**1**  **INTRODUCTION**

* 1. **Overview**

Emotion-based Recommender System (E-MRS) that can capture customer preferences according to their emotions. Emotion plays an important role in rational and intelligent behavior, thus, we incorporate user emotions into the recommendation process. One of the underlying targets of movies is to evoke emotions in their viewers. IMDb offers all the movies for all genres. Therefore the movie titles can be scraped from the IMDb list to recommend to the user. IMDb does not have an API, for accessing the information on movies and TV Series. Therefore we have to perform scraping. Scraping is used for accessing information from a website which is usually done with APIs. The scraper is written in Python and uses lxml for parsing the web pages. BeautifulSoup is used for pulling data out of HTML and XML files. Anticipation’, ‘Disgust’, ‘Think’, ‘Happy’, ‘Sad’, ‘Anger’. Here these are taken as input and the corresponding movies would be displayed for the emotion. The correspondence of every emotion with the genre of movies is listed below: Based on the input emotion, the corresponding genre would be selected and all the top movies of that genre would be recommended to the user.

* 1. **Purpose**
* The main purpose of the software is to suggest a list of movies to the viewer, from the large amount of content available online, based on his/her emotion.
* Another purpose is to suggest list of latest and trending movies.

1. **LITERATURE SURVEY**
   1. **Existing Problem**

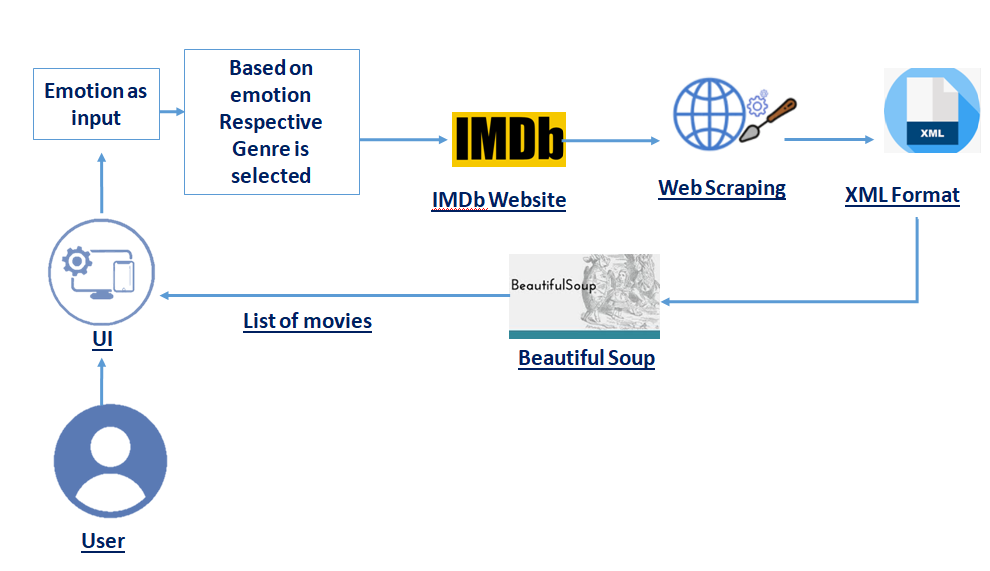
* Large number of unsorted movies is available online.
* Viewer is unable to find the movies that he/she wants to watch.
* Movies are not available on the basis of mood/emotion of the user.
* Existing software doesn’t suggest movies to improve viewer’s mood.

**2.2 Proposed Solution**

* We are making software that is based on the present mood of the user.
* This software will give movie suggestion to the viewer according to their present mood.
* This software will suggest them latest and trending movies on their mood from a large variety of movies online by “Web Scraping”.

1. **THEORITICAL ANALYSIS**

**3.1 Block Diagram**



**3.2 Hardware and Software Designing**

Hardware Requirements:

* Processor : Intel Core i3
* Processor Speed : 1.7 GHz
* RAM : 4 GB
* System Type : 64-Bit Operating System
* Monitor Resolution : 1280\*800

Software Requirements:

* Software : Spyder ( Anaconda3 )
* Operating System : Windows 10
* Front End : HTML, CSS
* Programming Language : Python

4 EXPERIMENTAL INVESTIGATIONS

**Web Scraping using Python:**

In today’s competitive world everybody is looking for ways to innovate and make use of new technologies. Web scraping (also called web data extraction or [data scraping](https://www.zyte.com/data-extraction/)) provides a solution for those who want to get access to structured web data in an automated fashion. Web scraping is useful if the public website you want to get data from doesn’t have an API, or it does but provides only limited access to the data.

