Scrapelancer

CS261- Project Proposal

A picture containing text, clipart

Description automatically generated

Project Supervisor

Mr. Samyan Qayyum Wahla

Project Members (Group Id:39)

|  |  |
| --- | --- |
| Khuzaima Umair | 2020-CS-75 |
| Hafiz Mohsin Raza | 2020-CS-81 |

Department of Computer Science

University of Engineering and Technology, Lahore

Pakistan

**Table of Content**

[Project Description: 3](#_Toc87008721)

[Project Features: 3](#_Toc87008722)

[Technology Stack: 4](#_Toc87008723)

[Learning Outcome: 4](#_Toc87008724)

[Algorithms and Analysis 4](#_Toc87008725)

[Sorting Algorithms: 4](#_Toc87008726)

[Insertion Sort: 7](#_Toc87008727)

[Bucket Sort: 9](#_Toc87008728)

[Counting Sort: 11](#_Toc87008729)

[Quick Sort: 13](#_Toc87008730)

[Bubble Sort 15](#_Toc87008731)

[Selection Sort: 17](#_Toc87008732)

[Radix Sort 19](#_Toc87008733)

[Hybrid Sort: 21](#_Toc87008734)

[Searching Algorithms 21](#_Toc87008735)

[Linear Search: 21](#_Toc87008736)

[Binary Search: 22](#_Toc87008737)

[Ternary Search: 22](#_Toc87008738)

[Exponential Search: 23](#_Toc87008739)

[Design 24](#_Toc87008740)

[Implementation 28](#_Toc87008741)

[Testing 31](#_Toc87008742)

[Exceptions: 31](#_Toc87008743)

[Data Storage: 32](#_Toc87008744)

[User Manual 32](#_Toc87008745)

[Project Code: 33](#_Toc87008746)

[Project Plan 57](#_Toc87008747)

[Conclusion: 58](#_Toc87008748)

**List of Figures:**

[Figure 1 for use case of proposed system 25](#_Toc87007473)

[Figure 2 Pencil Tool figure 25](#_Toc87007474)

[Figure 3 Main Window of Scrapelancer 26](#_Toc87007475)

[Figure 4 Dialog box of Sorting, Filtering and Searching data 26](#_Toc87007476)

[Figure 5 Loading data window of Scrapelancer 27](#_Toc87007477)

[Figure 6 Time Calculation window of Scrapelancer 27](#_Toc87007478)

[Figure 7 Storing data in csv file of Scrapelancer 28](#_Toc87007479)

[Figure 8 Gannt Chart for Scrapelancer planning 58](#_Toc87007480)

# Project Description:

Scrapelancer is a web scraper who can scrape data from any freelancing website. The main objective is to manage and manipulate huge amounts of data using the concepts of design and analysis of algorithms. We efficiently work on such sorting algorithms to sort all the scraping data and apply different types of filters in this project.

This project helps us to maintain huge amount of data and store in a local database system like in excel file and it helps us for upcoming freelancing websites and related businesses for analyzing marketplace competition and in this way, it is easy for compete them against their competitors. Most of websites has no option of saving and download data directly and it helpful for those websites who have restrictions of saving data. The data we scrape is use only for learning purpose.

In data structures and algorithm, to do anything, you really want to have information nearby. To get that information, you'll need to explore the necessary sources, and Scrapelancer helps you. Scrapelancer gathers and sorts every one of the necessary information in single area. We use different sorting algorithms to sort all scraped data and uses different types of filters for sorting data like sort by name alphabetically, sort by reviews of seller that has maximum and sort by ratings of seller that has maximum earned to minimum, sort by maximum price earned. We also use multilevel sorting in our project like two rows and column sort at a time.

Scrapelancer has fully automatic, and it also uses different sorting algorithm of searching like Google.

Google searching algorithm is very efficient and it uses whole world. These searching algorithms has very efficient and i.e., linear search and binary search, exponential search, and ternary search. We also use hybrid algorithm like merging two algorithms i.e., Merge sort and Insertion sort and it is very effective for sorting huge amount of data in a short time.

