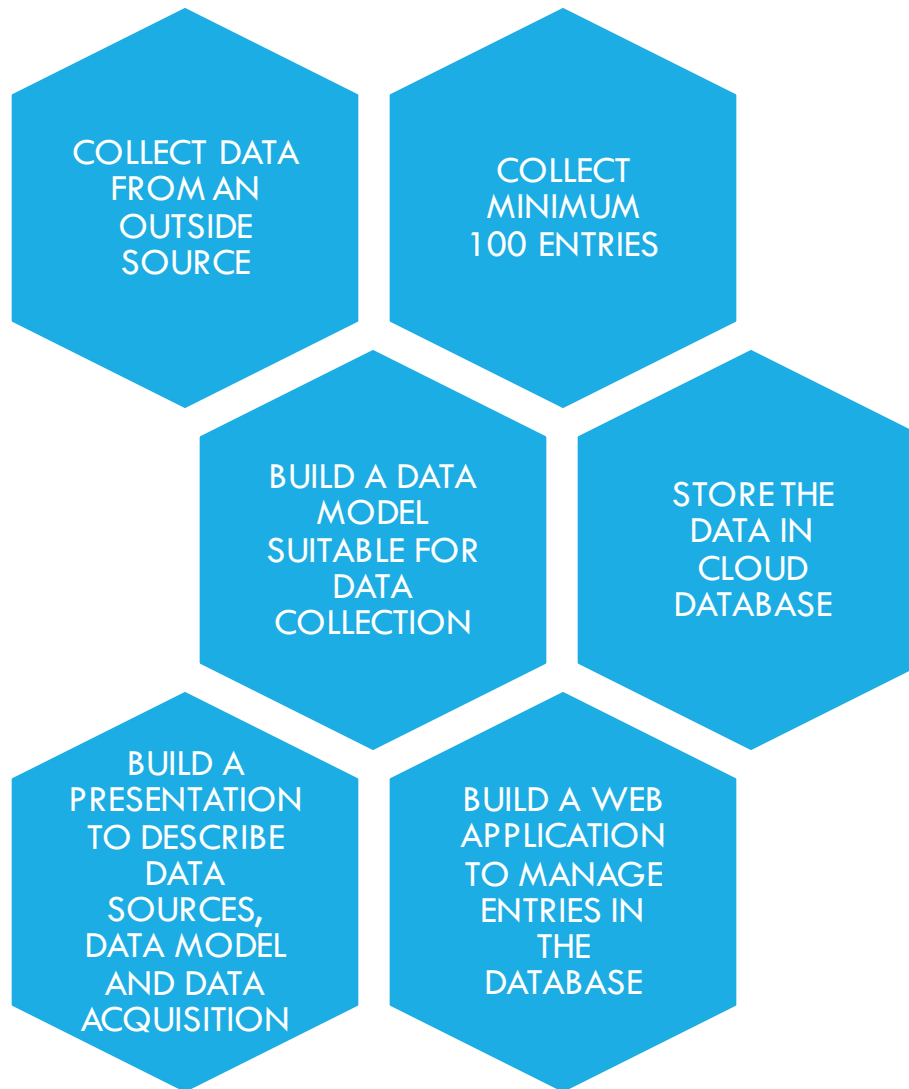


A complex network diagram with numerous nodes of varying sizes and colors (dark blue, light blue, grey) connected by thin grey lines. Some nodes are highlighted with larger concentric circles. The background is a light grey with faint circular patterns.

# SOCIAL DATA MINING TECHNIQUES

Assignment 1 – Social and Open Data Sources (Group 6)

## TOP 100 COMEDY MOVIES



# AGENDA

# TEAM PROFILE



**Dhruv Upadhyaya**

Data Analyst



**Dhruval  
Domadiya**

Business Analyst



**Jay Gajjar**

Python Developer



**Swapnil Shah**

Web Developer

# WORK-FLOW

1. Collect data from an open Source (Web Scraping)

Data Collection

Build Data Model

2. Build a data model for collected data

3. Store the data to cloud Database (MongoDB)

Store Data to Cloud Database

Build a CRUD Web Application

4. Build a web application to manage database entries

# ABOUT THE DATA

## What is Rotten Tomatoes?

- Rotten Tomatoes is an American review-aggregation website for film and television. The company was launched in August 1998 by three undergraduate students at the University of California, Berkeley: Senh Duong, Patrick Y. Lee, and Stephen Wang.
- Rotten Tomatoes is the world's most trusted recommendation resources for quality entertainment that helps fans to decide what to watch based on opinion of hundred of critics.