Web Scraping refers to access the HTML of the webpage and extract useful information and data from it. This method is also called web scratching or web reaping or web information extraction. Here in this task, we are going to utilize web scratching to separate information from the website page utilizing Python and BeautifulSoup**.** The scraper is written in Python and utilizations lxml for parsing the site pages.

Web scraping is used in a variety of digital businesses that rely on data harvesting. Legitimate use cases include: Search engine bots crawling a site, Movie recommendation, analyzing its content and then ranking it. Market research companies use scrapers to pull data from forums and social media (e.g., for sentiment analysis).

**Beautiful soup:**

It is a library of python which is utilized to pull the data from the web pages i.e HTML and XML files. It works with your preferred parser to give colloquial methods for exploring, looking and changing the parse tree. I choose Beautiful soup because it’s a very simple library with minimal lines of code required to get a lot done. Beautiful Soup or Soup is a library which can parse XML, JSON and HTML, HTML5 content step by step.

**Classify the emotions based and associated with the Genre of Movie**

There are 8 classes of feeling that would be compelling to classification to the text. These are: *‘Anger’, ‘Expectation’, ‘Disgust’, ‘Fear’, ‘Bliss’, ‘Sad’, ‘Surprise’, ‘Trust’*. Here these are taken as information and the relating motion pictures would be shown for the feeling.

The correspondence of each feeling with the class of films is recorded beneath:

* Sad – Drama
* Disgust – Musical
* Anger – Family
* Expectation – Thriller
* Fear – Sport
* Enjoyment – Thriller
* Trust – Western
* Surprise – Film-Noir

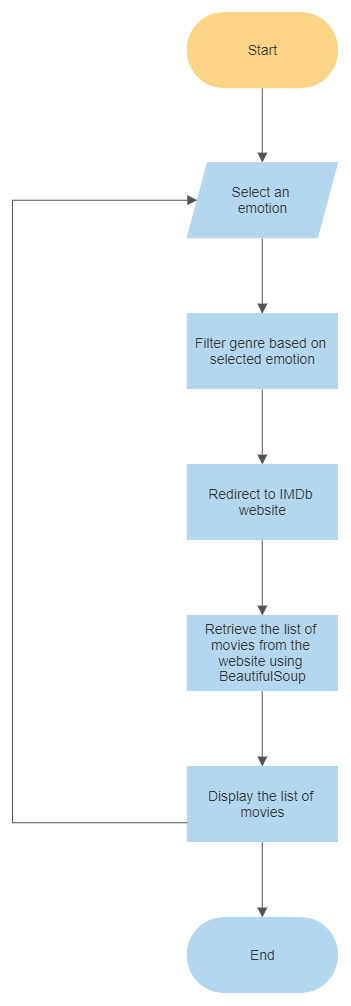
**Natural language processing (NLP):**

 Sentiment relates to the meaning of a word or sequence of words and is usually associated with an opinion or emotion. Well, analysis is the process of looking at data and making inferences; in this case, using machine learning to learn and predict whether a movie review is positive or negative. Because the movie review dataset consist of only the review text, we need to use text and natural language feature to build a meaningful dataset for our sentimental model. Maybe you’re interested in knowing whether movie reviews are positive or negative, companies use sentiment analysis in a variety of settings, particularly for marketing purposes. Uses include social media monitoring, brand monitoring, customer feedback, customer service and market research (“Sentiment Analysis”).

