Sri Lanka Institute of Information Technology

Information Retrieval and Web Analytics – IT3051

B.Sc. (Hons) in Information Technology

Movie Recommendation System – Final Report

Group Name: BrainLeftException

	Name with Initials	Registration Number	Contact Number	Email Address
1.	Meriyan S.M.D.S.A	IT21076916	0775876087	it21076916@my.sliit.lk
2.	Galappaththi A. I.	IT21076220	0713632251	it21076220@my.sliit.lk
3.	Thilakarathne D. I. S.	IT20109608	0775299676	it20109608@my.sliit.lk

DECLARATION

We hereby declare that the assignment submitted is original except for source material explicitly acknowledged. All the members of the group have read and checked that all parts of the piece of work, irrespective of whether they are contributed by individual members or all members as a group.

ABSTRACTION

Our project, "Building a Personalized Movie Recommendation Engine," addresses the contemporary challenge of information abundance. In this era, where choices abound, the need for tailored recommendations is evident. Our project aims to create an advanced recommendation engine that enhances user engagement by providing precise and captivating movie suggestions. To achieve this, we employ a multi-faceted approach, including data preprocessing, recommendation algorithms, and a user-friendly interface.

ACKNOWLEDGEMENT

The successful completion of this project would not have been possible without the invaluable support and guidance of several individuals and resources. We extend our sincere appreciation to Mr. Samadhi Chathuranga, our module lecturer, and assistant lecturers for their unwavering assistance and encouragement throughout the assignment. The lectures and course materials provided a solid foundation for conducting research and preparing the final submission.

We are also grateful to our colleagues who generously shared their insights and recommendations, contributing to the improvement of this report. The collaborative effort of our project team was fundamental to its successful execution.

1. Background

What has led to the essential status of recommender systems?

In our modern era of abundant choices, recommender systems have emerged as essential tools for enhancing user experiences. They act as personalized guides, helping users discover content or products that align with their preferences, from suggesting movies and shows to recommending social connections or online shopping items. The simple act of recommending popular items to everyone pales in comparison to the sophisticated algorithms employed by dedicated recommender systems.

From a business perspective, these systems are invaluable. The more personalized and accurate the recommendations, the more engaged users become, resulting in increased revenue for the platform. In some cases, as much as 35-40% of revenue for tech giants can be attributed to the effectiveness of their recommendation systems. This report will explore the different types of recommendation systems, shedding light on their workings and their profound influence on user engagement, satisfaction, and platform profitability. Understanding and harnessing these systems is key to delivering tailored, enjoyable experiences to users while simultaneously driving success for the platforms that employ them.

2. Choice of Technology

The project was developed using several key tools in Python, including Visual Studio Code, Jupyter Notebook, and Streamlit. Jupyter Notebook served as the primary platform for data preprocessing and model development, constituting the backend of the application. For the frontend, we leveraged Streamlit, a Python library that enabled us to create an interactive and user-friendly user interface. This integration of the frontend and backend was seamlessly achieved through Streamlit, enhancing the overall user experience.

Task	Technology
Model development	Python
Frontend Application Tool	Streamlit

3. Description of the original datasets

Movies Dataset

Variable name	Definition	Type of the variable
budget	The budget in which the movie was made.	Int
genre	The genre of the movie, Action, Comedy, Thriller etc.	Object
homepage	A link to the homepage of the movie.	Object
id	This is the movie_id as in the first dataset.	Int
keywords	The keywords or tags related to the movie.	Object
original_language	The language in which the movie was made.	Object
original_title	The title of the movie before translation or adaptation.	Object
overview	A brief description of the movie.	Object
popularity	A numeric quantity specifying the movie popularity.	Float
production_companies	The production house of the movie.	Object
production_countries	The country in which it was produced.	Object
release_date	The date on which it was released.	Object
revenue	The worldwide revenue generated by the movie.	Int
runtime	The running time of the movie in minutes.	Float
status	"Released" or "Rumored".	Object
tagline	Movie's tagline.	Object
title	Title of the movie.	Object
vote_average	average ratings the movie recieved.	Float
vote_count	the count of votes recieved.	Int

Credits Dataset

Variable name	Definition	Type of the variable
movie_id	A unique identifier for each movie.	Int
title		Object
cast	The name of lead and supporting actors.	Object
crew	The name of Director, Editor, Composer, Writer etc.	Object

Dataset Link: https://www.kaggle.com/datasets/tmdb/tmdb-movie-metadata/data

4. Data Identification

- Data collection and identification of the features of data

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

Python

+ Code + Markdown

movies.info(), credits.info() provides essential information about the dataset containing movie details and credit details . This function is useful for a quick overview of the dataset's structure and size.

In [4]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 20 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   budget                4803 non-null   int64
 1   genres                4803 non-null   object
 2   homepage              1712 non-null   object
 3   id                    4803 non-null   int64
 4   keywords              4803 non-null   object
 5   original_language     4803 non-null   object
 6   original_title        4803 non-null   object
 7   overview              4800 non-null   object
 8   popularity            4803 non-null   float64
 9   production_companies  4803 non-null   object
 10  production_countries  4803 non-null   object
 11  release_date          4802 non-null   object
 12  revenue               4803 non-null   int64
 13  runtime               4801 non-null   float64
 14  spoken_languages      4803 non-null   object
 15  status                4803 non-null   object
 16  tagline               3959 non-null   object
 17  title                 4803 non-null   object
 18  vote_average          4803 non-null   float64
 19  vote_count            4803 non-null   int64
dtypes: float64(3), int64(4), object(13)
memory usage: 750.6+ KB
```

In [5]:
```
credits.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4803 entries, 0 to 4802
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   movie_id  4803 non-null   int64
 1   title     4803 non-null   object
 2   cast      4803 non-null   object
 3   crew      4803 non-null   object
dtypes: int64(1), object(3)
memory usage: 150.2+ KB
```

```
In [8]: movies.head(3)
```

Out[8]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularity	production_comp
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...	150.437577	[{"name": "Inge Film Partners
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "...	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	en	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	139.082615	[{"name": "Walt I Pictures", "id": :
2	245000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.sonypictures.com/movies/spectre/	206647	[{"id": 470, "name": "spy"}, {"id": 818, "name...	en	Spectre	A cryptic message from Bond's past sends him o...	107.376788	[{"name": "Col Pictures", "i {"

```
credits.shape
```

Python

```
(4803, 4)
```

```
movies.columns
```

Python

```
Index(['budget', 'genres', 'homepage', 'id', 'keywords', 'original_language',
       'original_title', 'overview', 'popularity', 'production_companies',
       'production_countries', 'release_date', 'revenue', 'runtime',
       'spoken_languages', 'status', 'tagline', 'title', 'vote_average',
       'vote_count', 'movie_id', 'cast', 'crew'],
      dtype='object')
```

5. Data Preprocessing

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

Python

Movies, Credits, the two datasets will be merged based on the ‘title’ column to streamline and simplify the data for more convenient analysis and insights.

