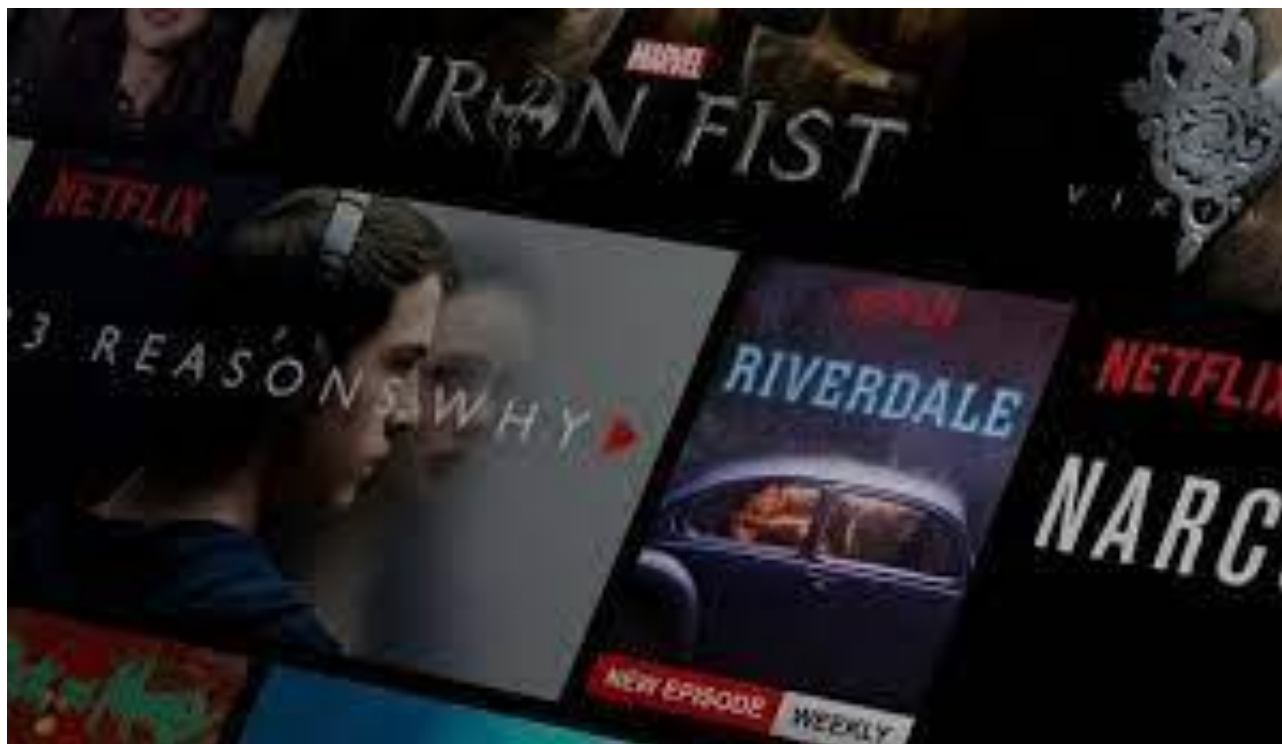


## ALGORITHMS FOR WEB AND INFORMATION RETRIEVAL

PROJECT TITLE :  
**MOVIE RECOMMENDATION  
 SYSTEM**



## TEAM DETAILS :

ID	NAME	SRN	SECTION
1	RISHAB KSHATRI	PES2UG19CS327	E
2	RAHUL S BHAT	PES2UG19CS315	E
3	RAHUL RADHAKRISHNA	PES2UG19CS314	E
4	SAI KALYAAN PALLA	PES2UG19CS354	F