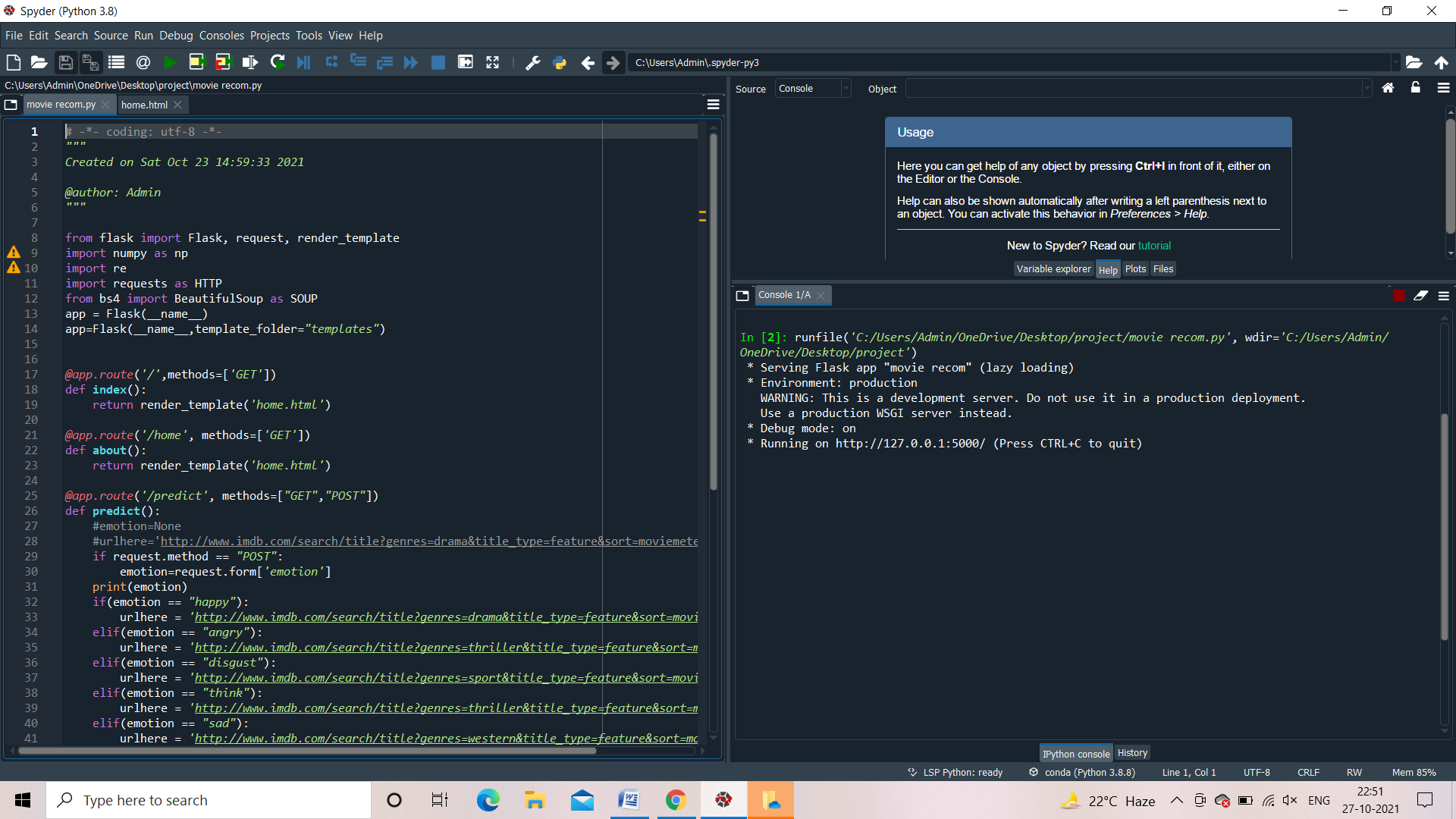
**Python flask:**

Flask is a micro web framework written in Python. Extensions exists for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Flask is used for the backend, but it makes use of a templating language called Jinja2 which is used to create HTML, XML or other markup formats that are returned to the user via an HTTP request.  Flask offers a diversified working style while Django offers a Monolithic working style. It is designed as a web framework for restful API development