```python
movies['original_language'].value_counts()

#Majority movies are english movies and this column is irelevant for our system
```
Python

```
en    4510
fr      70
es      32
zh      27
de      27
hi      19
ja      16
it      14
ko      12
cn      12
ru      11
pt       9
da       7
sv       5
nl       4
fa       4
th       3
he       3
ta       2
cs       2
ro       2
id       2
ar       2
vi       1
sl       1
...
is       1
xx       1
te       1
el       1
Name: original_language, dtype: int64
```
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

- In here it has run to gain an idea on the influence of different languages to the data.
- Since the majority are in English, this column is considered irrelevant for our recommendation system.

```python
# Keeping important columns for recommendation
movies = movies[['movie_id','title','overview','genres','keywords','cast','crew']]
```
Python

- After a careful observation on the columns of the merged dataset it was clear that, 'movie_id,' 'title,' 'overview,' 'genres,' 'keywords,' 'cast,' and 'crew', are only only the essential columns for the recommendation system.

```python
movies.head()
```
Python

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

- A quick overview on the selected columns to further examine the preprocessing steps.

```python
movies.shape
```
Python

```
(4809, 7)
```

- To obtain a view of the dataset's dimensions, which includes the number of rows and columns. This is a useful step to assess the size of the merged dataset.

Checking null values

```python
#handle list of dictionaries inside columns
movies.isnull().sum()
```

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

In data preprocessing, checking for missing values is a crucial step. Addressing missing data is crucial to ensure the quality and completeness of our dataset for robust recommendation system development. By identifying and handling missing values, we aim to enhance the accuracy and reliability of our recommendations, providing users with a more seamless experience.

- The output of "movies.isnull().sum()" reveals that the 'overview' column has three missing values.

```python
movies.dropna(inplace = True)
```

- Ensuring dataset's quality, the code "movies.dropna(inplace=True)" is used to remove rows with missing values from the "movies" dataset.

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

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

Now it has no missing values in the dataset.

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

```
0
```

This is used to count the number of duplicate rows within the "movies" dataset.

Genre column

```python
movies.iloc[0]['genres']
#type(movies.iloc[0]['genres']) #In genre column there has the id too, should keep only the 'genre'.
```

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

In the 'genres' column of our dataset, it has come to our attention that it contains both genre names and their corresponding IDs. To ensure data consistency and relevance for our recommendation system, we have opted to retain only the 'genre' names, which are integral to our analysis.

```python
#Since this is a string it should convert to a list and then the id can be removed

import ast #A function that help to :convert a string to a list

def convert_genre_column(text):

    l = []                          #an empty list
    for i in ast.literal_eval(text):   #helps to convert a string to list
        l.append(i['name'])

    return l
```

- This function effectively converts the 'genres' column, originally in string format, into a list.
- By doing so, it allows us to remove genre IDs while keeping the 'genre' names.

  This conversion is important to ensure the accuracy and relevance of the genre-based recommendations in the recommendation system.

```python
movies['genres'] = movies['genres'].apply(convert_genre_column)
```

- Applies the function.

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

```
0       [Action, Adventure, Fantasy, Science Fiction]
1                      [Adventure, Fantasy, Action]
2                        [Action, Adventure, Crime]
3                   [Action, Crime, Drama, Thriller]
4               [Action, Adventure, Science Fiction]
                            ...
4804                      [Action, Crime, Thriller]
4805                              [Comedy, Romance]
4806              [Comedy, Drama, Romance, TV Movie]
4807                                             []
4808                                  [Documentary]
Name: genres, Length: 4806, dtype: object
```

```python
#Check converted genre column
movies.head(2)
```

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[Action, Adventure, Fantasy, Science Fiction]	[{"id": 1463, "name": "culture clash"}, {"id":...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[Adventure, Fantasy, Action]	[{"id": 270, "name": "ocean"}, {"id": 726, "na...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...

- Now that the 'genre' column has been appropriately formatted, it is ready for use in our recommendation system.

- We will apply similar steps to format the 'keyword' column, ensuring consistency and relevancy in our dataset.

```python
movies.iloc[0]['keywords'] #Same as genre column, should remove the unnessary id
```


'[{"id": 1463, "name": "culture clash"}, {"id": 2964, "name": "future"}, {"id": 3386, "name": "space war"}, {"id": 3388, "name": "space colony"}, {"id": 3679, '

```python
import ast #A function that help to :convert a string to a list

def convert_keyword_column(text):

    l = []
    for i in ast.literal_eval(text):   #helps to convert a string to list
        l.append(i['name'])

    return l
```
Python

```python
movies['keywords'] = movies['keywords'].apply(convert_keyword_column)
```
Python

```python
movies['keywords']
```
Python

```
0       [culture clash, future, space war, space colon...
1       [ocean, drug abuse, exotic island, east india ...
2       [spy, based on novel, secret agent, sequel, mi...
3       [dc comics, crime fighter, terrorist, secret i...
4       [based on novel, mars, medallion, space travel...
                              ...
4804    [united states–mexico barrier, legs, arms, pap...
4805                                                   []
4806    [date, love at first sight, narration, investi...
4807                                                   []
4808            [obsession, camcorder, crush, dream girl]
Name: keywords, Length: 4806, dtype: object
```

```python
movies.head(2)
```
Python

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[culture clash, future, space war, space colon...	[culture clash, future, space war, space colon...	[{"cast_id": 242, "character": "Jake Sully", "...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[ocean, drug abuse, exotic island, east india ...	[ocean, drug abuse, exotic island, east india ...	[{"cast_id": 4, "character": "Captain Jack Spa...	[{"credit_id": "52fe4232c3a36847f800b579", "de...