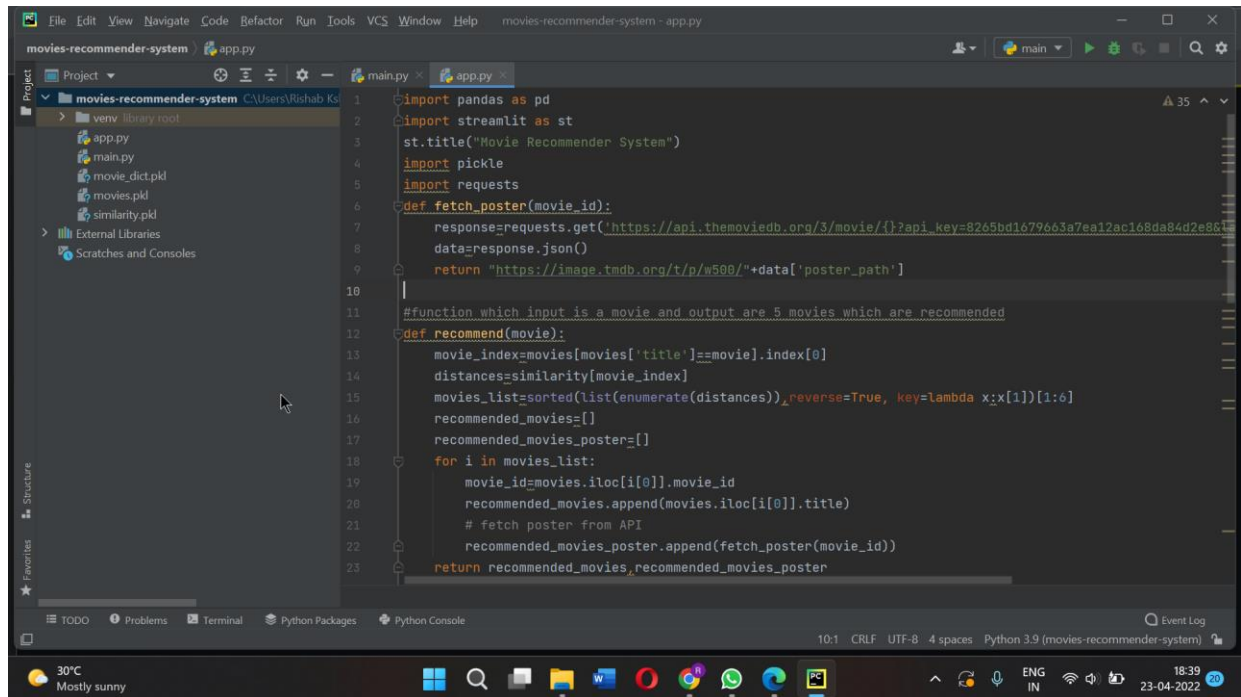
## Github Link :

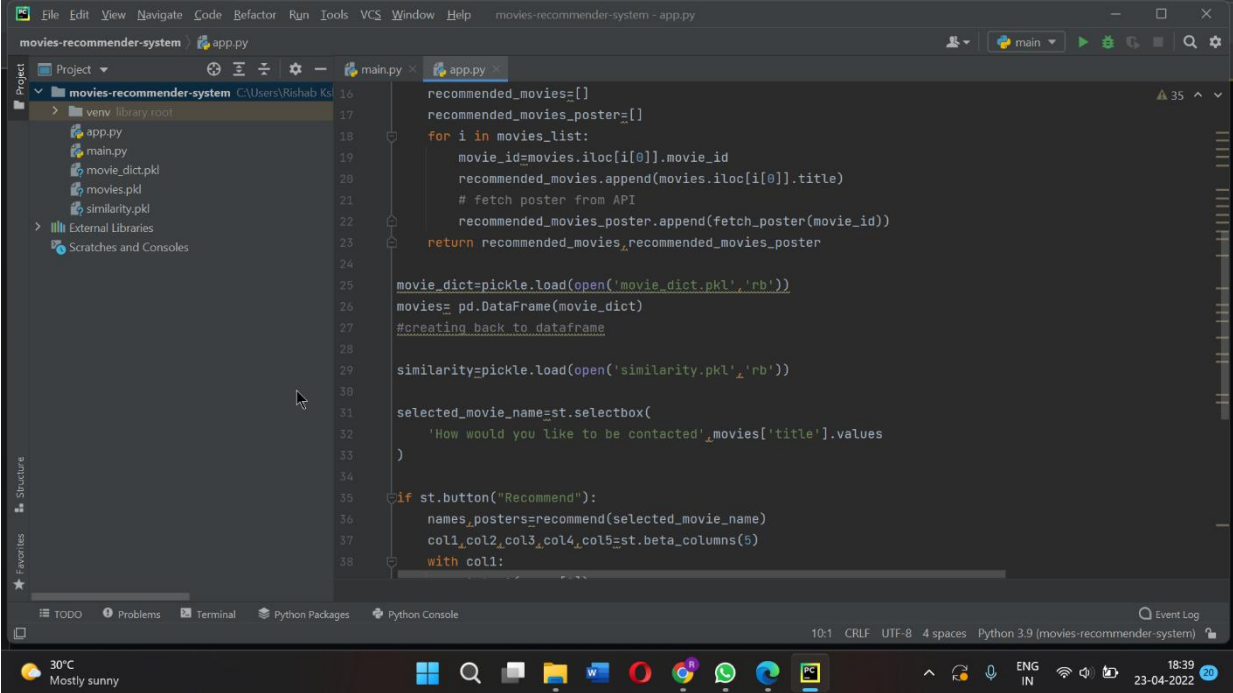
Refer the Github link for the .ipynb file.