**5 FLOWCHART**

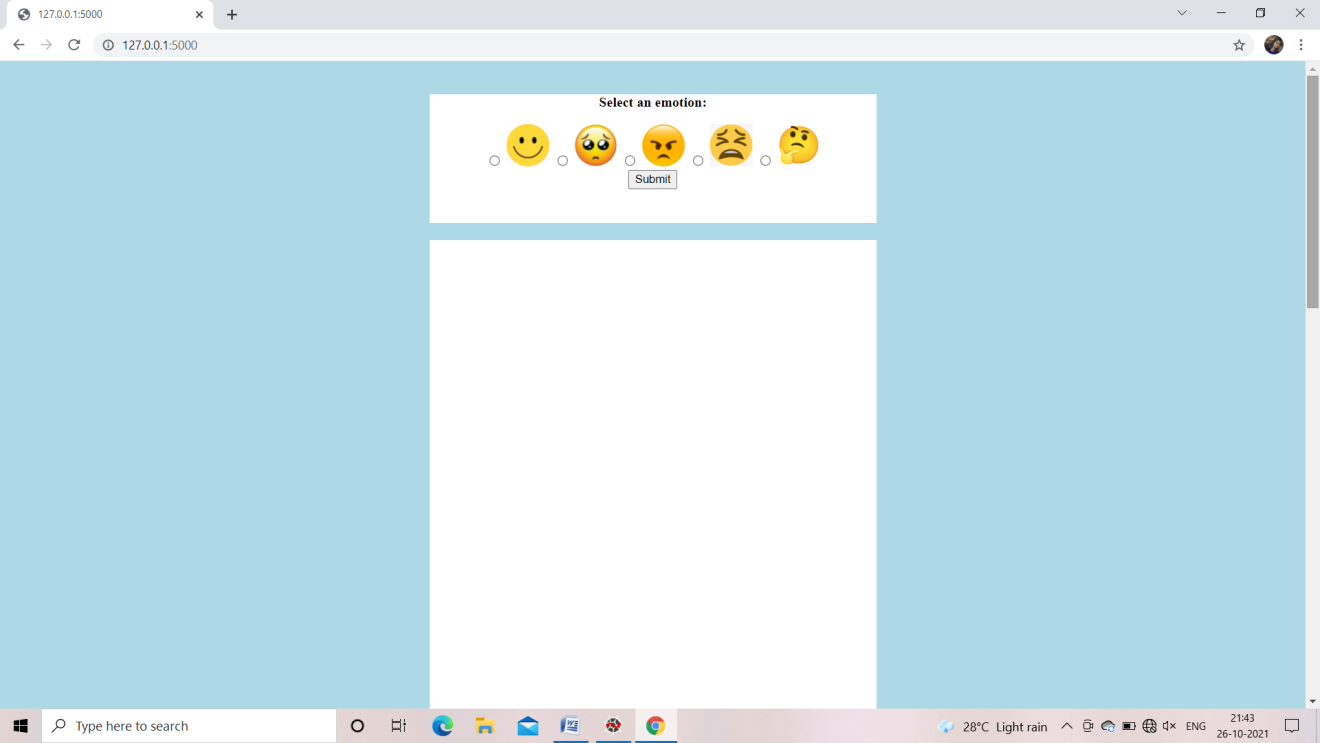
****

**6 RESULT**

****

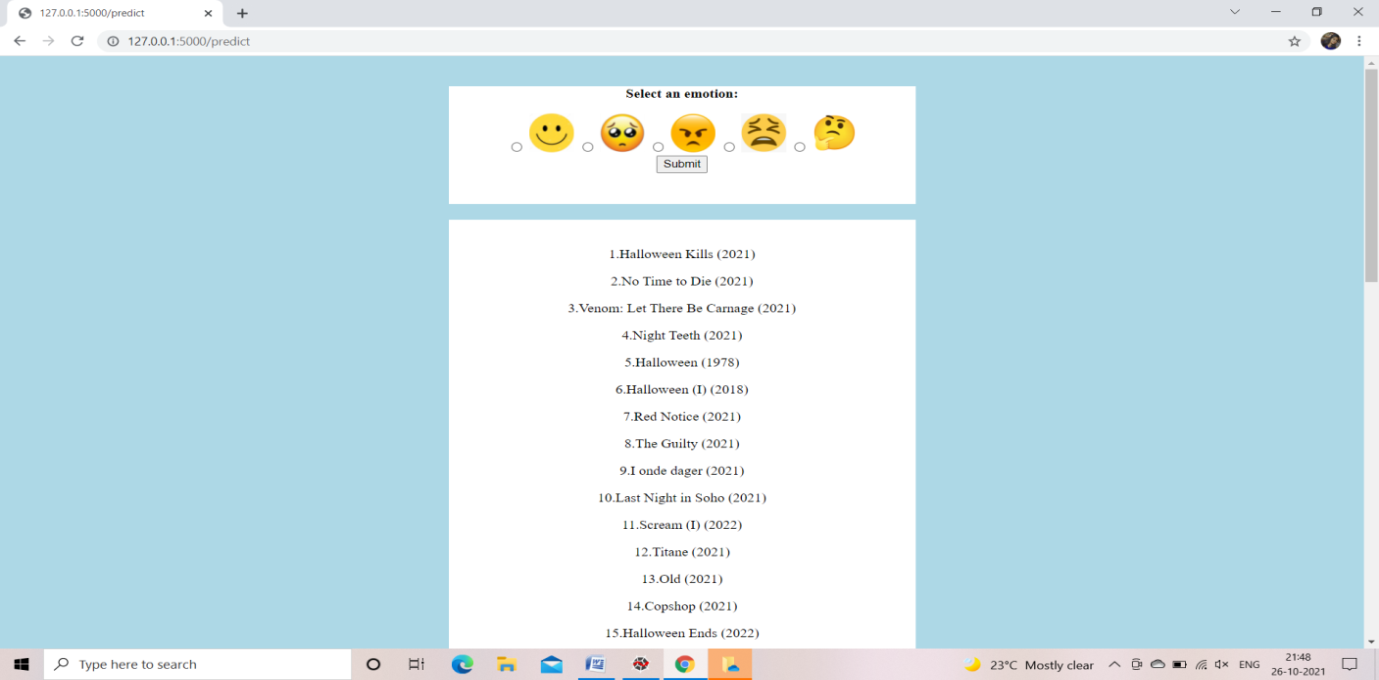
**Fig 6.1: Output Page**

The Fig 6.1 