## Why use of this data source?

- Rotten tomatoes is the simplest and effective source of the data from where we can easily scrap the data to build a web application and manage CRUD operations.
- The data provided on the website is changing with the time for top entries depends upon critic reviews and ratings as new releases.



# WEB SCRAPING

- Web Scraping refers to the extraction of the data from a data source(website).
- The information from the HTML page can be exported into the format which is more useful for users to perform needed operations.
- The data from the website can be collected in the form of spreadsheets or an API.
- Although web scraping can be done manually, in most cases, automated tools are preferred when scraping web data as they can be less costly and work at a faster rate.
- In many cases, web scraping is not a simple task. Websites come in many shapes and forms, as a result, web scrapers vary in functionality and features

# TECHNOLOGIES



Python is an interpreted high-level general-purpose, object-oriented programming language. It has great number of libraries that provides high-level functionalities.



Flask is a micro web framework written in Python. It is an API of Python that allows us to build up web-applications. It is easier to learn because it has less base code to implement a simple web-Application.



MongoDB is based on a NoSQL database that is used for storing data in a key-value pair. Its working is based on the concept of document and collection. It is also an open-source, a document-oriented, cross-platform database system that is written using C++.

# DATA COLLECTION

- Web Scraping with the use of Python programming and BeautifulSoup library is performed.
- By importing required libraries, we can fetch the data from the website required for further operations.

```
WebCrawler_CRUD.py > ...
1  import pandas as pd
2  import numpy as np
3  import requests
4  import json
5  import csv
6  from pprint import pprint
7  from bs4 import BeautifulSoup as bs
8  from pymongo import MongoClient
9  from flask import Flask, jsonify, request, render_template, redirect
10 from requests.models import REDIRECT_STATI
11
12
```



# DATA MODEL

Field	Data Type	Description
Rank	Int	Rank of the movie (PK)
Rating	Decimal	Overall ratings
Title	Varchar, String	Name of the movie
Reviews	Int	No. of reviews