```python
def convert_cast_column(text):

    l = [] #an empty list
    counter = 0
    for i in ast.literal_eval(text):   #helps to convert a string to list
        if counter < 3:
            l.append(i['name'])


    return l
```
Python

```python
movies['cast'] = movies['cast'].apply(convert_cast_column)
```
Python

- In the original dataset, the 'cast' column contained unsuitable information. To facilitate further processing and align with the requirements of our system, we made the decision to represent only the first three cast members.
- Above function is written to convert the column's string representation into a list, allowing us to extract and retain the names of up to three cast members.

```python
movies['cast']
```
Python

```
0       [Sam Worthington, Zoe Saldana, Sigourney Weave...
1       [Johnny Depp, Orlando Bloom, Keira Knightley, ...
2       [Daniel Craig, Christoph Waltz, Léa Seydoux, R...
3       [Christian Bale, Michael Caine, Gary Oldman, A...
4       [Taylor Kitsch, Lynn Collins, Samantha Morton,...
                              ...
4804    [Carlos Gallardo, Jaime de Hoyos, Peter Marqua...
4805    [Edward Burns, Kerry Bishé, Marsha Dietlein, C...
4806    [Eric Mabius, Kristin Booth, Crystal Lowe, Geo...
4807    [Daniel Henney, Eliza Coupe, Bill Paxton, Alan...
4808    [Drew Barrymore, Brian Herzlinger, Corey Feldm...
Name: cast, Length: 4806, dtype: object
```

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

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	In the 22nd century, a paraplegic Marine is di...	[culture clash, future, space war, space colon...	[culture clash, future, space war, space colon...	[Sam Worthington, Zoe Saldana, Sigourney Weave...	[{"credit_id": "52fe48009251416c750aca23", "de...
1	285	Pirates of the Caribbean: At World's End	Captain Barbossa, long believed to be dead, ha...	[ocean, drug abuse, exotic island, east india ...	[ocean, drug abuse, exotic island, east india ...	[Johnny Depp, Orlando Bloom, Keira Knightley, ...	[{"credit_id": "52fe4232c3a36847f800b579", "de...

```
# handle crew

movies.iloc[0]['crew']
```

```
'[{"credit_id": "52fe48009251416c750aca23", "department": "Editing", "gender": 0, "id": 1721, "job": "Editor", "name": "Stephen E. Rivkin"}, {"credit_id": "539c
```

As for the output for crew column, it is vivid that cast column is contained with some unneccasary data.

```
def fetch_director(text):
    L = []
    for i in ast.literal_eval(text):
        if i['job'] == 'Director':
            L.append(i['name'])
            break
    return L
```

- 'crew' column originally contained various crew members and their roles.
- To refine the data for our recommendation system, we have implemented the 'fetch director' function and it extracts the name of the director, a critical figure in a movie's production which will be important to the recommendation system.

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

```
movies['crew']
```

```
0           [James Cameron]
1          [Gore Verbinski]
2             [Sam Mendes]
3       [Christopher Nolan]
4          [Andrew Stanton]
              ...
4804    [Robert Rodriguez]
4805        [Edward Burns]
4806         [Scott Smith]
4807         [Daniel Hsia]
4808    [Brian Herzlinger]
Name: crew, Length: 4806, dtype: object
```

```
movies.head()
```

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

```python
# handle overview (converting to list)

movies.iloc[0]['overview']
```

Python

```
'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 ali
```

- a lambda function to the 'overview' column of the dataset. The lambda function uses the split() method to break each overview text into individual words.

```python
#split the paragraph into words by commas
movies['overview']=movies['overview'].apply(lambda x:x.split())
movies.head()
```

[42] Python

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

- The objective of the above step is to tokenize the paragraph in the 'overview' column into words. In this case, the code splits the overview text into a list of words.

```python
'Anna Kendrick'
'AnnaKendrick'

#Space between the words should be removed - in a situation of the same word being in few places

def remove_space(L):
    L1 = []
    for i in L:
        L1.append(i.replace(" ",""))
    return L1
```

Python

Above step is a crucial step that removes spaces between words in situations where the same word appears in different places within the dataset, ensuring uniformity and enhancing data accuracy.

```python
movies['cast'] = movies['cast'].apply(remove_space)
movies['crew'] = movies['crew'].apply(remove_space)
movies['genres'] = movies['genres'].apply(remove_space)
movies['keywords'] = movies['keywords'].apply(remove_space)
```

Python

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

Python

	movie_id	title	overview	genres	keywords	cast	crew
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[cultureclash, future, spacewar, spacecolony, ...	[cultureclash, future, spacewar, spacecolony, ...	[SamWorthington, ZoeSaldana, SigourneyWeaver, ...	[JamesCameron]
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[ocean, drugabuse, exoticisland, eastindiatrad...	[ocean, drugabuse, exoticisland, eastindiatrad...	[JohnnyDepp, OrlandoBloom, KeiraKnightley, Ste...	[GoreVerbinski]

```python
# Concatenate all
movies['tags'] = movies['overview'] + movies['genres'] + movies['keywords'] + movies['cast'] + movies['crew']
```

Python

- Concatenates data from multiple columns in the "movies" dataset into a new 'tags' column, forming a comprehensive set of descriptors for use in our recommendation system.

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

Python

	movie_id	title	overview	genres	keywords	cast	crew	tags
0	19995	Avatar	[In, the, 22nd, century,, a, paraplegic, Marin...	[cultureclash, future, spacewar, spacecolony, ...	[cultureclash, future, spacewar, spacecolony, ...	[SamWorthington, ZoeSaldana, SigourneyWeaver, ...	[JamesCameron]	[In, the, 22nd, century,, a, paraplegic, Marin...
1	285	Pirates of the Caribbean: At World's End	[Captain, Barbossa,, long, believed, to, be, d...	[ocean, drugabuse, exoticisland, eastindiatrad...	[ocean, drugabuse, exoticisland, eastindiatrad...	[JohnnyDepp, OrlandoBloom, KeiraKnightley, Ste...	[GoreVerbinski]	[Captain, Barbossa,, long, believed, to, be, d...