displays Output of the Python Code

****

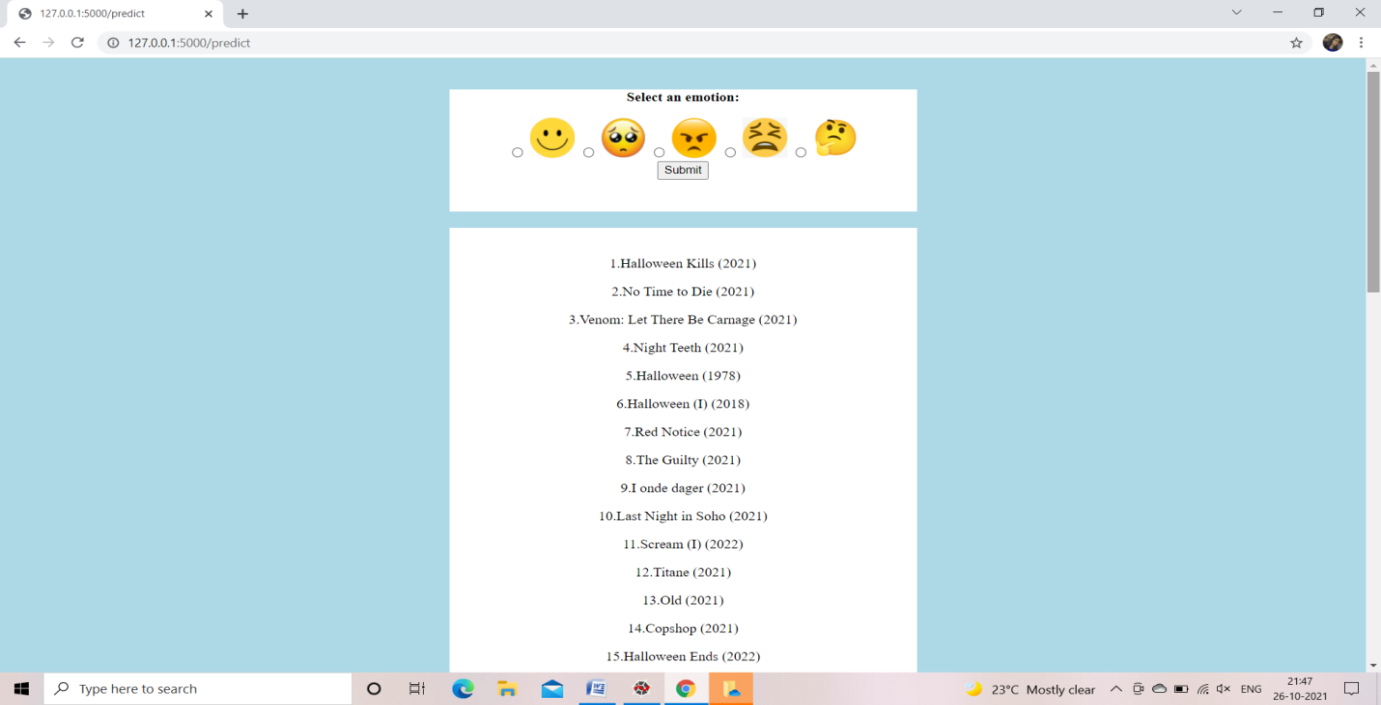
**Fig 6.2: Home Page**

The Fig 6.2 displays Different Emotions for Selection.

