head(2)
```

Out[34]:

	genres	movie_id	keywords	title	overview	cast	crew
0	[Action, Adventure, Fantasy, Science Fiction]	19995	[culture clash, future, space war, space colon...	Avatar	In the 22nd century, a paraplegic Marine is di...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[{"credit_id": "52fe48009251416c750aca23", "de...
1	[Adventure, Fantasy, Action]	285	[ocean, drug abuse, exotic island, east india ...	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[{"credit_id": "52fe4232c3a36847f800b579", "de...
2	[Action, Adventure, Crime]	206647	[spy, based on novel, secret agent, sequel, mi...	Spectre	A cryptic message from Bond's past sends him o...	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[{"credit_id": "54805967c3a36829b5002c41", "de...
3	[Action, Crime, Drama, Thriller]	49026	[dc comics, crime fighter, terrorist, secret i...	The Dark Knight Rises	Following the death of District Attorney Harve...	[Christian Bale, Michael Caine, Gary Oldman]	[{"credit_id": "52fe4781c3a36847f81398c3", "de...
4	[Action, Adventure, Science Fiction]	49529	[based on novel, mars, medallion, space travel...	John Carter	John Carter is a war-weary, former military ca...	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[{"credit_id": "52fe479ac3a36847f813eaa3", "de...

In [165]:

```
# for crew we need only that name which department is director

 # movies['crew'][0]
```

In [36]:

```
# If the job is 'Director', the name of that crew member (i['name']) is added to the list
L.

def fetch_director(obj):
    obj = json.loads(obj)
    L = []
    for i in obj:
        if i['job'] == 'Director':
            L.append(i['name'])
            break
```

```
    return L
```

In [37]:

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

In [38]:

```
movies.head()
```

Out[38]:

	genres	movie_id	keywords	title	overview	cast	crew
0	[Action, Adventure, Fantasy, Science Fiction]	19995	[culture clash, future, space war, space colon...	Avatar	In the 22nd century, a paraplegic Marine is di...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[James Cameron]
1	[Adventure, Fantasy, Action]	285	[ocean, drug abuse, exotic island, east india ...	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[Gore Verbinski]
2	[Action, Adventure, Crime]	206647	[spy, based on novel, secret agent, sequel, mi...	Spectre	A cryptic message from Bond’s past sends him o...	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[Sam Mendes]
3	[Action, Crime, Drama, Thriller]	49026	[dc comics, crime fighter, terrorist, secret i...	The Dark Knight Rises	Following the death of District Attorney Harve...	[Christian Bale, Michael Caine, Gary Oldman]	[Christopher Nolan]
4	[Action, Adventure, Science Fiction]	49529	[based on novel, mars, medallion, space travel...	John Carter	John Carter is a war-weary, former military ca...	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[Andrew Stanton]

In [39]:

```
## Overview is a string so we need to convert this list

movies['overview'][0]
```

Out[39]:

```
'In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique m
ission, but becomes torn between following orders and protecting an alien civilization.'
```

In [40]:

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

In [41]:

```
movies.head()
```

Out[41]:

	genres	movie_id	keywords	title	overview	cast	crew
0	[Action, Adventure, Fantasy, Science Fiction]	19995	[culture clash, future, space war, space colon...	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[James Cameron]
1	[Adventure, Fantasy, Action]	285	[ocean, drug abuse, exotic island, east india ...	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[Gore Verbinski]
2	[Action, Adventure, Crime]	206647	[spy, based on novel, secret agent, sequel, mi...	Spectre	[A, cryptic, message, from, Bond’s, past, send...	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[Sam Mendes]
3	[Action, Crime, Drama, Thriller]	49026	[dc comics, crime fighter, terrorist, secret i...	The Dark Knight Rises	[Following, the, death, of, District, Attorney...	[Christian Bale, Michael Caine, Gary Oldman]	[Christopher Nolan]

	genres	movie_id	keywords	title	overview	cast	crew
4	[Action, Adventure, Science Fiction]	49529	[based on novel, mars, medallion, space travel...	John Carter	[John, Carter, is, a, war-weary,, former, mili...	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[Andrew Stanton]

In [42]:

```
# For genres we remove extra space , because they hamper recommendation also
# i.replace(" ", ""): The replace(" ", "") method is used to remove spaces in the string i.
# If a genre contains spaces (e.g., "Action Movie"), this will remove the spaces and make it "ActionMovie".

movies['genres'] = movies['genres'].apply(lambda x:[i.replace(" ","") for i in x])
```

In [43]:

```
movies.head()
```

Out[43]:

	genres	movie_id	keywords	title	overview	cast	crew
0	[Action, Adventure, Fantasy, ScienceFiction]	19995	[culture clash, future, space war, space colon...	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[Sam Worthington, Zoe Saldana, Sigourney Weaver]	[James Cameron]
1	[Adventure, Fantasy, Action]	285	[ocean, drug abuse, exotic island, east india ...	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[Johnny Depp, Orlando Bloom, Keira Knightley]	[Gore Verbinski]
2	[Action, Adventure, Crime]	206647	[spy, based on novel, secret agent, sequel, mi...	Spectre	[A, cryptic, message, from, Bond’s, past, send...	[Daniel Craig, Christoph Waltz, Léa Seydoux]	[Sam Mendes]
3	[Action, Crime, Drama, Thriller]	49026	[dc comics, crime fighter, terrorist, secret i...	The Dark Knight Rises	[Following, the, death, of, District, Attorney...	[Christian Bale, Michael Caine, Gary Oldman]	[Christopher Nolan]
4	[Action, Adventure, ScienceFiction]	49529	[based on novel, mars, medallion, space travel...	John Carter	[John, Carter, is, a, war-weary,, former, mili...	[Taylor Kitsch, Lynn Collins, Samantha Morton]	[Andrew Stanton]

In [44]:

```
movies['keywords'] = movies['keywords'].apply(lambda x:[i.replace(" ","") for i in x])
movies['cast'] = movies['cast'].apply(lambda x:[i.replace(" ","") for i in x])
movies['crew'] = movies['crew'].apply(lambda x:[i.replace(" ","") for i in x])
```

In [45]:

```
movies.head()
```

Out[45]:

	genres	movie_id	keywords	title	overview	cast	crew
0	[Action, Adventure, Fantasy, ScienceFiction]	19995	[cultureclash, future, spacewar, spacecolony, ...	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron]
1	[Adventure, Fantasy, Action]	285	[ocean, drugabuse, exoticisland, eastindiatrad...	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[JohnnyDepp, OrlandoBloom, KeiraKnightley]	[GoreVerbinski]
2	[Action, Adventure, Crime]	206647	[spy, basedonnovel, secretagent, sequel, mi6, ...	Spectre	[A, cryptic, message, from, Bond’s, past, send...	[DanielCraig, ChristophWaltz, LéaSeydoux]	[SamMendes]
3	[Action, Crime, Drama, Thriller]	49026	[dccomics, crimefighter, terrorist,	The Dark Knight Rises	[Following, the, death, of, District,	[ChristianBale, MichaelCaine,	[ChristopherNolan]

	genres	movie_id	keywords	title	overview	cast	crew
4	[Action, Adventure, ScienceFiction]	49529	[basedonnovel, mars, medallion, spacetravel, p...	John Carter	[John, Carter, is, a, war-weary,, former, mili...	[TaylorKitsch, LynnCollins, SamanthaMorton]	[AndrewStanton]

In [46]:

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

In [47]:

```
movies.head()
```

Out[47]:

	genres	movie_id	keywords	title	overview	cast	crew	tags
0	[Action, Adventure, Fantasy, ScienceFiction]	19995	[cultureclash, future, spacewar, spacecolony, ...	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[SamWorthington, ZoeSaldana, SigourneyWeaver]	[JamesCameron]	[In, the, 22nd, century,, a, paraplegic, Marin...
1	[Adventure, Fantasy, Action]	285	[ocean, drugabuse, exoticisland, eastindiatrad...	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[JohnnyDepp, OrlandoBloom, KeiraKnightley]	[GoreVerbinski]	[Captain, Barbossa,, long, believed, to, be, d...
2	[Action, Adventure, Crime]	206647	[spy, basedonnovel, secretagent, sequel, mi6, ...	Spectre	[A, cryptic, message, from, Bond’s, past, send...	[DanielCraig, ChristophWaltz, LéaSeydoux]	[SamMendes]	[A, cryptic, message, from, Bond’s, past, send...
3	[Action, Crime, Drama, Thriller]	49026	[dccomics, crimefighter, terrorist, secretiden...	The Dark Knight Rises	[Following, the, death, of, District, Attorney...	[ChristianBale, MichaelCaine, GaryOldman]	[ChristopherNolan]	[Following, the, death, of, District, Attorney...
4	[Action, Adventure, ScienceFiction]	49529	[basedonnovel, mars, medallion, spacetravel, p...	John Carter	[John, Carter, is, a, war-weary,, former, mili...	[TaylorKitsch, LynnCollins, SamanthaMorton]	[AndrewStanton]	[John, Carter, is, a, war-weary,, former, mili...

In [48]:

```
new_df = movies[['movie_id','title','tags']]

new_df.head()
```

Out[48]:

	movie_id	title	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...
2	206647	Spectre	[A, cryptic, message, from, Bond’s, past, send...
3	49026	The Dark Knight Rises	[Following, the, death, of, District, Attorney...
4	49529	John Carter	[John, Carter, is, a, war-weary,, former, mili...

In [49]:

```
## Convert tags columns -{list into strings}
```

```
new_df['tags'] =new_df['tags'].apply(lambda x:" ".join(x))
```

```
C:\Users\sachin.kumar\AppData\Local\Temp\ipykernel_5052\4083836742.py:3: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  new_df['tags'] =new_df['tags'].apply(lambda x:" ".join(x))
```

In [50]:

```
new_df.head()
```

Out[50]:

	movie_id	title	tags
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...
2	206647	Spectre	A cryptic message from Bond’s past sends him o...
3	49026	The Dark Knight Rises	Following the death of District Attorney Harve...
4	49529	John Carter	John Carter is a war-weary, former military ca...

In [51]:

```
new_df['tags'][0]
```

Out[51]:

```
'In the 22nd century, a paraplegic Marine is dispatched to the moon Pandora on a unique m
ission, but becomes torn between following orders and protecting an alien civilization. A
ction Adventure Fantasy ScienceFiction cultureclash future spacewar spacecolony society s
pacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle lov
eaffair antiwar powerrelations mindandsoul 3d SamWorthington ZoeSaldana SigourneyWeaver J
amesCameron'
```

In [52]:

```
## convert tags into lowercase
# Lowercasing is common normalization step for textual data used in (NLP) tasks

new_df['tags'] = new_df['tags'].apply(lambda x:x.lower())
```

```
C:\Users\sachin.kumar\AppData\Local\Temp\ipykernel_5052\2693314476.py:4: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  new_df['tags'] = new_df['tags'].apply(lambda x:x.lower())
```

In [68]:

```
#(nltk) is a popular Python library for (NLP) tasks such as tokenization, stemming, lemma
tization

import nltk
```

In [69]:

```
#Porter Stemmer is an algorithm for removing suffixes from words, leaving the "stem" or r
oot form

from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
```

In [74]:

```
#function named stem that takes one input argument text
# function applies a stemming operation on each word using ps.stem(i)
# ps.stem(i) applies a stemming algorithm to the word i to reduce it to its root form.
# For example, words like "running" or "runner" might be reduced to the root "run".
#split() method is used on the text input to break it into individual words


def stem(text):
    y = []  ## Initialize an empty list to store the stemmed words

    for i in text.split():    ## Loop through each word in the input text
        y.append(ps.stem(i))      ## Apply stemming to each word and append to list `y`

    return " ".join(y)             ## Join the list of stemmed words back into a single
string and return it
```

In [80]:

```
new_df['tags'] = new_df['tags'].apply(stem)
```

```
C:\Users\sachin.kumar\AppData\Local\Temp\ipykernel_5052\3213734980.py:1: SettingWithCopyW
arning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_g
uide/indexing.html#returning-a-view-versus-a-copy
  new_df['tags'] = new_df['tags'].apply(stem)
```

In [54]:

```
new_df['tags'][0]
```

Out[54]:

```
'in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique m
ission, but becomes torn between following orders and protecting an alien civilization. a
ction adventure fantasy sciencefiction cultureclash future spacewar spacecolony society s
pacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle lov
eaffair antiwar powerrelations mindandsoul 3d samworthington zoesaldana sigourneyweaver j
amescameron'
```

Text Vectorization

In [81]:

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(max_features=5000,stop_words='english')
```

In [82]:

```
#CountVectorizer is used to convert text data into a matrix of token counts
# transform(): Converts the text data into a sparse matrix
# sparse matrix created by fit_transform() is memory-efficient but not easily interpretab
le
# The .toarray() method converts the sparse matrix into a dense NumPy array for easier ma
nipulation and visualization.


cv.fit_transform(new_df['tags']).toarray()
```

Out[82]:

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
```

```
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [83]:

```
# 4806 = movies & 5000 given by user in max features

cv.fit_transform(new_df['tags']).toarray().shape
```

Out[83]:

```
(4806, 5000)
```

In [84]:

```
vectors = cv.fit_transform(new_df['tags']).toarray()
```

In [85]:

```
vectors
```

Out[85]:

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=int64)
```

In [86]:

```
# Most likely Avatar movies

vectors[0]
```

Out[86]:

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [87]:

```
cv
```

Out[87]:

```
▾                    CountVectorizer
CountVectorizer(max_features=5000, stop_words='english')
```

In [88]:

```
# cv.get_feature_names_out() allows you to inspect the vocabulary of the processed text d
ata
# get_feature_names() has been deprecated in recent versions

cv.get_feature_names_out()
```

Out[88]:

```
array(['000', '007', '10', ..., 'zone', 'zoo', 'zooeydeschanel'],
      dtype=object)
```

In [89]:

```
len(cv.get_feature_names_out())
```

Out[89]:

```
5000
```

In [64]:

```
import nltk
```

In [76]:

```
#'in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique mission,
# but becomes torn between following orders and protecting an alien civilization.
# action adventure fantasy sciencefiction cultureclash future spacewar spacecolony society spacetravel
# futuristic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar
# powerrelations mindandsoul 3d samworthington zoesaldana sigourneyweaver jamescameron'
```

In [78]:

```
stem('in the 22nd century, a paraplegic marine is dispatched to the moon pandora on a unique mission, but becomes torn between following orders and protecting an alien civilization. action adventure fantasy sciencefiction cultureclash future spacewar spacecolony society spacetravel futuristic romance space alien tribe alienplanet cgi marine soldier battle loveaffair antiwar powerrelations mindandsoul 3d samworthington zoesaldana sigourneyweaver jamescameron')
```

Out[78]:

```
'in the 22nd century, a parapleg marin is dispatch to the moon pandora on a uniqu mission, but becom torn between follow order and protect an alien civilization. action adventur fantasi sciencefict cultureclash futur spacewar spacecoloni societi spacetravel futurist romanc space alien tribe alienplanet cgi marin soldier battl loveaffair antiwar powerrel mindandsoul 3d samworthington zoesaldana sigourneyweav jamescameron'
```

In [94]:

```
# metrics: submodule within scikit-learn provides functions to evaluate performance of ml models (e.g., accuracy, precision)
# pairwise: submodule inside metrics that contains functions to compute various pairwise distances or similarities between data points
# (e.g., Euclidean distance, cosine similarity, etc.
# cosine_similarity:computes the cosine similarity between two vectors.
# In the context of a recommender system, it is commonly used to measure the similarity between items (e.g., movies)
# Cosine similarity ranges from -1 (completely opposite) to 1 (identical), with 0 indicating no similarity.
# A higher cosine similarity means that the two vectors are more similar

from sklearn.metrics.pairwise import cosine_similarity
```

In [95]:

```
cosine_similarity(vectors)
```

Out[95]:

```
array([[1.        , 0.08346223, 0.0860309 , ..., 0.04499213, 0.        ,
        0.        ],
       [0.08346223, 1.        , 0.06063391, ..., 0.02378257, 0.        ,
        0.02615329],
       [0.0860309 , 0.06063391, 1.        , ..., 0.02451452, 0.        ,
        0.        ],
       ...,
       [0.04499213, 0.02378257, 0.02451452, ..., 1.        , 0.03962144,
        0.04229549],
       [0.        , 0.        , 0.        , ..., 0.03962144, 1.        ,
        0.08714204],
       [0.        , 0.02615329, 0.        , ..., 0.04229549, 0.08714204,
        1.        ]])
```

In [97]:

```
#cosine similarity= A·B/‖A‖‖B‖
# If vectors is an n×m n×m matrix, the output of cosine_similarity(vectors) will be an n×n n×n similarity matrix,
# meaning that .shape() will return a tuple (n, n)
```

```
cosine_similarity(vectors).shape
```

Out[97]:

```
(4806, 4806)
```

In [101]:

```
similarity = cosine_similarity(vectors)
```

In [102]:

```
similarity
```

Out[102]:

```
array([[1.        , 0.08346223, 0.0860309 , ..., 0.04499213, 0.        ,
        0.        ],
       [0.08346223, 1.        , 0.06063391, ..., 0.02378257, 0.        ,
        0.02615329],
       [0.0860309 , 0.06063391, 1.        , ..., 0.02451452, 0.        ,
        0.        ],
       ...,
       [0.04499213, 0.02378257, 0.02451452, ..., 1.        , 0.03962144,
        0.04229549],
       [0.        , 0.        , 0.        , ..., 0.03962144, 1.        ,
        0.08714204],
       [0.        , 0.02615329, 0.        , ..., 0.04229549, 0.08714204,
        1.        ]])
```

In [104]:

```
# similarity[0] retrieves the similarity of the first movie (or user) with all other movi
es (or users) in the system.

similarity[0]
```

Out[104]:

```
array([1.        , 0.08346223, 0.0860309 , ..., 0.04499213, 0.        ,
       0.        ])
```

In [105]:

```
similarity[1]
```

Out[105]:

```
array([0.08346223, 1.        , 0.06063391, ..., 0.02378257, 0.        ,
       0.02615329])
```

In [106]:

```
similarity[2]
```

Out[106]:

```
array([0.0860309 , 0.06063391, 1.        , ..., 0.02451452, 0.        ,
       0.        ])
```

In [125]:

```
#sorted(similarity[0], reverse=True) sorts the similarity scores in descending order.
# The highest values (which indicate the most similar movies) will come first.

# sorted(similarity[0],reverse=True)

#1.0000000000000002,
 #0.28676966733820225,
 #0.26901379342448517,
 # 0.2605130246476754,
 # 0.255608593705383, It disturb the entire indexing
```

In [111]:

```
# This is boolean indexing (or masking) in pandas.filters rows in new_df DataFrame where condition new_df['title'] == 'Avatar' is True.
# As a result, only the rows where movie title is 'Avatar' will be returned.

new_df[new_df['title'] == 'Avatar']
```

Out[111]:

	movie_id	title	tags
0	**19995**	**Avatar**	**in the 22nd century, a parapleg marin is dispa...**

In [115]:

```
#This accesses index of filtered DataFrame (new_df[new_df['title'] == 'Avatar']).
# In pandas, the index refers to row labels of DataFrame
#  In this case, 0 is the first index where 'Avatar' appears in the DataFrame.

new_df[new_df['title'] == 'Avatar'].index[0]
```

Out[115]:

```
0
```

In [166]:

```
# enumerate() function adds an index to each element in the list, making it easier to tra ck the original positions of elements.
# It sorts the list of tuples by the second element (x[1]), which is the similarity score .
# The key=lambda x: x[1] is sorting key function that tells Python to sort based on secon d item in each tuple (similarity score)

# sorted(list(enumerate(similarity[0])), key=lambda x: x[1], reverse=True)
```

In [134]:

```
sorted(list(enumerate(similarity[0])), key=lambda x: x[1], reverse=True)[1:6]
```

Out[134]:

```
[(1216, 0.28676966733820225),
 (2409, 0.26901379342448517),
 (3730, 0.2605130246476754),
 (507, 0.255608593705383),
 (539, 0.2503866978335957)]
```

 # function named recommend, which takes a single input parameter movie # Find the Movie Index: It finds the index of that movie in the DataFrame new_df. # Get Similarity Scores: It retrieves the similarity scores of the chosen movie with all other movies from the similarity matrix. # Sort Movies by Similarity: It sorts the movies based on how similar they are to the input movie and selects the top 5 most similar ones (excluding the movie itself). # Output: It prints the indices of the 5 most similar movies. def recommend(movie): movie_index = new_df[new_df['title'] == movie].index[0] distances = similarity[movie_index] movie_list = sorted(list(enumerate(distances)), key=lambda x: x[1], reverse=True)[1:6] for i in movie_list: print(i[0])

In [146]:

```
recommend('Avatar')
```

```
Aliens vs Predator: Requiem
Aliens
Falcon Rising
Independence Day
Titan A.E.
```

In [143]:

```
## Index se movie ka naam extract
```

```
#.iloc is a Pandas indexing function that allows you to access rows & columns by their in
teger position (zero-based indexing).

new_df.iloc[1216].title
```

Out[143]:

```
'Aliens vs Predator: Requiem'
```

In [145]:

```
def recommend(movie):
    movie_index = new_df[new_df['title'] == movie].index[0]
    distances = similarity[movie_index]
    movie_list = sorted(list(enumerate(distances)), key=lambda x: x[1], reverse=True)[1:
6]

    for i in movie_list:
        print(new_df.iloc[i[0]].title)
```

In [147]:

```
recommend('Batman')
```

```
Batman
Batman & Robin
Batman Begins
Batman Returns
The R.M.
```

In [148]:

```
recommend('Batman & Robin')
```

```
Batman
Batman
Batman Forever
The Dark Knight Rises
Batman Begins
```

In [149]:

```
# pickle module, which is used for serializing (saving) and deserializing (loading) Pytho
n objects
# pickle is used to serialize Python objects into byte stream (binary format) that can be
stored in file or transmitted over network.

import pickle
```

In [150]:

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

In [153]:

```
## in this array type of data is come

new_df['title'].values
```

Out[153]:

```
array(['Avatar', "Pirates of the Caribbean: At World's End", 'Spectre',
       ..., 'Signed, Sealed, Delivered', 'Shanghai Calling',
       'My Date with Drew'], dtype=object)
```

In [167]:

```
#to_dict() method in pandas converts the DataFrame into a dictionary.
# Each row or column in the DataFrame is represented as a key-value pair in the dictionar
y

# new_df.to_dict()
```

In [156]:

```
pickle.dump(new_df.to_dict(),open('movie_dict.pkl','wb'))
```

In [159]:

```
# similarity.pkl file can be loaded quickly, making the recommender system faster and mor
e user-friendly.

pickle.dump(similarity,open('similarity.pkl','wb'))
```

In [ ]:

```

```

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

def fetch_poster(movie_id):
    response = requests.get('https://api.themoviedb.org/3/movie/{}?
api_key=3b2cdf5f970802a64feac20aa4096fa1&language=en-US'.format(movie_id))
    data = response.json()
    #print(data)
    #return "https://image.tmdb.org/t/p/w500" + data['poster_path']
    if 'poster_path' in data:
        return "https://image.tmdb.org/t/p/w500" + data['poster_path']
    else:
        # Return a placeholder image if no poster is found
        return "https://via.placeholder.com/500x750?text=No+Poster+Available"




def recommend(movie):
    movie_index = movies[movies['title'] == movie].index[0]
    distances = similarity[movie_index]
    movie_list = sorted(list(enumerate(distances)), key=lambda x: x[1], reverse=True)[1:6]

    recommended_movies = []
    recommended_movies_posters = []

    for i in movie_list:
        movie_id = movies.iloc[i[0]].movie_id
        # Fetch poster from API
        recommended_movies.append(movies.iloc[i[0]].title)

        # Fetch poster from API
        recommended_movies_posters.append(fetch_poster(movie_id))
    return recommended_movies,recommended_movies_posters



movies_dict = pickle.load(open('movie_dict.pkl','rb'))
movies = pd.DataFrame(movies_dict)


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


st.title('Movie Recommender System')

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


selected_movie_name = st.selectbox(
    'How would you like to be contacted ?',
    movies['title'].values)

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



    with col1:
       st.text(names[0])
       st.image(posters[0])

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

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

    with col1:
        st.text(names[3])
        st.image(posters[3])

    with col1:
        st.text(names[4])
        st.image(posters[4])
```