```python
movies.iloc[0]['tags']
```

Python

```
['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',
...
 'AliciaVela-Bailey',
 'RichardWhiteside',
 'NikieZambo',
 'JuleneRenee',
 'JamesCameron']
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Remove unnecessary columns.

```python
new_df=movies[['movie_id','title','tags']]
new_df.head()
```

[50] Python

	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...

After concatenating, the overview, cast, crew, and genre columns will be removed since we concatenated those to the tags column.

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

[51] Python

```
C:\Users\94775\AppData\Local\Temp\ipykernel_41016\684433085.py:1: SettingWithCopyWarning:
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_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['tags']=new_df['tags'].apply(lambda x: " ".join(x))
```

The code transforms the lists within the 'tags' column of the new data frame ,'new_df' into strings by using the join method.

```
[52]  new_df.head()
```

	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...

```
[54]  new_df.iloc[0]['tags']

'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 /
```

Convert to lowercase.

```
#convert to lowercase
new_df['tags']=new_df['tags'].apply(lambda x:x.lower())
new_df.head(2)
[55]

C:\Users\94775\AppData\Local\Temp\ipykernel_41016\4123649750.py:2: SettingWithCopyWarning:
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_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['tags']=new_df['tags'].apply(lambda x:x.lower())
```

	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...

```
new_df.iloc[0]['tags']

'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 ali
```

```
new_df.head()
```

	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...

Summarization on what's next: we'll discuss the essential steps taken to prepare our movie dataset for content-based recommendations. Additionally, we'll explore how cosine similarity is utilized to measure the likeness between movies, a key component in content-based recommendation systems.

Stemming

- Integration of NLTK and Porter Stemmer for Text Processing

```
#perform stemming
#now can apply count vectorizer since all the columns were preprocessed
import nltk
from nltk.stem import PorterStemmer
[58]

ps=PorterStemmer()
[59]
```

Utilizing NLTK and Porter Stemmer:

- NLTK Library: NLTK serves as the cornerstone of our text processing and analysis endeavors. It equips us with a comprehensive set of tools and resources for working with textual data, enabling us to delve deeper into text-based analysis.
- Porter Stemmer: We've specifically incorporated the Porter Stemmer, a well-established algorithm for stemming words. Stemming, the process of reducing words to their root form, is instrumental in simplifying text data for more precise analysis.

Create a function to perform stemming.

```
def stems(text):
    l=[]
    for i in text.split():
        l.append(ps.stem(i))
    return " ".join(l)
```
[60] Python

For providing users with precise and relevant movie recommendations, stem function plays a pivotal role in our text processing and analysis efforts, and it is expertly tailored to convert paragraphs or text into a list of words.

Functionality and Significance:

- Converting Text to Words: The 'stems' function expertly breaks down text into individual words, which is a foundational step in text-based analysis.
- Porter Stemmer Integration: By applying the Porter Stemmer, we standardize and simplify the text data, transforming words into their root forms, which is essential for precision in text analysis.

```
new_df['tags']=new_df['tags'].apply(stems)
```
[61] Python

```
C:\Users\94775\AppData\Local\Temp\ipykernel_41016\1522715013.py:1: SettingWithCopyWarning:
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_guide/indexing.html#returning-a-view-versus-a-copy
  new_df['tags']=new_df['tags'].apply(stems)
```

- Applies the custom 'stems' function to the 'tags' column in the 'new_df' dataframe. This operation converts the text within the 'tags' column to its stemmed form, reducing words to their root forms,

Tags column after applying the stem function.

```
new_df.iloc[0]['tags']
```
[62] Python

```
'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 fantas
```

- Perform Count Vectorizer

Utilizing Count Vectorization:

- Count Vectorizer Implementation: We have instantiated the Count Vectorizer as 'cv' with a maximum of 5000 features and the inclusion of English stop words. This vectorizer is a pivotal component in transforming our text data into numerical format for further analysis.

Objective and Benefits:

- The implementation of Count Vectorization is aimed at converting our text-based 'tags' data into a numerical representation. This transformation is critical for mathematical analysis, such as calculating cosine similarity, and is a significant step in the process of making content-based movie recommendations. The inclusion of English stop words further streamlines this text-to-numeric conversion.

```
[63]
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=5000,stop_words='english')
```

```
[64]
vector=cv.fit_transform(new_df['tags']).toarray()
```

The code "vector = cv.fit_transform(new_df['tags']).toarray()" represents the Count Vectorizer, previously configured as 'cv,' to transform the textual data in the 'tags' column of the 'new_df' dataframe into a numerical format, represented as a NumPy array.

```
[65]
vector.shape
... (4806, 5000)
```

- Provides information about the shape of the 'vector' NumPy array, which represents the transformed and numerical data derived from the Count Vectorization. It reveals the dimensions, indicating the number of rows and columns in the array.

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

- Calculates the number of unique feature names extracted by the Count Vectorizer. These feature names represent the words or terms from the text data that have been transformed into numerical features.

Cosine Similarity

To make content-based movie recommendations, it's essential to measure the similarity between movies based on their textual descriptors. In this context, we've applied the cosine similarity metric, which is a critical component of our recommendation system.