<https://github.com/rishabkshatri/movie-recommender-system>

**CODE :**

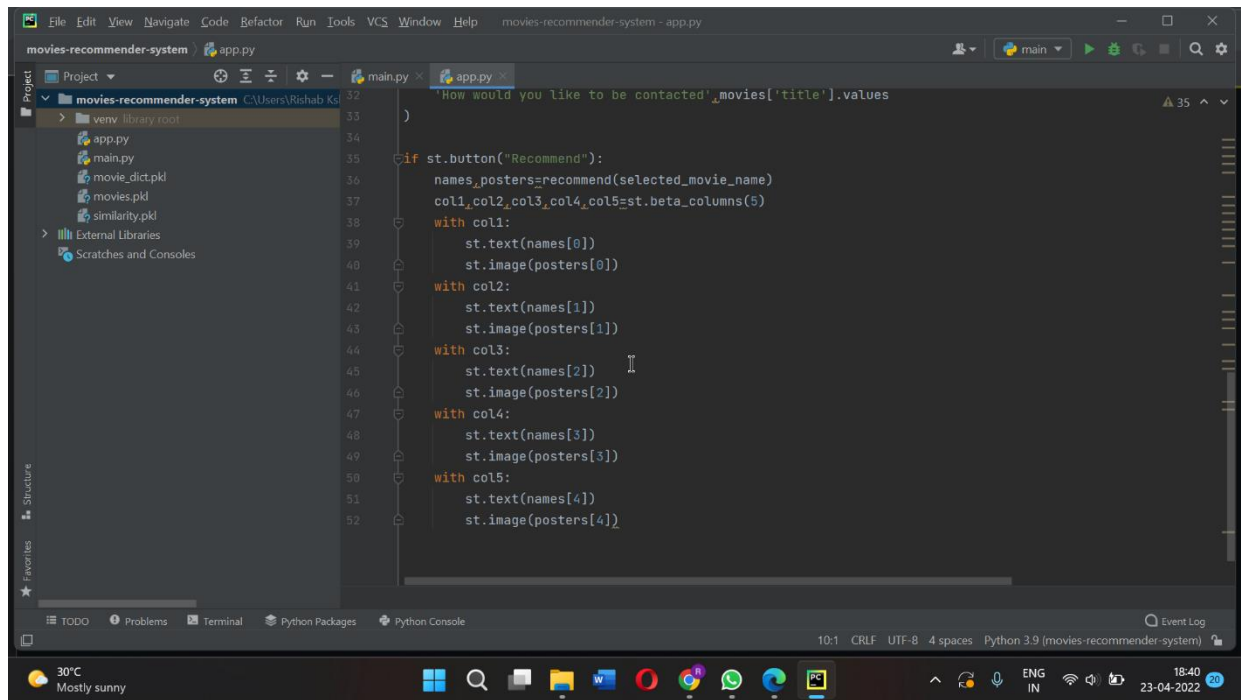
**APP.PY**





```
16 recommended_movies=[]
17 recommended_movies_poster=[]
18 for i in movies_list:
19     movie_id=movies.iloc[i[0]].movie_id
20     recommended_movies.append(movies.iloc[i[0]].title)
21     # fetch poster from API
22     recommended_movies_poster.append(fetch_poster(movie_id))
23     return recommended_movies,recommended_movies_poster
24
25 movie_dict=pickle.load(open('movie_dict.pkl','rb'))
26 movies= pd.DataFrame(movie_dict)
27 #creating back to dataframe
28
29 similarity=pickle.load(open('similarity.pkl','rb'))
30
31 selected_movie_name=st.selectbox(
32     'How would you like to be contacted',movies['title'].values
33 )
34
35 if st.button("Recommend"):
36     names_posters=recommend(selected_movie_name)
37     col1,col2,col3,col4,col5=st.beta_columns(5)
38     with col1:
```

The screenshot shows a PyCharm IDE window titled "movies-recommender-system - app.py". The left sidebar displays the project structure, including files like "app.py", "main.py", "movie\_dict.pkl", "movies.pkl", and "similarity.pkl". The main editor area shows the Python code for the movie recommender system. The code includes a loop to process a list of movies, a function to fetch posters, and a recommendation logic using a similarity matrix. The bottom status bar indicates the file encoding as UTF-8, 4 spaces indentation, and Python 3.9 interpreter.