**Rotten  
Tomatoes**

# BEAUTIFULSOUP

- BeautifulSoup is the library in Python to pull the data from the HTML and XML files.
- It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.
- BeautifulSoup act as a fast Parser.
- It helps to fetch the content from the website by removing markups in HTML formats and save this information that can be used in form of user choice.
- It is considered as the best option when user want to pull out some of the data fields having multiple range of data.

```

#CODE TO SCRAPE THE DATA
if r.status_code == 200:
    rt = bs(r.text,"html.parser")
    tab = rt.find('table', attrs={'class':'table'})
    # print(tab.prettify())

    #get columns
    for data in tab.find_all('tr'):
        row_data = []

        #rank of the movie
        if data.find('td', attrs = {'class' : 'bold'}) is not None:
            (variable) row_data: list = {'class' : 'bold'})
            row_data.append(rnk.text.strip())
            print("'" + rnk.text + "'")

        #Ratings of the movie
        if data.find('span', attrs = {'class' : 'tMeterScore'}) is not None:
            rating = data.find('span', attrs = {'class' : 'tMeterScore'})
            row_data.append(rating.text.strip())
            print("'" + rating.text.strip() + "'")

        #Name of the movie
        if data.find('a', attrs = {'class' : 'unstyled articleLink'}) is not None:
            title = data.find('a', attrs = {'class' : 'unstyled articleLink'})
            row_data.append(title.text.strip())
            print("'" + title.text.strip() + "'")

        #Number of reviews
        if data.find('td', attrs = {'class' : 'right hidden-xs'}) is not None:
            review = data.find('td', attrs = {'class' : 'right hidden-xs'})
            row_data.append(review.text.strip())
            print("'" + review.text + "'")

        table_content_list.append(row_data)

# Saving in CSV file using pandas dataframe
df = pd.DataFrame(table_content_list, columns=['Rank','Rating','Title','Reviews'])
df.to_csv("D:\Project\Python\SDM_ASG\Top_100_Comedy_Movie_list.csv")

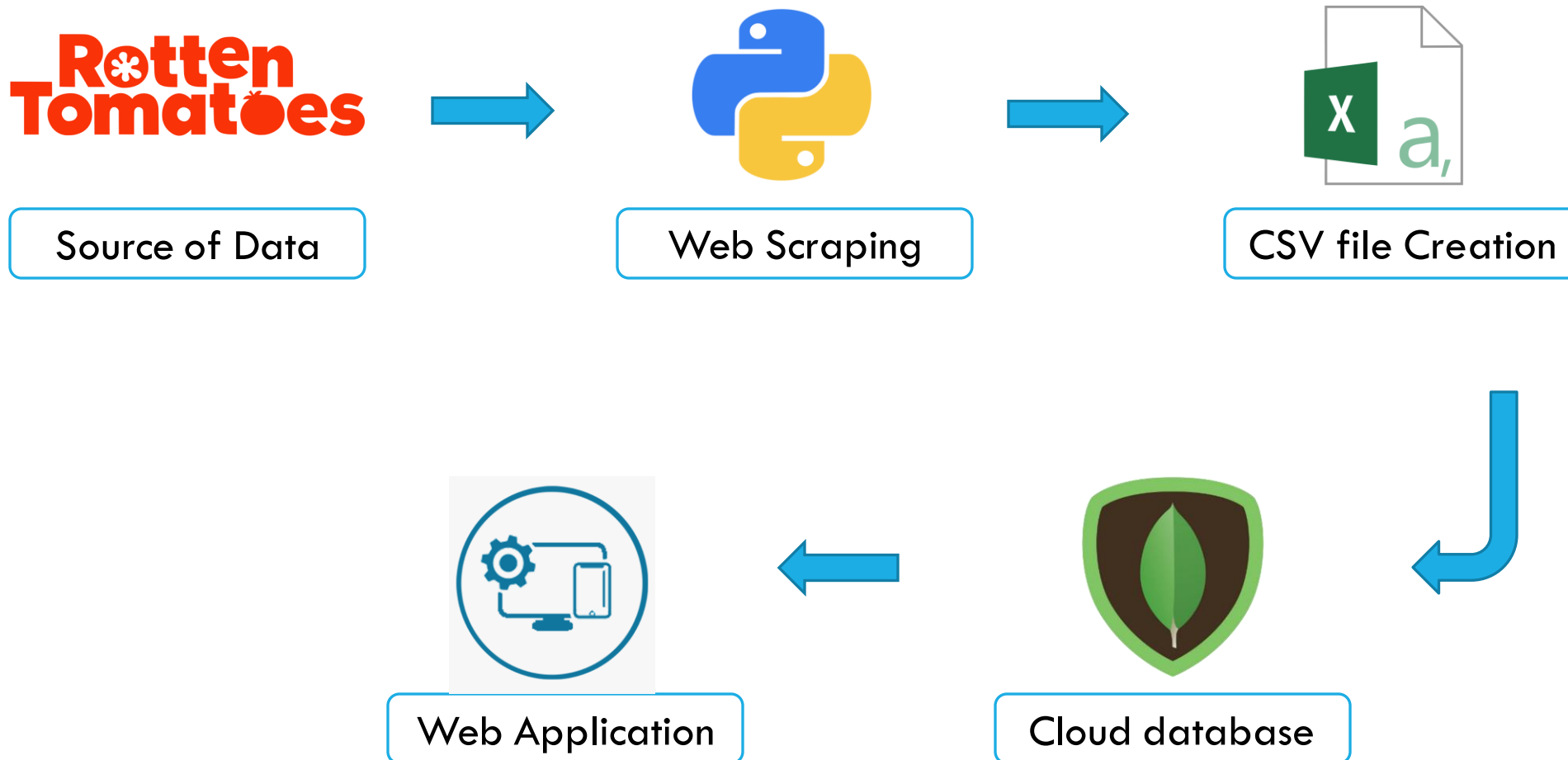
```