Cosine Similarity Calculation:

- Utilizing scikit-learn: We've employed the cosine_similarity function from scikit-learn to calculate similarity scores. This function operates on the numerical representation of movie descriptors derived from the Count Vectorization process.
- How Cosine Similarity Works: Cosine similarity quantifies the cosine of the angle between two vectors, which, in our case, represent the descriptor vectors of movies. Higher similarity scores indicate that two movies share more common descriptors, indicating a closer match in content.

```
[66]
from sklearn.metrics.pairwise import cosine_similarity
```

```
[67]
#gives all the similarities
similarity=cosine_similarity(vector)
```

Check the similarity between each word.

```
similarity
```

```
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.        ]])
```

```
similarity.shape
```

```
(4806, 4806)
```

Get the Recommendations

```
#this is how we can get the movie recommendations
new_df[new_df['title']=='Spider-Man'].index[0]
```

```
159
```

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

```
recommend('Spider-Man')
```

```
Spider-Man 3
Spider-Man 2
The Amazing Spider-Man 2
Arachnophobia
Kick-Ass
```

The function takes a user's selected movie as input and calculates the cosine similarity between that movie and all other movies in our dataset. The key steps include:

- User's Movie Input: The function begins by taking the user's choice of a movie as input. This is the reference point for generating recommendations.
- Index Identification: It identifies the index of the chosen movie in our dataset, allowing us to pinpoint its position for subsequent calculations.
- Similarity Calculation: Using the cosine similarity scores previously calculated, the function sorts of movies in the dataset by their similarity to the user's chosen movie.
- Recommendations Display: The top five movies with the highest similarity scores are presented as recommendations to the user. These recommendations are generated based on the content and style of the user's chosen movie.

Import trained model as a pickle file for create the web application.

```
#use this to create the system
import pickle
pickle.dump(new_df,open('artifacts/movie_list.pkl','wb'))
pickle.dump(similarity,open('artifacts/similarity.pkl','wb'))
```

User Rating Implementation

```
# Function to append user ratings to an existing CSV file
def add_user_ratings_to_csv(user_id, movie_id, rating, csv_file):
    new_data = {'userId': [user_id], 'movieId': [movie_id], 'rating': [rating]}
    new_ratings = pd.DataFrame(new_data)
```

```python
    # Append the new ratings to the existing CSV file
    existing_ratings = pd.read_csv(csv_file)
    updated_ratings = pd.concat([existing_ratings, new_ratings], ignore_index=True)
```

```python
    # Save the updated ratings to the CSV file
    updated_ratings.to_csv(csv_file, index=False)
```

```python
# Input user ratings for 5 movies and append to the existing CSV file
for i in range(5):
    user_id = int(input(f"Enter User ID for Movie {i+1}: "))
    movie_id = int(input(f"Enter Movie ID for Movie {i+1}: "))
    rating = float(input(f"Enter Rating for Movie {i+1}: "))

    # Append the user rating to the existing CSV file 'ratings_small.csv'
    add_user_ratings_to_csv(user_id, movie_id, rating, 'ratings_small.csv')
```

Get Top Rated Movies Based on the User Feedbacks

- Import necessary packages to implement the user feedback part

```python
import pickle
from surprise import Reader, Dataset, SVD
from surprise.model_selection import cross_validate, train_test_split
import pandas as pd
```

- Load the dataset, which is having the rating information of the user, to train the model to identify the top rated movies based on the user feedbacks

```python
reader = Reader()
ratings = pd.read_csv('ratings_small.csv')
ratings.head()

data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
```

- Split the dataset into train and test data to train the model, so that can use the model to recommend movies based on user feedbacks.

```python
trainset, testset = train_test_split(data, test_size=0.2)  # 80% training, 20% testing
```

- Singular value Decomposition (SVD) complex matrix has been used to train the ratings part

```python
svd = SVD()

cross_validate(svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)

trainset = data.build_full_trainset()

svd.fit(trainset)
```

- Defined a function to get recommended movies for a user according to their ratings

```python
# Define a function to get recommended movies for a user with a rating threshold
def get_top_rated_movies(user_id, threshold=3.5, num_recommendations=5):
    # Filter movies that the user has not rated
    movies_not_rated_by_user = [i for i in range(1, 193609) if not trainset.ur[trainset.to_inner_uid(user_id)]
                                or i not in [j[0] for j in trainset.ur[trainset.to_inner_uid(user_id)]]]

    # Predict ratings for all unrated movies for the user
    predicted_ratings = [(movie_id, svd.predict(user_id, movie_id).est) for movie_id in movies_not_rated_by_user]

    # Sort the predictions in descending order of estimated ratings
    predicted_ratings.sort(key=lambda x: x[1], reverse=True)

    # Get the top-rated movies above the threshold
    top_rated_movies = [movie for movie in predicted_ratings if movie[1] > threshold]

    return top_rated_movies[:num_recommendations]
```

- Check whether the trained model is working properly or not

```python
# Example usage:
user_id = 3
recommended_movies = get_top_rated_movies(user_id, threshold=3.5, num_recommendations=5)

print(f"Top 5 movies recommended for User {user_id} with ratings > 3.5:")
for movie in recommended_movies:
    print(f"Movie ID: {movie[0]}, Predicted Rating: {movie[1]}")
```

- Save the trained model, so that can be used the saved model later without rerun the notebook again and again.

```python
# Save the SVD model
with open('artifacts/svd_model.pkl', 'wb') as file:
    pickle.dump(svd, file)
```

```python
# Save the get_top_rated_movies function
with open('artifacts/recommendation_function.pkl', 'wb') as file:
    pickle.dump(get_top_rated_movies, file)
```

5. Web Application

- Imported necessary packages to enable the execution of the web application using Streamlit.

```python
app.py > ...
import pandas as pd
import pickle
import streamlit as st
import requests
```

- Defined a function to display posters for each movies which are going to display by the recommendation system.

```python
def fetch_poster(movie_id):
    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
```

- Defined a function to recommend top movies related to the selection of the user.

```python
def recommend(movie):
    index=movies[movies['title']==movie].index[0]
    distances=sorted(list(enumerate(similarity[index])),reverse=True,key=lambda x: x[1])
    recommended_movies_name=[]
    recommended_movies_poster=[]
    for i in distances[1:6]:
        movie_id=movies.iloc[i[0]]['movie_id']
        recommended_movies_poster.append(fetch_poster(movie_id))
        recommended_movies_name.append(movies.iloc[i[0]]['title'])
    return recommended_movies_name,recommended_movies_poster
```