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help movies-recommender-system - app.py
movies-recommender-system app.py
Project
  movies-recommender-system C:\Users\Rishab Ks
    venv library root
    app.py
    main.py
    movie_dict.pkl
    movies.pkl
    similarity.pkl
  External Libraries
  Scratches and Consoles
Structure
  app.py
  main.py
  movie_dict.pkl
  movies.pkl
  similarity.pkl
  Python Console
  Python Packages
  Terminal
  Problems
  TODO
  Event Log
  10:1 CRLF UTF-8 4 spaces Python 3.9 (movies-recommender-system)
  30°C Mostly sunny 18:40 23-04-2022
```

```
32 'How would you like to be contacted', movies['title'].values
33 )
34
35 if st.button("Recommend"):
36     names_posters=recommend(selected_movie_name)
37     col1,col2,col3,col4,col5=st.beta_columns(5)
38     with col1:
39         st.text(names[0])
40         st.image(posters[0])
41     with col2:
42         st.text(names[1])
43         st.image(posters[1])
44     with col3:
45         st.text(names[2])
46         st.image(posters[2])
47     with col4:
48         st.text(names[3])
49         st.image(posters[3])
50     with col5:
51         st.text(names[4])
52         st.image(posters[4])
```

To run the code :

```
streamlit run app.py [ generates embeddings.pkl ,
filenames.pkl ]streamlit run "path to main.py"
```

SNAPSHOTS OF THE FINAL OUTPUT OF .IPYNB FILE:

OUTPUT OF THE RECOMMENDATIONS OF DIFFERENT MOVIES:

The screenshot shows a Jupyter Notebook titled 'movie-recommender-system' running on localhost:8888. The notebook has several tabs open: 'machine-learning-project', 'movie-recommender-sys', 'movie-recommender-sys', 'Jupyter Notebook', and 'Python | Pandas dataframe'. The code in the notebook defines a 'recommend' function that takes a movie title as input and returns a list of recommended movies. The function uses a similarity matrix and a list of movie titles to find the top 5 most similar movies. The output of the function is displayed for two inputs: 'Avatar' and 'Batman Begins'.

```
In [113]: def recommend(movie):
          movie_index=new_df[new_df['title']==movie].index[0]
          distances=similarity[movie_index]
          movies_list=sorted(list(enumerate(distances)),reverse=True, key=lambda x:x[1])[1:6]
          for i in movies_list:
              print(new_df.iloc[i[0]].title)

In [114]: recommend('Avatar')
          Lifeorce
          Aliens vs Predator: Requiem
          Battle: Los Angeles
          Titan A.E.
          Independence Day

In [115]: recommend('Batman Begins')
          The Dark Knight
          Amidst the Devil's Wings
          The Dark Knight Rises
          Batman & Robin
          Mi America
```

Using the count Vectorizer Module:

The screenshot shows a Jupyter Notebook titled 'movie-recommender-system' running on localhost:8888. The notebook has several tabs open: 'machine-learning-project', 'movie-recommender-sys', 'movie-recommender-sys', 'Jupyter Notebook', and 'Python | Pandas dataframe'. The code in the notebook implements a Count Vectorizer module using the sklearn library. The code defines a function 'calculate\_similarity' that takes two tags as input and returns a similarity score. The function uses a Count Vectorizer to convert the tags into vectors and then calculates the cosine similarity between the two vectors. The output of the function is displayed for two inputs: 'Avatar' and 'Batman Begins'.