	Rank	Rating	Title	Reviews
1	1	99%	It Happened One Night (1934)	97
2	2	98%	Modern Times (1936)	108
3	3	97%	Toy Story 4 (2019)	452
4	4	99%	Lady Bird (2017)	398
5	5	96%	BlackKkKlansman (2018)	447
6	6	100%	The Philadelphia Story (1940)	101
7	7	99%	Eighth Grade (2018)	318
8	8	96%	Booksmart (2019)	376
9	9	97%	Coco (2017)	352
10	10	97%	The Farewell (2019)	343
11	11	97%	A Night at the Opera (1935)	69
12	12	100%	Singin' in the Rain (1952)	67
13	13	98%	The Big Sick (2017)	303
14	14	100%	The Kid (1921)	48
15	15	85%	Once Upon a Time In Hollywood (2019)	568
16	16	91%	La La Land (2016)	464
17	17	98%	Zootopia (2016)	297
18	18	99%	Paddington 2 (2018)	246
19	19	98%	A Hard Day's Night (1964)	110
20	20	100%	Top Hat (1935)	42
21	21	98%	Up (2009)	298
22	22	99%	His Girl Friday (1940)	67
23	23	98%	Toy Story 3 (2010)	309
24	24	100%	Toy Story 2 (1999)	169
25	25	90%	Three Billboards Outside Ebbing, Missouri (2017)	409

**Code used for fetching data from website to the CSV file**

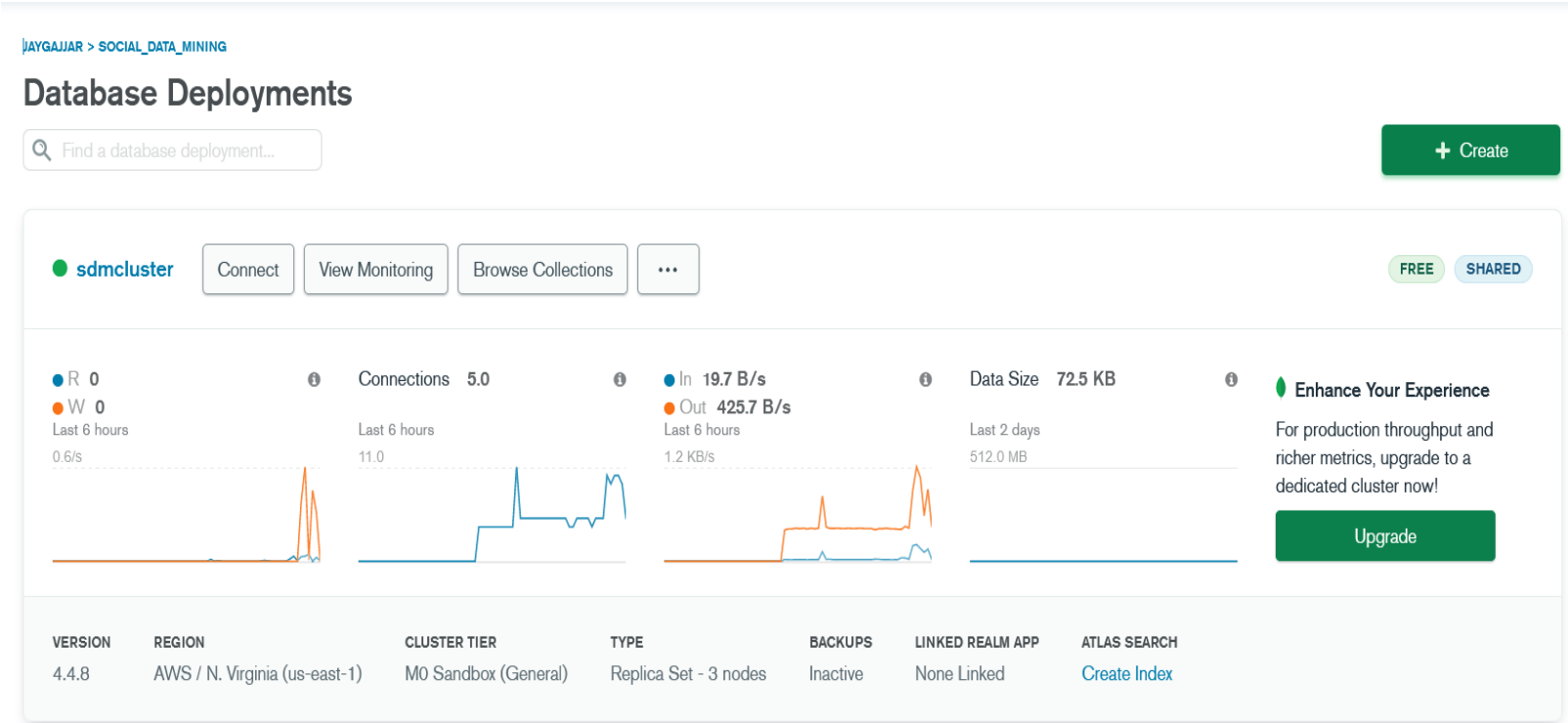
**CSV file created with extracted Data**