- Set up the page configurations and added a sidebar with small description of how to use the application, so that users will be able to use the application easily along with the descriptions.

```python
# Set page configuration
st.set_page_config(page_title="Movie Recommender", page_icon="🎬", layout="wide")

# Load data and model
movies = pd.read_pickle(open('artifacts/movie_list.pkl','rb'))
similarity = pickle.load(open('artifacts/similarity.pkl','rb'))


# Sidebar with a title and background image
st.sidebar.image("images.jpg")
# Add CSS to make the image fill the sidebar width
st.sidebar.markdown(
    f'<style>div[data-testid="stSidebar"] div[data-testid="stBlock"] img {{width: 100%;}}</style>',
    unsafe_allow_html=True,
)
st.sidebar.title("🎬 How to use?")
# Add simplified instructions
st.sidebar.markdown("1. **Search for a Movie:** Start by typing the name of a movie you like in the search bar.")
st.sidebar.markdown("2. **Get Recommendations:** Click the 'Get Recommendations' button.")
st.sidebar.markdown("3. **Explore Similar Movies:** Discover and explore movies similar to your selection.")
st.sidebar.markdown("4. **Enjoy Your Movie Journey:** Search for more and enjoy your movie journey.")
st.sidebar.markdown("5. **Rate Your Experience:** Don't forget to share your experience about our recommendations.")
```

- Main content area which displays the main content of the web application including all the movies that were recommended by the system, according to the selection.

```python
# Main content area
st.header('Movie Recommender System Using ML')

movie_list = movies['title'].values
selected_movie = st.selectbox(
    "Choose a movie",
    movie_list
)

if st.button('Get Recommendations',key="recommend_button"):
    st.spinner("Finding recommendations...")
    recommended_movies_name,recommended_movies_poster=recommend(selected_movie)
    st.success("Here are some of the recommendations for your search:")

    # Display recommendations in a carousel
    col1,col2,col3,col4,col5=st.columns(5)
    cols = [col1, col2, col3, col4, col5]

    for i in range(5):
        with cols[i]:
            st.image(recommended_movies_poster[i], width=200)
            st.write(recommended_movies_name[i])

```

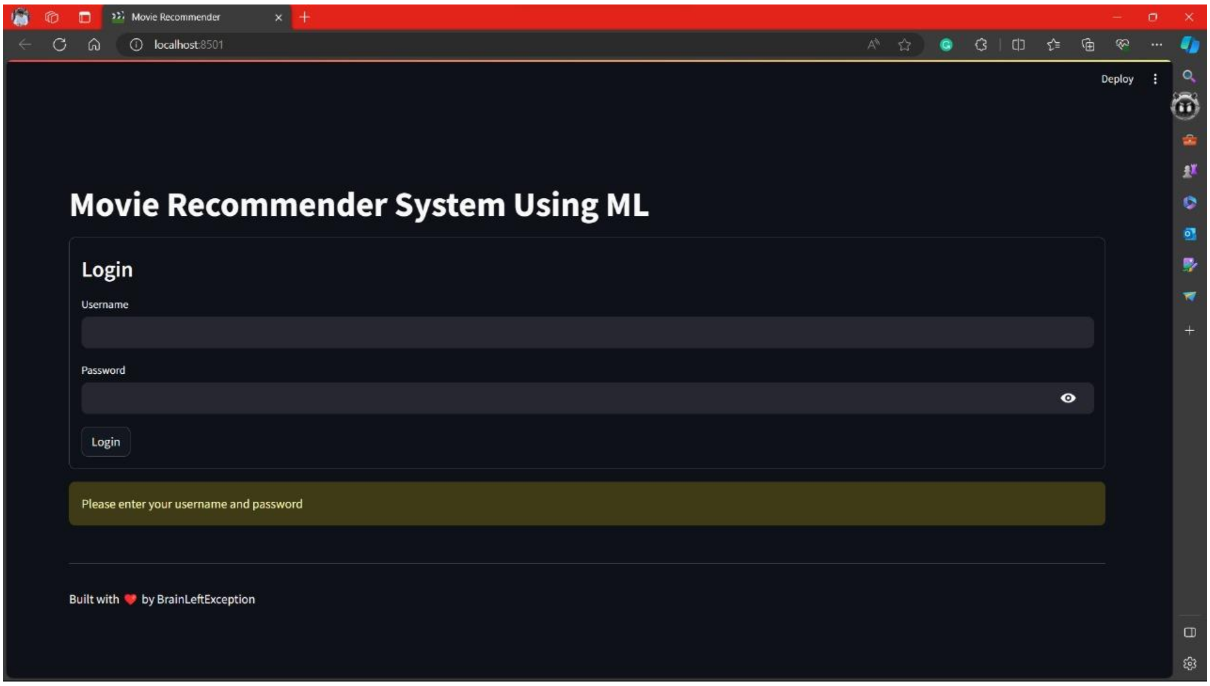
- Users are given a chance to rate the system based on their experiences throughout the application, and based on their ratings, application will recommend the content.