```
In [84]: #calculating similarity score between 2 tags-use vectorization w/o stop words
          #converting every tag into a corresponding vector
          #5000 vectors

In [85]: #ways to convert text into vector
          #tf-idf already used, we will use bag of words
          #concatenate all tags and extract top 5k words

In [86]: #create a table of movie-x and words-y for how many times each words is appearing in that tag
          #each row is a vector and fetch the closest 5 vector to recommend to user

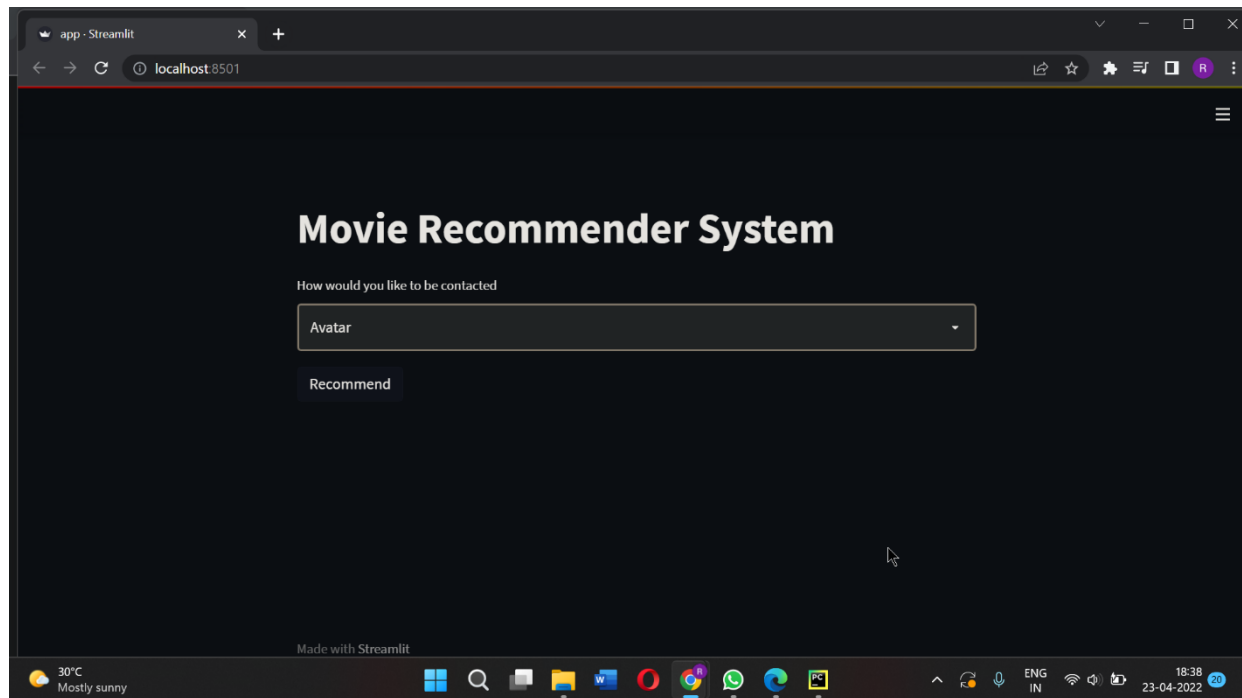
In [87]: from sklearn.feature_extraction.text import CountVectorizer
          cv = CountVectorizer(max_features=5000,stop_words='english')

In [92]: vectors = cv.fit_transform(new_df['tags']).toarray()