****

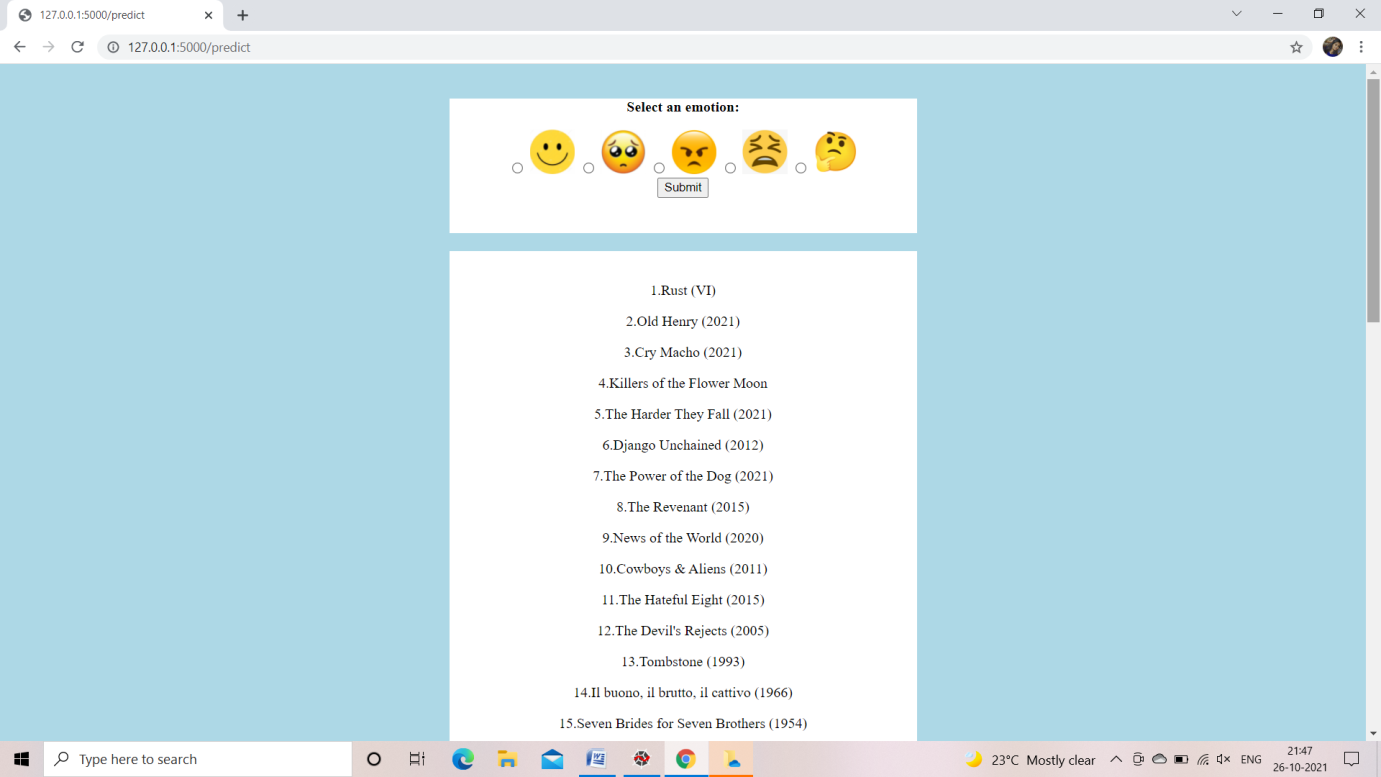
### Fig 6.3: Movie Recommendation Page

The Fig 6.3 displays different movie titles on selecting think emotion.

****

### Fig 6.4: Movie Recommendation Page

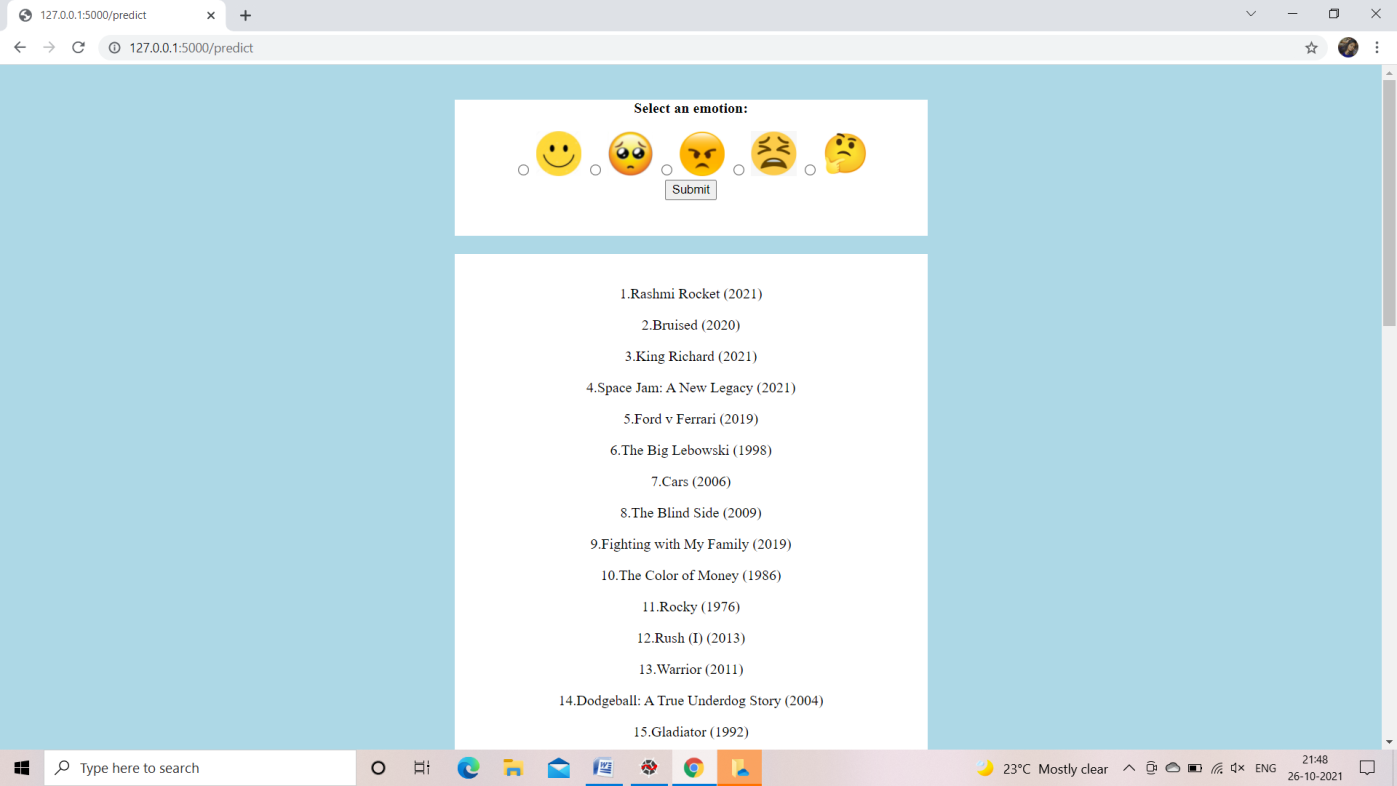
The Fig 6.4 displays different movie titles on selecting angry emotion.