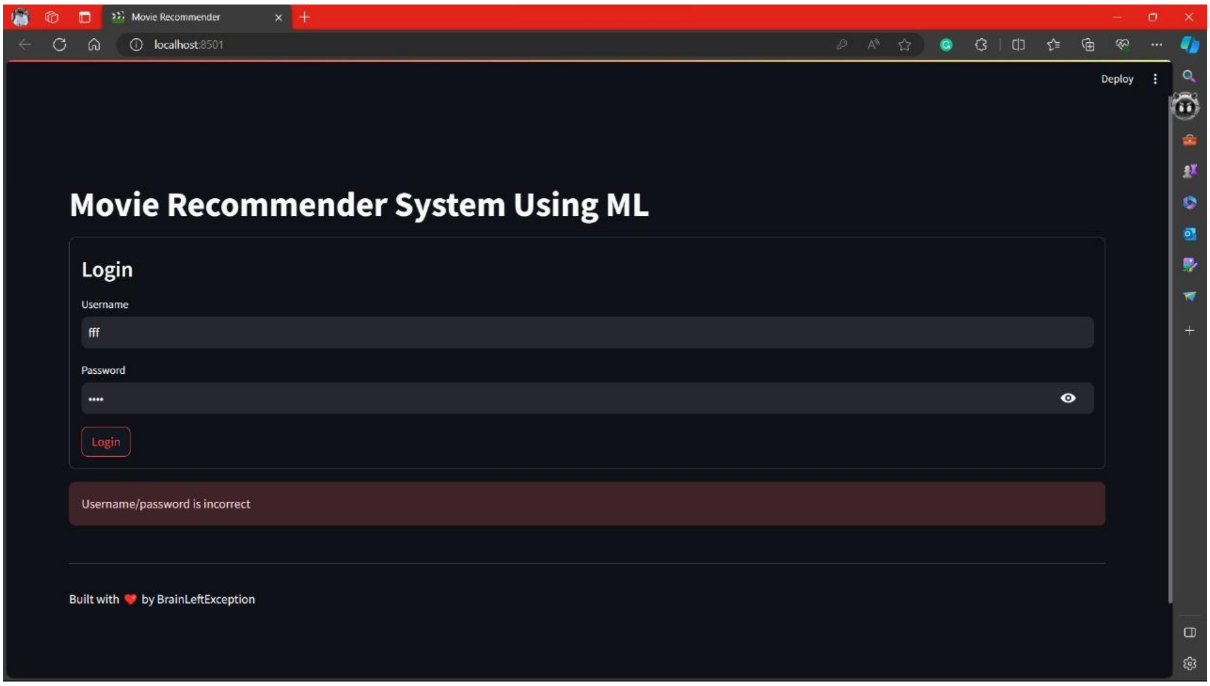
```python

    #add ratings
    from streamlit_star_rating import st_star_rating
    st_star_rating(label = "How would you rate us?", maxValue = 5, defaultValue = 3, key = "rating")
    # def function_to_run_on_click(value):
    #     st.write(f"**{value}** stars!")

    # st.write(stars)
```

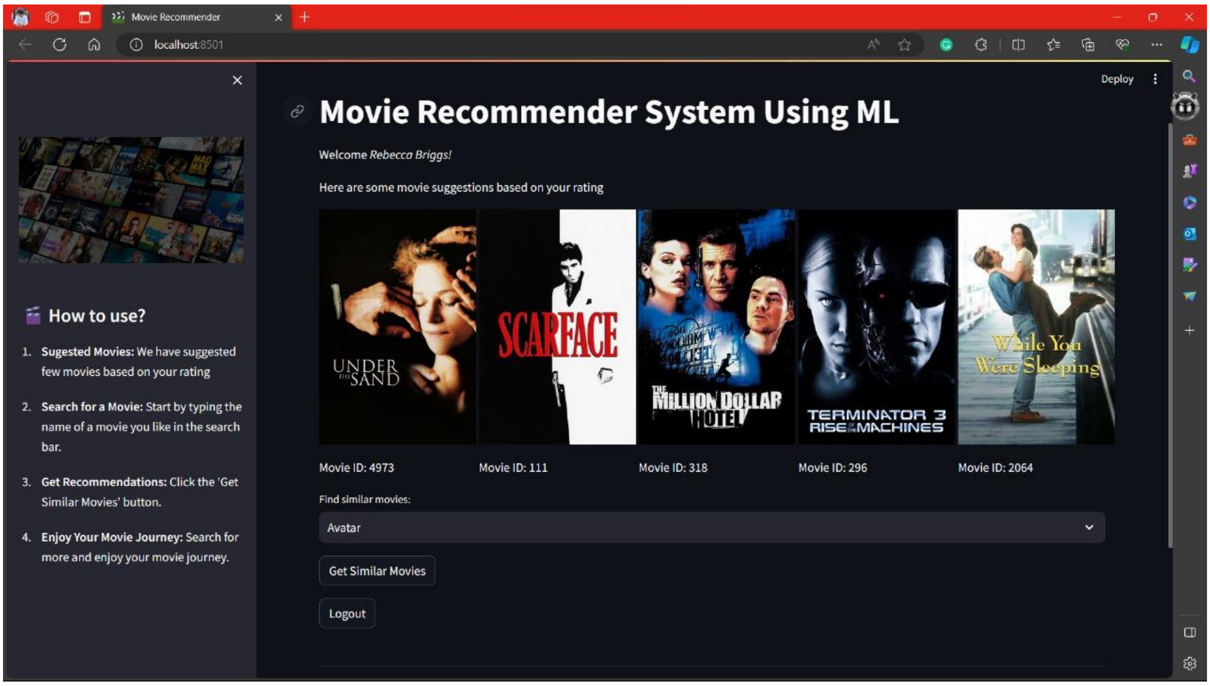
- Footer of the web application

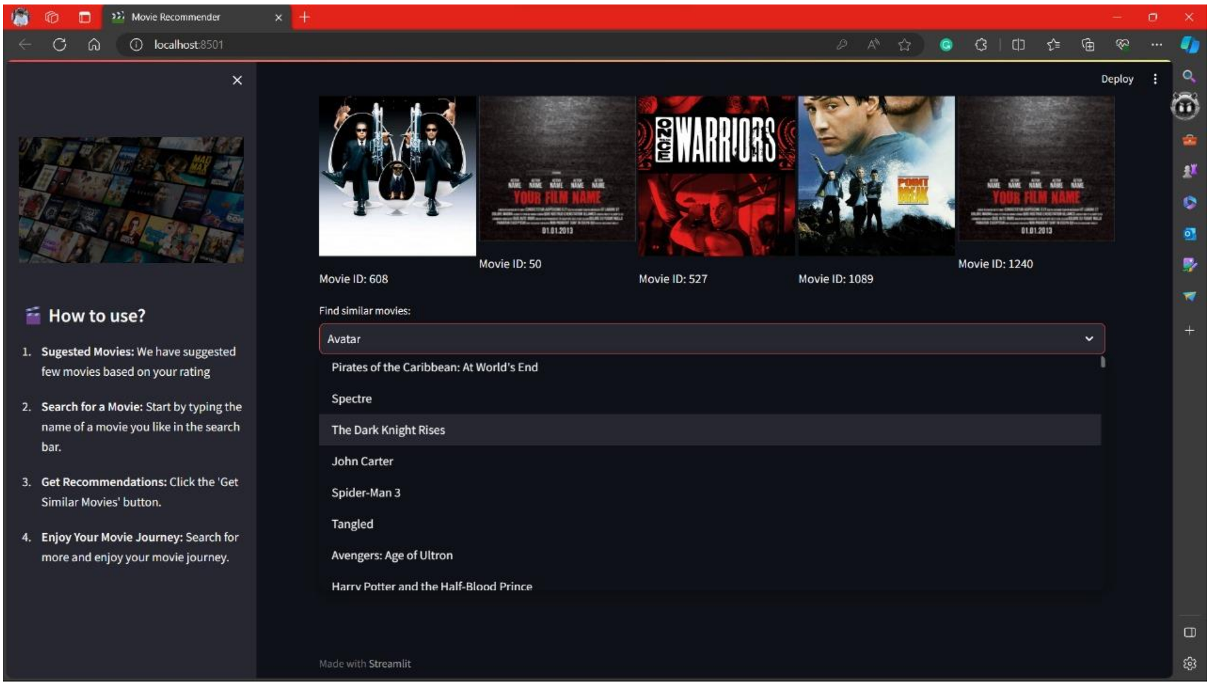
```python
# Add a custom footer
st.markdown("---")
st.write("Built with ❤️ by BrainLeftException")
```