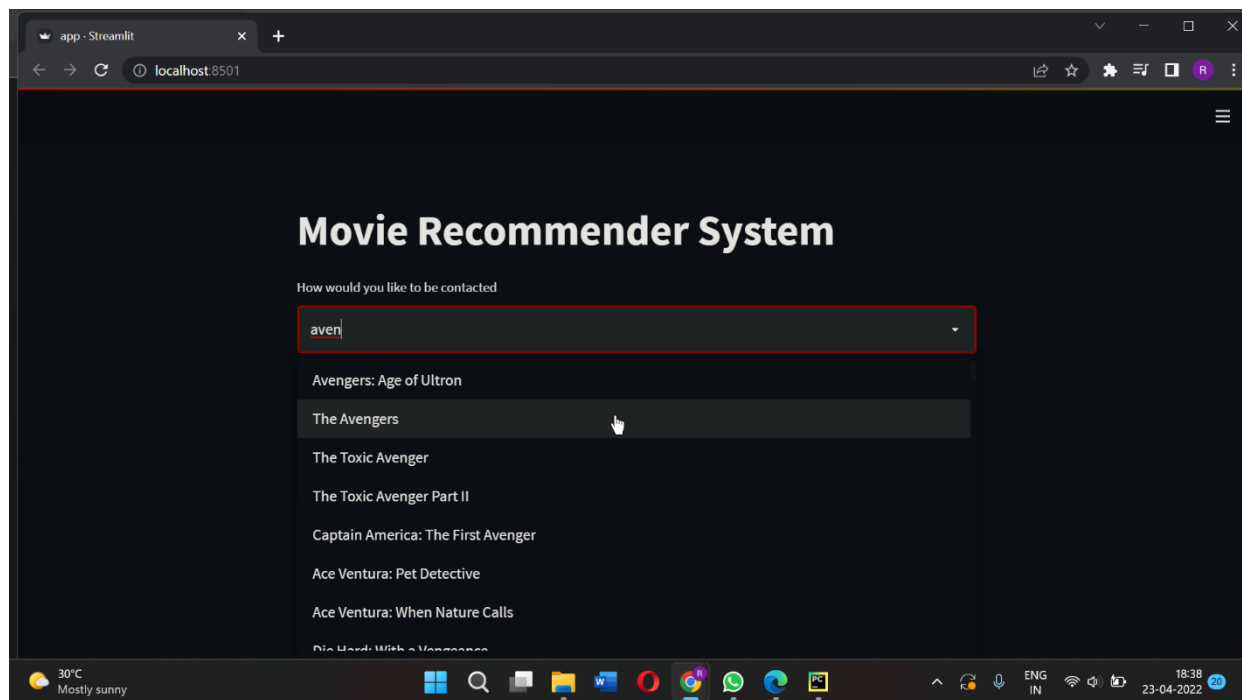
In [93]: vectors
          Out[93]: array([[0, 0, ..., 0, 0, 0],
          [0, 0, ..., 0, 0, 0],
          [0, 0, ..., 0, 0, 0],
          ...,
          [0, 0, ..., 0, 0, 0],
          [0, 0, ..., 0, 0, 0],