# FLOW OF APPLICATION



```
65
66 #connection string of mongoDB
67 client = MongoClient('mongodb+srv://jay:haha1234@sdmcluster.6rfbh.mongodb.net/exchange?ssl=true&ssl_cert_reqs=CERT_NONE')
68 db = client.get_database('rottentomatoes')
69 records = db.movies
70
71 FilePath = r'C:\Users\JAY\Desktop\Georgian - BDAT\Semester 2\1007 - Social Data Mining Techniques\Web Scrapping'
72
```

Connect to the MongoDB



Overview

Real Time

Metrics

Collections

Search

Profiler

Performance Advisor

Online Archive

Command Line Tools

DATABASES: 1 COLLECTIONS: 2 |

VISUALIZE YOUR DATA

REFRESH

+ Create Database

Q NAMESPACES

▼ rottentomatoes

| movies

reviews

## rottentomatoes.movies

COLLECTION SIZE: 10.19KB TOTAL DOCUMENTS: 100 INDEXES TOTAL SIZE: 20KB

Find

Indexes

Schema Anti-Patterns 0

Aggregation

Search Indexes ●

INSERT DOCUMENT

FILTER { field: 'value' }

► OPTIONS

Apply

Reset

QUERY RESULTS 1-20 OF MANY

```
_id: ObjectId("614a03da035d3d99e7a13856")
Rank: 1
Rating: "99%"
Title: "It Happened One Night (1934)"
Reviews: 97
```

**Collection of data in a database**

#### QUERY RESULTS 1-20 OF MANY

`_id: ObjectId("614a03da035d3d99e7a13856")`  
`Rank: 1`  
`Rating: "99%"`  
`Title: "It Happened One Night (1934)"`  
`Reviews: 97`

`_id: ObjectId("614a03da035d3d99e7a13857")`  
`Rank: 2`  
`Rating: "98%"`  
`Title: "Modern Times (1936)"`  
`Reviews: 108`

`_id: ObjectId("614a03da035d3d99e7a13858")`  
`Rank: 3`  
`Rating: "97%"`  
`Title: "Toy Story 4 (2019)"`  
`Reviews: 452`

`_id: ObjectId("614a03da035d3d99e7a13859")`  
`Rank: 4`  
`Rating: "99%"`  
`Title: "Lady Bird (2017)"`  
`Reviews: 398`



**Data entries in a MongoDB**

# VIEW OF THE DATA

90	91.0	94%	The Edge of Seventeen (2016)	216.0
91	92.0	97%	Bull Durham (1988)	71.0
92	93.0	96%	The Rules of the Game (La règle du jeu) (1939)	50.0
93	94.0	93%	The Apartment (1960)	72.0
94	95.0	94%	The Women (1939)	63.0
95	96.0	92%	Silver Linings Playbook (2012)	260.0
96	97.0	95%	Lost In Translation (2003)	232.0
97	98.0	98%	Broadcast News (1987)	52.0
98	99.0	96%	Tangerine (2015)	160.0

[Get the full list of movies](#)[Create a new entry](#)[Update the existing record](#)[Delete the record](#)