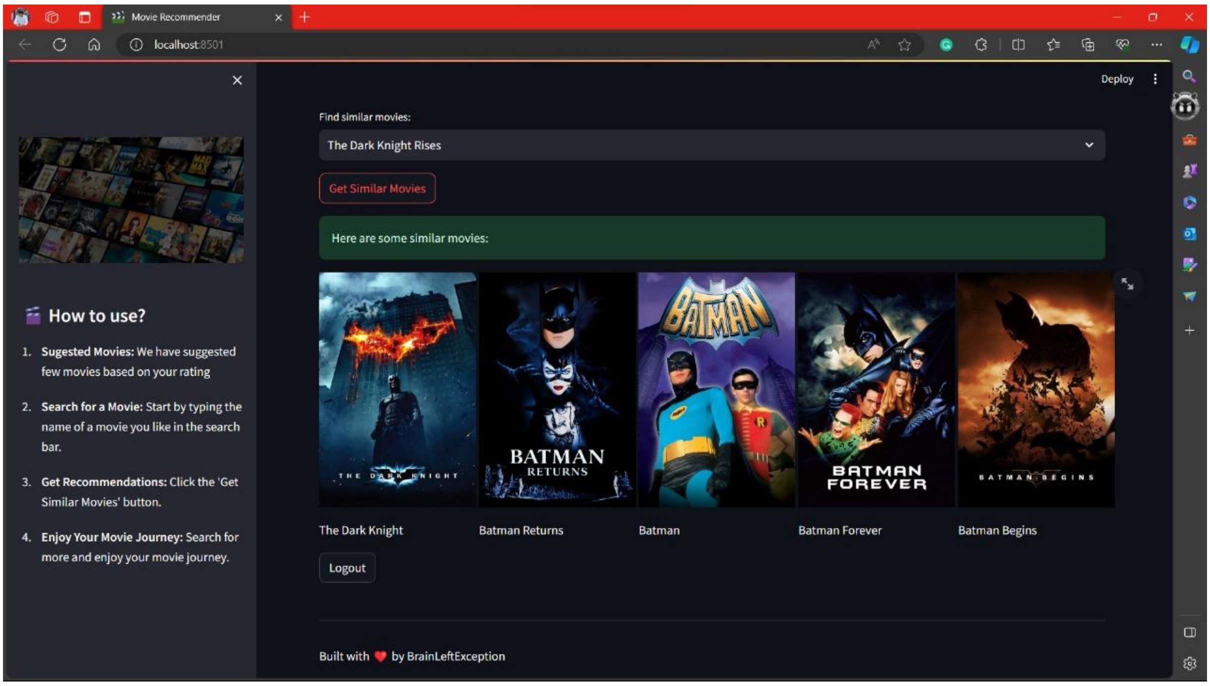
6. Web Application User Interface






Movie Recommender
localhost:8501
Deploy
Movie Recommender System Using ML
Welcome Rebecca Briggs!
Here are some movie suggestions based on your rating
Movie ID: 4973
Movie ID: 111
Movie ID: 318
Movie ID: 296
Movie ID: 2064
Find similar movies:
Avatar
Get Similar Movies
Logout
How to use?
1. Sugested Movies: We have suggested few movies based on your rating
2. Search for a Movie: Start by typing the name of a movie you like in the search bar.
3. Get Recommendations: Click the 'Get Similar Movies' button.
4. Enjoy Your Movie Journey: Search for more and enjoy your movie journey.


Movie Recommender
localhost:8501
Deploy
Movie ID: 608
Movie ID: 50
Movie ID: 527
Movie ID: 1089
Movie ID: 1240
Find similar movies:
Avatar
Pirates of the Caribbean: At World's End
Spectre
The Dark Knight Rises
John Carter
Spider-Man 3
Tangled
Avengers: Age of Ultron
Harry Potter and the Half-Blood Prince
Made with Streamlit
How to use?
1. Sugested Movies: We have suggested few movies based on your rating
2. Search for a Movie: Start by typing the name of a movie you like in the search bar.
3. Get Recommendations: Click the 'Get Similar Movies' button.
4. Enjoy Your Movie Journey: Search for more and enjoy your movie journey.


Movie Recommender
localhost:8501
Deploy
How to use?
1. Sugested Movies: We have suggested few movies based on your rating
2. Search for a Movie: Start by typing the name of a movie you like in the search bar.
3. Get Recommendations: Click the 'Get Similar Movies' button.
4. Enjoy Your Movie Journey: Search for more and enjoy your movie journey.
Find similar movies:
The Dark Knight Rises
Get Similar Movies
Here are some similar movies:
BATMAN RETURNS
BATMAN FOREVER
The Dark Knight
Batman Returns
Batman
Batman Forever
Batman Begins
Logout
Built with by BrainLeftException

7. Project Folder Structure