          [0, 0, ..., 0, 0, 0]], dtype=int64)
```

## OUTPUT1: (RECOMMENDATION WRT “THE AVENGERS”)

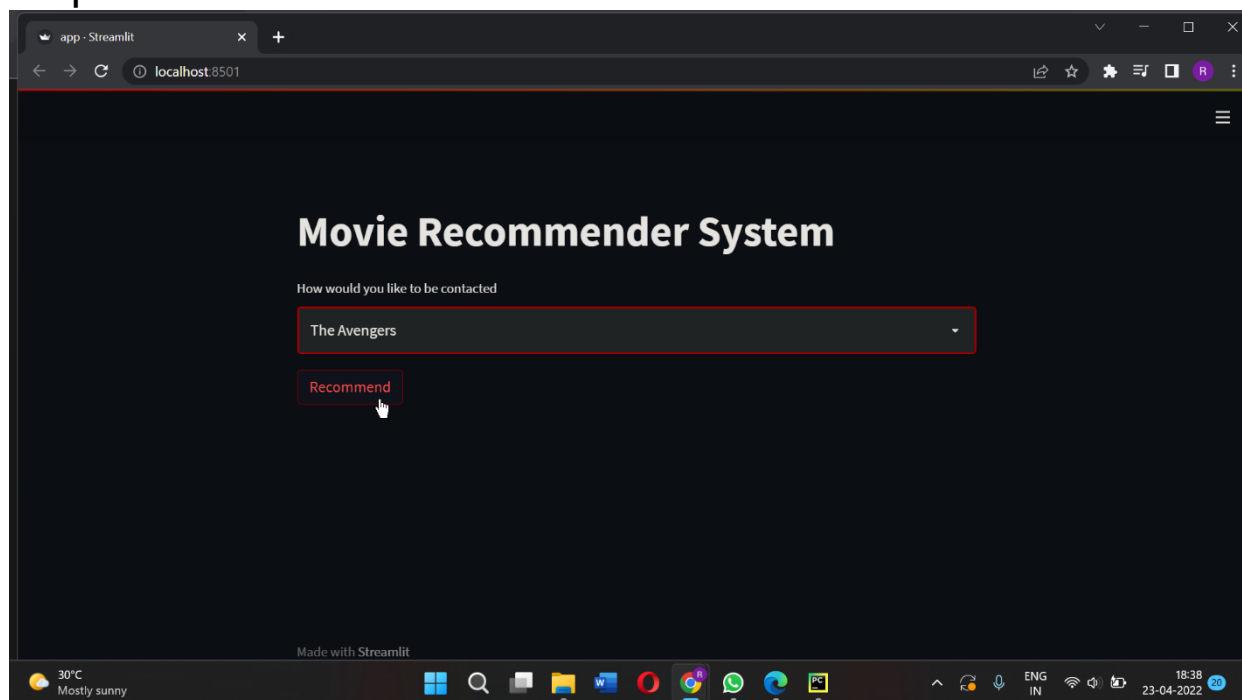
Step 1: Log on to the site.



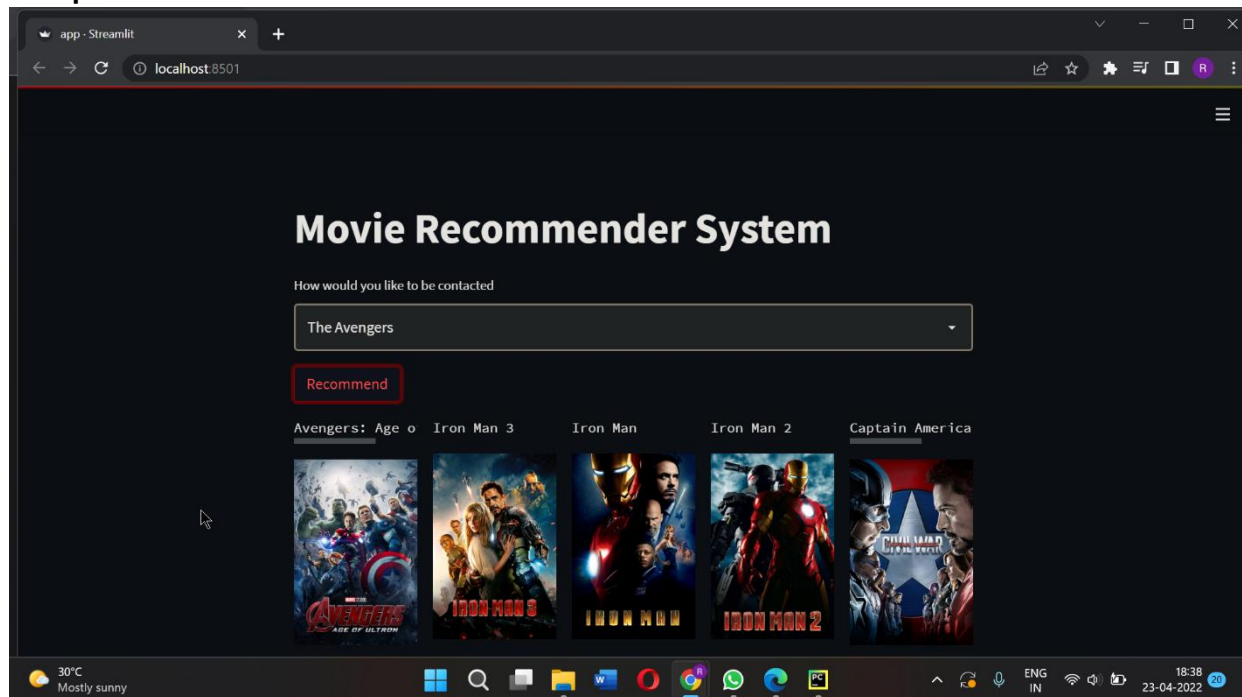
## Step 2: Select the suggested movie from the drop down



## Step 3: Click On Recommend.