****

**Fig 6.5: Movie Recommendation Page**

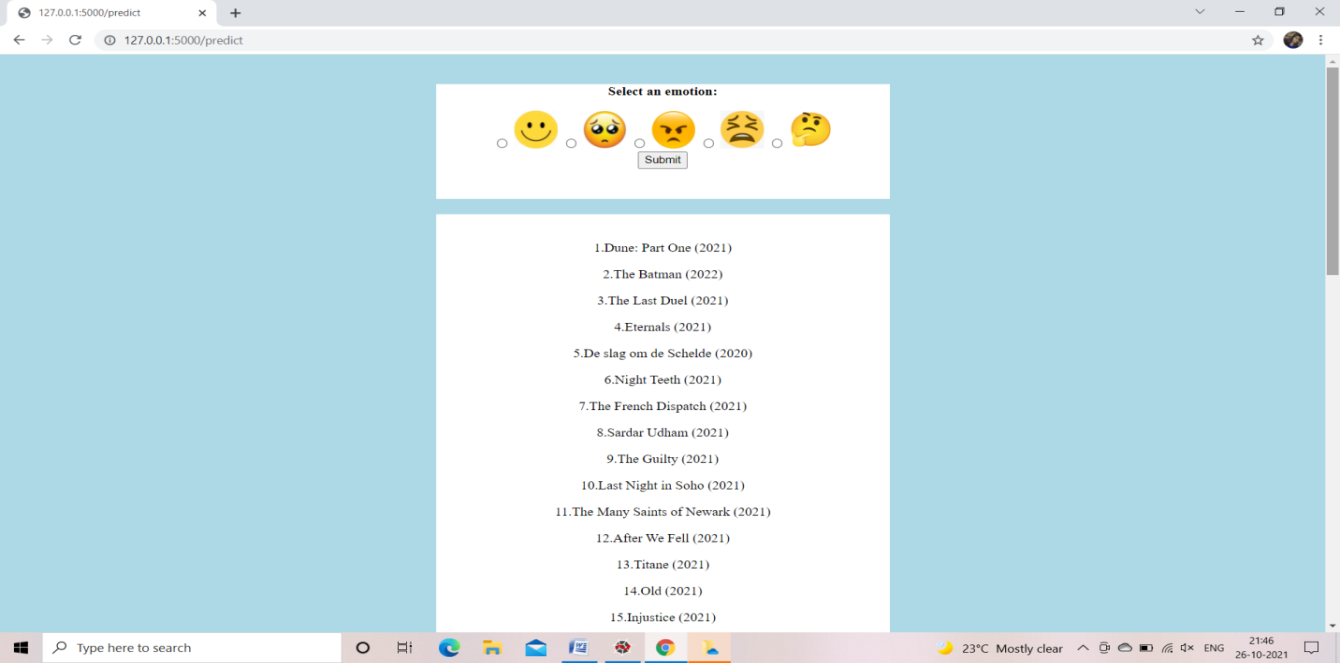
The Fig 6.5 displays different movie titles on selecting sad emotion.