	Rank	Rating	Title	Reviews
0	1.0	99%	It Happened One Night (1934)	97.0
1	2.0	98%	Modern Times (1936)	108.0
2	3.0	97%	Toy Story 4 (2019)	452.0
3	4.0	99%	Lady Bird (2017)	398.0
4	5.0	96%	BlacKkKlansman (2018)	447.0
5	6.0	100%	The Philadelphia Story (1940)	101.0
6	7.0	99%	Eighth Grade (2018)	318.0
7	8.0	96%	Booksmart (2019)	376.0
8	9.0	97%	Coco (2017)	352.0
9	10.0	100%	Singin' in the Rain (1952)	67.0
10	11.0	97%	The Farewell (2019)	343.0
11	12.0	97%	A Night at the Opera (1935)	69.0
12	13.0	98%	The Big Sick (2017)	303.0
13	14.0	100%	The Kid (1921)	48.0
14	15.0	85%	Once Upon a Time In Hollywood (2019)	568.0
15	16.0	91%	La La Land (2016)	464.0
16	17.0	98%	Zootopia (2016)	297.0
17	18.0	99%	Paddington 2 (2018)	246.0
18	19.0	98%	A Hard Day's Night (1964)	110.0
19	20.0	100%	Top Hat (1935)	42.0
20	21.0	98%	Up (2009)	298.0

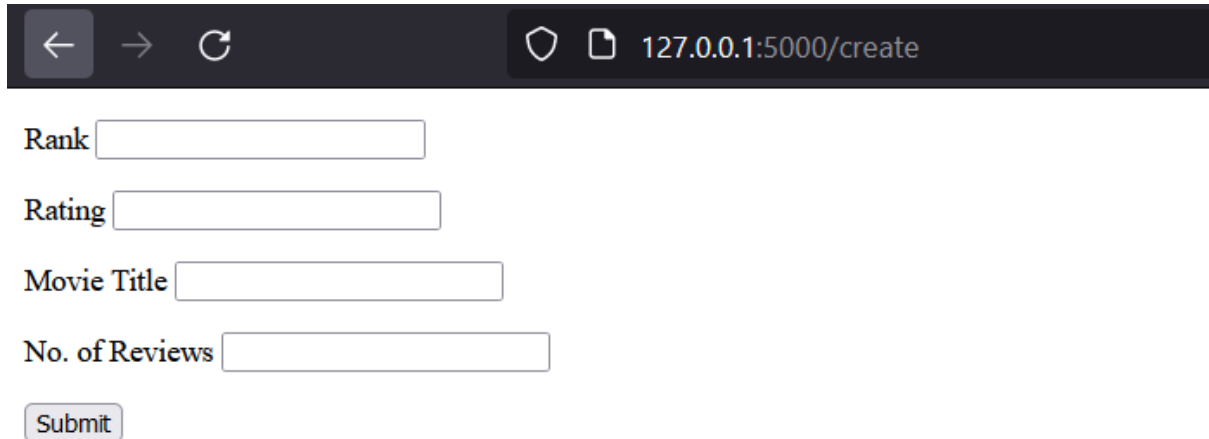
**Web Application Home page view**



# PYTHON CODE

```
@app.route('/', methods = ['GET','POST'])
def RetrieveDataList():
    if request.method == 'POST':
        if request.form.get('get') == 'Get the full list of movies':
            return render_template('home.html')
        elif request.form.get('create') == 'Create a new entry':
            return render_template('CreateView.html')
        elif request.form.get('update') == 'Update the existing record':
            return render_template('UpdateView.html')
        elif request.form.get('delete') == 'Delete the record':
            return render_template('DeleteView.html')
        else:
            pass
    elif request.method == 'GET':
        dataset = mongoDocExport()
        dataset.columns = ['Rank', 'Rating', 'Title', 'Reviews']
        header = 'Top movies of Rotten tomatoes'
        return render_template('home.html', tables=[dataset.to_html(classes='data')], titles = dataset.columns )
```

