 **CS261-Data Structure and Algorithms** 

**Mid Project Proposal (Fall 2021)**

# Proposer Details

|  |  |
| --- | --- |
| Group Number | 62 |
| Registration Number of Group Members | 2020-CS-58  2020-CS-89 |

**Proposal Details**

Proposed Project Title:

Cine-Scrap

### Executive Summary:

Web scrapping is mostly used by Business companies and industries usually with or without any technical knowledge. This project is entirely based upon the scrapping of movies. It works in a way that it extracts the relevant data provided by the user, scrapes it, and will move it on the list shown on GUI. The purpose of web scrappers is to collect as much data as it can. Same is the case with this project, It will scrap data of almost 1 million movies / TV shows. It facilitates the user as much as it can i.e. It will ask user for the genre of movie:

* Comedy
* Sci- Fi
* Horror
* Comedy-Romance
* Fantasy etc.

Every movie has separate attributes through which 7 of them will be scrapped. The attributes for scrapping are:

* Title
* Director
* Ratings
* Duration
* Year
* Type (Language)
* Genre
* Cast

Name of the movie/show

It is GUI based project. The Libraries used for this purpose will be:

* **Pandas** (in case to assemble the data in Data Frame)
* **Beautiful Soap** (that enables us to parse the HTML files)
* **WebDriver** (that simply allows us to automate the chrome)
* **Chrome Options** (allows to manipulate the different properties of chrome)
* **Sleep (**will helps to suspend the code execution for a desired time, as we need to pause it for some time)

The Library for GUI is **PyQt (**This allows flexibility for handling GUI events and makes the coding easier). There are different types of cinemas and theatres. This project will ask the name of the Cinema in which movie is being theatered. Cinema can be DS or Marvel or any required.

This gui based project consists of viewing a list of thousands and millions of shows with sorting and searching techniques. Also it will pause scrapping if the user press the pause button.

## Data Scrapping:

We have scrapped data of movies from [www.imdb.com](http://www.imdb.com) along with there seven attributes like title, year, duration, ratings, genre, certificate, vote that is saved in a csv file.

## GUI:

### Table:

There is a table having 7 columns like title, year, duration, ratings, genre, certificate, vote.

### Labels:

There is a label to start, pause and strop scrapping.

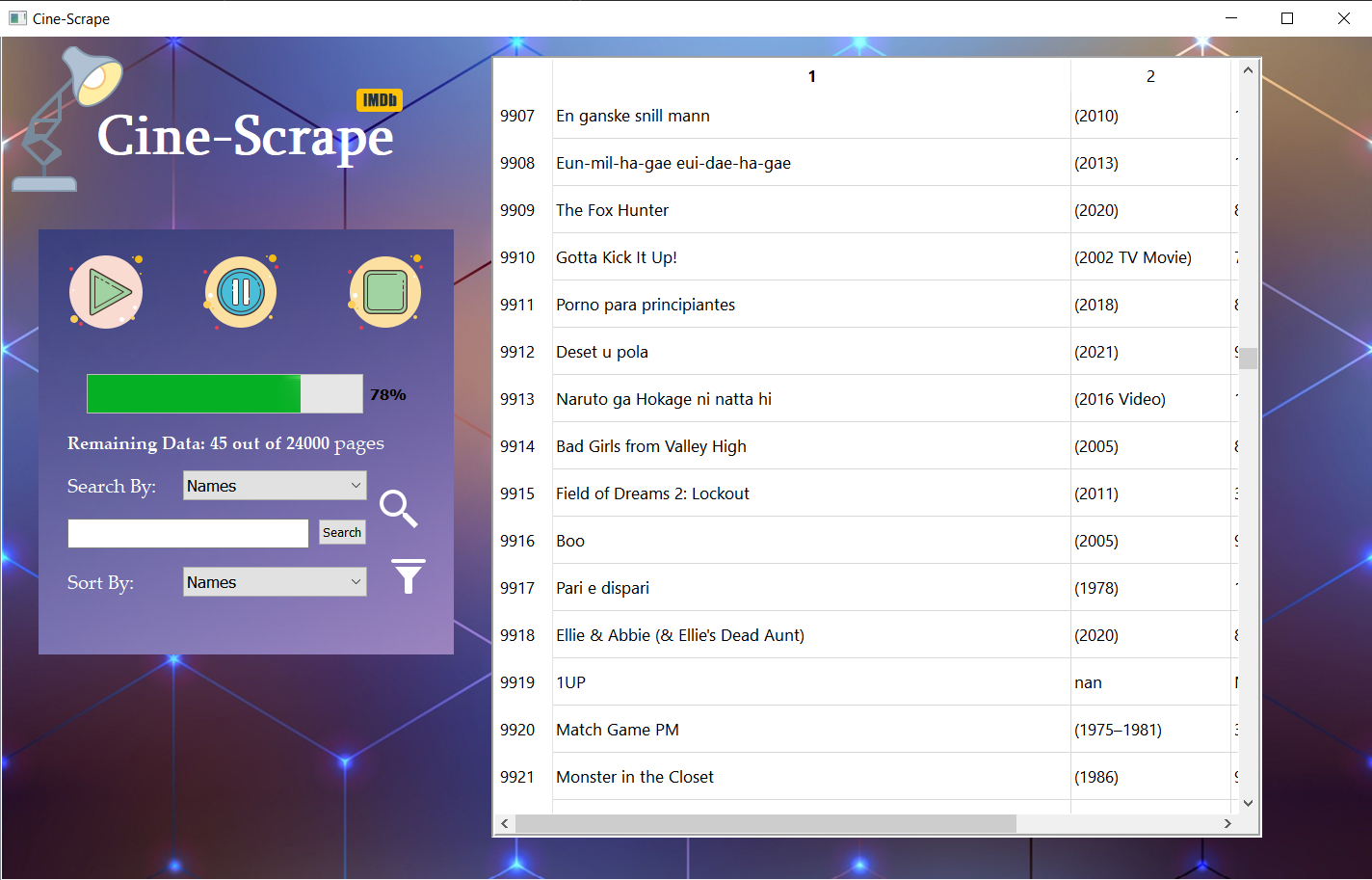
### Progress Bar:

Progress bar represents the scrapping progress.

Combo Box:

A combo box to select sorting algorithms, and to select searching algorithms with provided

filters.



# **Algorithms:**

# Pseudo Codes:

# Insertion Sort:

For i =2 to array. length do

Key = array[i]

j= i-1

while j>0 and array[j]>key do

array[i+1] = array[i]

i=i-1

array[i+1]=key

# Bucket Sort:

For i=0 to A.length - 1 do

B[i]=0

For i=0 to A.length - 1 do

B[i]=B[A.length\*A[i]]

*Sort with Insertion sort and concatenate*

# **Counting Sort:**

k = range of elements of array

count=array of k + 1 zeros

output= array of same length as input

for i = 0 to length(input) - 1 do

j = key(input[i])

count[j] += 1

for i = 1 to k do

count[i] += count[i - 1]

for i = length(input) - 1 down to 0 do

j = key(input[i])

count[j] -= 1

output[count[j]] = input[i]

# **Selection Sort:**

for i = 1 to A.length - 1 do

min = i

for j = i+1 to A.length do

if A[j] < A[min] then

min = j;

if min != i then

swap A[min] and A[i]

# Bubble sort:

for i = 0 to A.length-1 do:

flag= false

for j = 0 to A.length -1 do:

if A[j] > A[j+1] then

swap( A[j], A[j+1] )

swapped = true

If (flag==false) then

break

# Merge Sort (arr[], l, r)

If r > l

m = l+ (r-l)/2

mergeSort(arr, l, m)

mergeSort(arr, m+1, r)

merge(arr, l, m, r)

# Radix sort:

Any stable sort.

# Quick sort:

if low < high:  
 pi = partition(array, low, high)  
 quickSort(array, low, pi - 1)  
 quickSort(array, pi + 1, high)

# Time Complexities:

## Selection Sort:

Best Case: Ω (n^2)

Average Case: θ(n^2)

Worst Case: O(n^2)

## Insertion sort:

Best Case: Ω (n )

Average Case: θ (n^2)

Worst Case: O(n^2)

## Merge sort:

Best Case: Ω (n lg n)

Average Case: θ (n lg n)

Worst Case: O(n lg n)

## Quick sort:

Best Case: Ω (n lg n)

Average Case: θ (n lg n)

Worst Case: O(n^2)

## Bucket sort:

Best Case: Ω (n +k)

Average Case: θ (n + k)

Worst Case: O(n^2)

## Counting sort:

Best Case: Ω (n +k)

Average Case: θ (n + k)

Worst Case: O(n +k)

## Bubble sort:

Best Case: Ω (n)

Average Case: θ (n)

Worst Case: O(n^2)

## Radix sort:

Best Case: Ω (n k)

Average Case: θ (n k)

Worst Case: O(n k)

## Heap sort:

Best Case: Ω (n lg n)

Average Case: θ (n lg n)

Worst Case: O(n lg n)

# **Integration:**

We have 3 files for our program to perform completely. One is the file used for scrapping. Second is the py file of GUI. The third one is the csv file having data. We have to connect our front end with back end code. We will show the data in csv file in a table in Gui using the python code of gui.