.

****

**Fig 6.6: Movie Recommendation Page**

The Fig 6.6 displays different movie titles on selecting disgust emotion.



### Fig 6.7: Movie Recommendation Page

The Fig 6.7 displays different movie titles on selecting happy emotion.

**7 ADVANTAGES AND DISADVANTAGES**

Advantages

* Web scraping is highly beneficial in extracting data and doing analysis on it. Without this internet wouldn’t exist.so this make to extract movie according to the emotion.
* Recommender system helps to personalize a platform and help the user find movie according to their liked emotions.
* It helps us to filter the movies according the emotion
* It help to predict the user likes
* It increases their profit and also benefits their customer.

Disadvantages

* Lack of Data. Perhaps the biggest issue facing recommender systems is that they need a lot of data to effectively make recommendations.
* Changing Data.
* Changing User Preferences.
* Unpredictable Items.
* This Stuff is Complex

8 APPLICATIONS

Almost any business can benefit from a recommendation system. There are two important aspects that determine how much a business benefits from a recommendation system:

* **Breadth of data:** A business serving only a handful of customers that behave in different ways will not receive much benefit from an automated recommendation system. Humans are still much better than machines in the area of learning from a few examples. In such cases, your employees will use their logic, qualitative and quantitative understanding of customers to make accurate recommendations.
* **Depth of data**: Having a single data point on each customer is also not helpful to recommendation systems. Deep data about customers online activities and if possible offline purchases can guide accurate recommendations

**9 CONCLUSION**

This project has the novelty of incorporating user emotions into the user profile to provide users with well recommended products based on their emotional state. Because movie is a complex and subjective domain, it is necessary to incorporate user emotion into the user profile. Users can give their feedback about how a recommended movie meets their preferences. This feedback (in the form of user’s rating) improves the recommendation quality over time. Because emotion can influence interactions, behaviors and thinking of the user, we believe that E-MRS with the Emotion detector can greatly improve the efficiency of the movie recommendation.

**10 FUTURE WORK**

We have worked hard in order to present an improved application better than the existing one’s regarding the information about the various activities. Still, we found out that the project can be done in a better way. We believe that more emotions can the added and more specific recommendations can be given, thus improving the user satisfaction. We can also include other types of filters such as based on actors, directors or rating based.

**11 BIBILOGRAPHY**

<https://github.com/Guided-Projects/Movie-Recommendation-Based-on-Emotion-Using-Web-Scraping>

<https://github.com/Sk70249/Movie-Recommender-pro>

**APPENDIX**

from flask import Flask, request, render\_template

import numpy as np

import re

import requests as HTTP

from bs4 import BeautifulSoup as SOUP

app = Flask(\_\_name\_\_)

app=Flask(\_\_name\_\_,template\_folder="templates")

@app.route('/',methods=['GET'])

def index():

return render\_template('home.html')

@app.route('/home', methods=['GET'])

def about():

return render\_template('home.html')

@app.route('/predict', methods=["GET","POST"])

def predict():

#emotion=None

#urlhere='http://www.imdb.com/search/title?genres=drama&title\_type=feature&sort=moviemeter, asc'

if request.method == "POST":

emotion=request.form['emotion']

print(emotion)

if(emotion == "happy"):

urlhere = 'http://www.imdb.com/search/title?genres=drama&title\_type=feature&sort=moviemeter, asc'

elif(emotion == "angry"):

urlhere = 'http://www.imdb.com/search/title?genres=thriller&title\_type=feature&sort=moviemeter, asc'

elif(emotion == "disgust"):

urlhere = 'http://www.imdb.com/search/title?genres=sport&title\_type=feature&sort=moviemeter, asc'

elif(emotion == "think"):

urlhere = 'http://www.imdb.com/search/title?genres=thriller&title\_type=feature&sort=moviemeter, asc'

elif(emotion == "sad"):

urlhere = 'http://www.imdb.com/search/title?genres=western&title\_type=feature&sort=moviemeter, asc'

response = HTTP.get(urlhere)

data = response.text

soup = SOUP(data, "lxml")

supa = soup.find\_all('h3', attrs={'class' : 'lister-item-header'})

list = []

for header in supa:

name = "";

aElement\_soup = header.find\_all('a')

spanElement\_soup = header.find\_all('span')

spanElement = spanElement\_soup[0]

name = name + spanElement.text

aElement = aElement\_soup[0]

name = name + "" + aElement.text

if len(spanElement\_soup)>1:

spanElement = spanElement\_soup[1]

name = name + "\n" + spanElement.text

list.append(name)

return render\_template('home.html',prediction\_text="{}".format(emotion),data=list)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True,use\_reloader=False)