# ADD RECORD



Rank

Rating

Movie Title

No. of Reviews

```
@app.route('/create' , methods = ['GET','POST'])
def create():
    if request.method == 'GET':
        return render_template('CreateView.html')

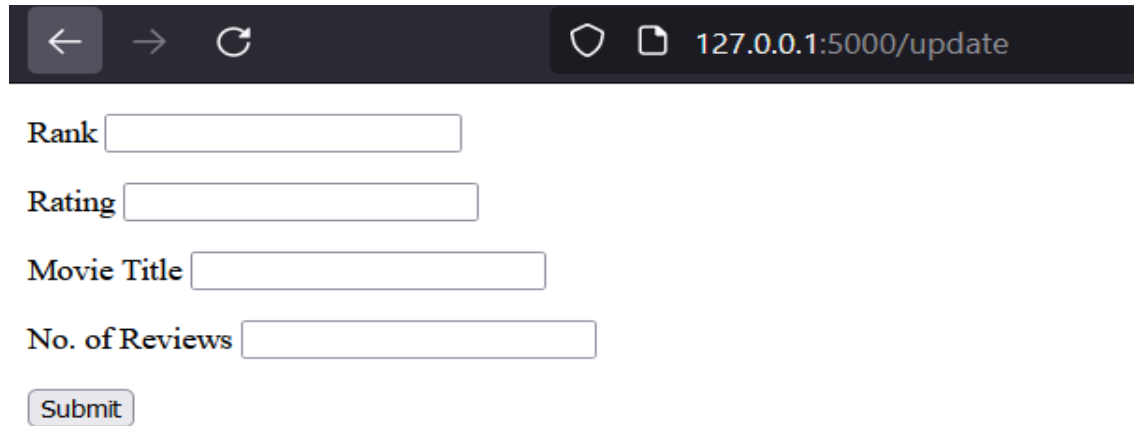
    if request.method == 'POST':
        rank = request.form['rank']
        rating = request.form['rating']
        title = request.form['title']
        reviews = request.form['reviews']

        mongoInsert(rank, rating, title, reviews)

        return redirect('/')
```

**Add a new record to the list/database**

# UPDATE RECORD



Rank

Rating

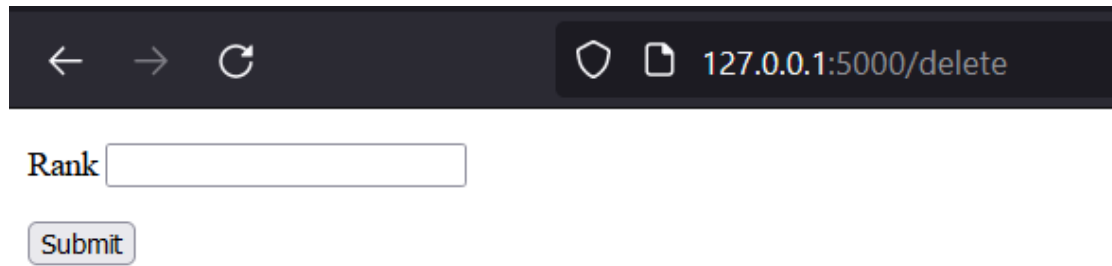
Movie Title

No. of Reviews

```
@app.route('/update', methods = ['GET', 'POST'])  
def update():  
    if request.method == 'GET':  
        return render_template('UpdateView.html')  
  
    if request.method == 'POST':  
        rank = request.form['rank']  
        rating = request.form['rating']  
        title = request.form['title']  
        reviews = request.form['reviews']  
  
        mongoUpdate(rank, rating, title, reviews)  
        return redirect('/')
```

**Update the existing record**

# DELETE RECORD



A screenshot of a web browser interface. The address bar shows the URL `127.0.0.1:5000/delete`. Below the address bar, there is a form with a label "Rank" followed by a text input field. Below the input field is a "Submit" button.

```
@app.route('/delete', methods=['GET', 'POST'])
def delete():
    if request.method == 'GET':
        return render_template('DeleteView.html')

    if request.method == 'POST':
        rank = request.form['rank']
        mongoDelete(rank)
        return redirect('/')
```

**Delete the entry from the list**

# LINK TO THE VIDEO PRESENTATION

<https://www.youtube.com/watch?v=6gx0ZD4N1aM>

[https://github.com/GajjarJay/WebScrapping\\_and\\_CRUD\\_Application.git](https://github.com/GajjarJay/WebScrapping_and_CRUD_Application.git)

# REFERENCES

- <https://kb.objectrocket.com/mongo-db/how-to-import-and-export-mongodb-data-using-pandas-in-python-355>
- <https://predictivehacks.com/?all-tips=how-to-add-action-buttons-in-flask>



Thank You