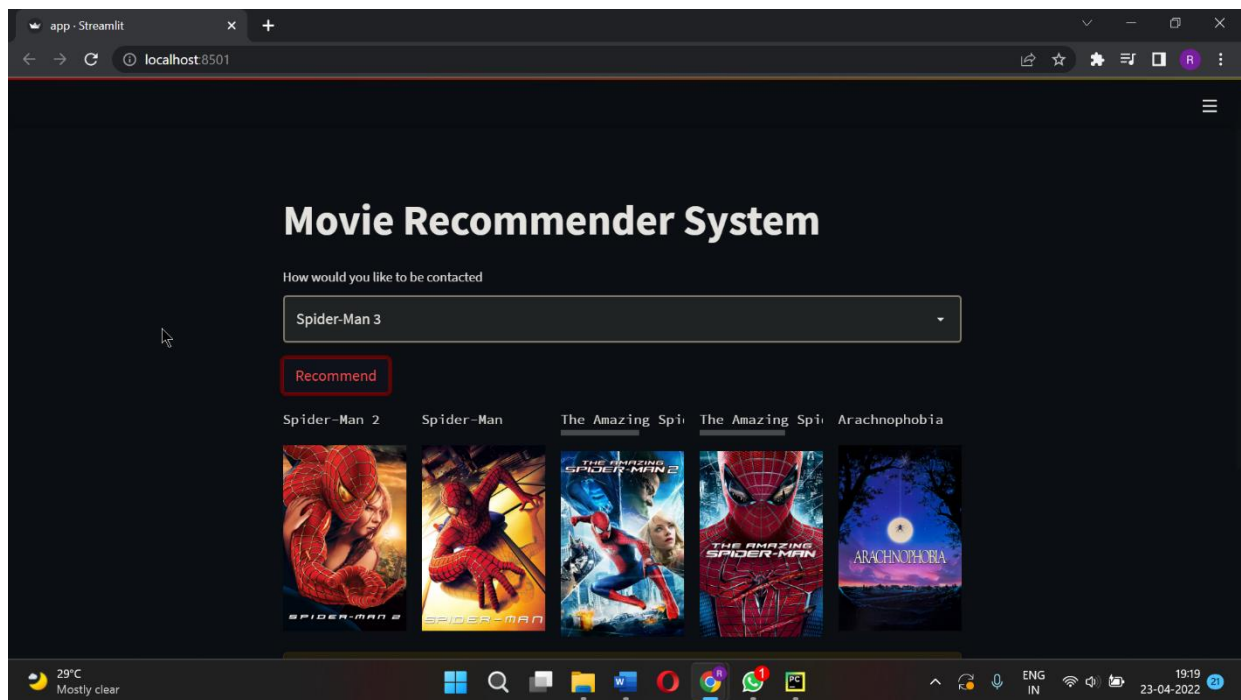


Step 4: See the results of the recommendation.

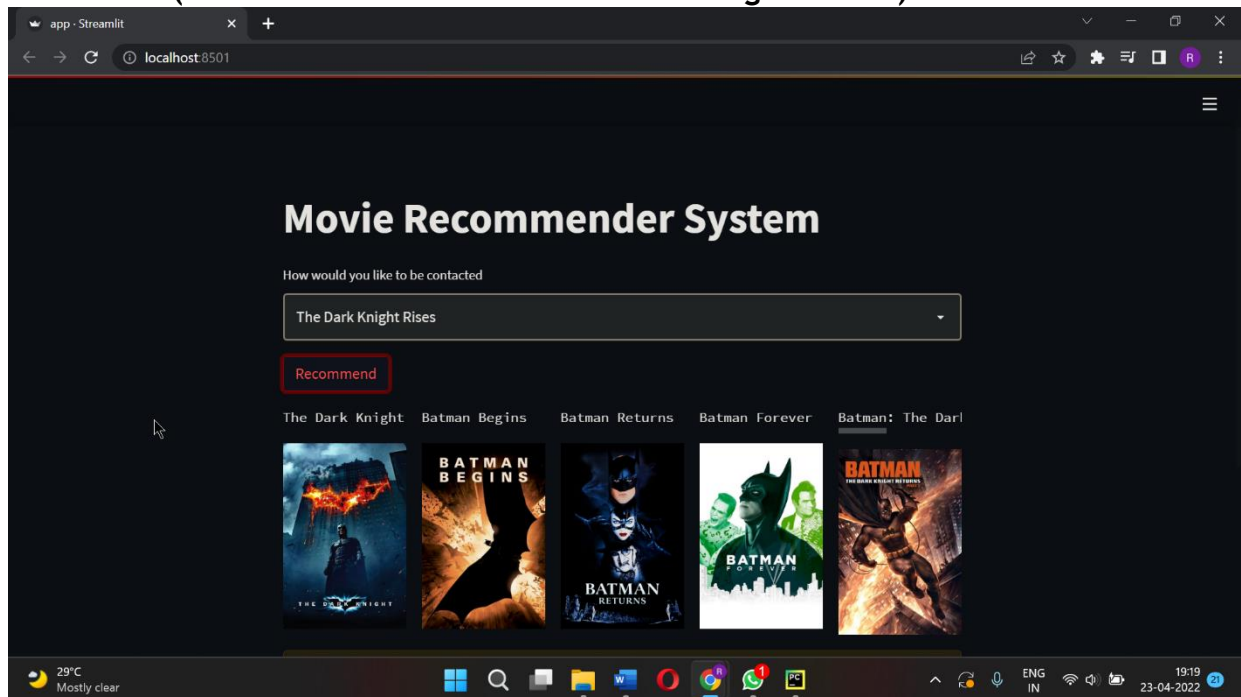




## OUTPUT2: (RECOMMENDATION WRT “THE SPIDERMAN”)



## OUTPUT2: (RECOMMENDATION WRT “The Dark Knight Rises”)



)