In this way, this project saves a lot of time for user that need this type of data and easy for analyzing and manipulating data in a short time.

We have faced many difficulties during this project like first we select **fiverr** freelancing website for scraping freelancer’s profile data and their users. But after few days of struggle and use all methods, we are unable to scrape data from this website. The main reason is that **fiverr** has some security restrictions for scraping their user’s information so that we are searching for new websites that has freelancers’ information. Finally, we find a **freelancer** website for scraping their user’s information that has minimum seven entities. At the end, we successfully scraped one million data that was final target of this project and after we apply sorting and searching algorithms on that data for end user project for helpful to analyzing such huge information.

## Project Features:

A scrapelancer has following features:

* It has a load button, when the user taps on the load button, it begins loading scraped information.
* It has a Pause button, when the user clicks on the Pause button, it Pause Scraping information.
* It has a Resume button, when the user clicks on the resume button, it begins again scraping information.
* It has a stop button, when the user clicks on the stop button, it quits scraping information.
* It has a sort of button, when the user clicks on the kind of button, it sorts all scraping information.
* It has a filter button, when the user clicks on the filter button, it begins filtering scraped information by applying different filters.
* It has a search button, when user clicks on search button it searches data which you want.

After scraping the data from a website, Scrapelancer can store the data as a CSV file, in this file detailed of data scraped from the test freelancing website, and detail of every attribute of which correspond to each entry.

# Technology Stack:

|  |  |
| --- | --- |
| Language | Python |
| Platform (Web/Desktop) | Web |
| Frontend Technology (Simple HTML/Bootstrap/any other library) | PyQt5/Qt Designer |
| IDEs | Jupyter Notebook, PyCharm |
| Web Browser for Testing | Primarily Google Chrome, Firefox and Microsoft Edge. |

# Learning Outcome:

The objective of scrapelancer is to be learn about different libraries of python and how to scrape, manage and manipulate huge information in automatic way and then apply different algorithms and checking which algorithm is more efficient. In the end, Scrapelancer involved many programming tasks and in this way, we are able to scrape, parse and investigate any kind of information in a short time.

# Algorithms and Analysis

# Sorting Algorithms:

We use following sorting algorithms in our project for sorting all scraped data using different filters.

### **Merge Sort:**

It is a sorting strategy utilized for sorting out and arranging the unordered array in an ordered manner and using recursive methods (divide and conquer approach). This algorithm partitions the array elements into equivalent parts, sort them in a proper way and merge both halves.

#### **Pseudocode:**

|  |
| --- |
| Procedure Merge\_Sort(Arr, L, M):      if L > M          return      N = (L+M)/2      Merge\_Sort(Array, L, N)      Merge\_Sort(Array, M, N)      Merge(Array, L, M, N) |

#### **Python Code:**

|  |
| --- |
| def merge(arr, s, m, e):#S=start,m=middle,e=end,l=left,r=right      i = j = 0      k = start      left = arr[s:m]      right = arr[m:e]        while i < len(l) and j < len(r):          if l[i] < r[j]:              arr[k] = l[i]              i += 1          else:              arr[k] = r[j]              j+= 1          k += 1        while i < len(l):          arr[k] = l[i]          i += 1          k += 1        while j < len(r):          arr[k] = r[j]          j += 1          k += 1    def mergeSort(arr, e, s = 0):      if e == s +1:          return arr[s]        m = (s + e) // 2      mergeSort(arr, m, s)      mergeSort(arr, e, m)      merge(arr, s, m, e) |

**Analysis of Merge Sort:**

,

**Array:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0  Start | 2 | 9 | 3 | 8 | 7  End |

Input Array: [0,2 ,9 ,3, 8 ,7 ]                                                    Output:[0, 2, 3, 7, 8,9]

|  |  |
| --- | --- |
| Strengths | Weakness |
| Merge sort is faster for bigger lists because unlike bubble and insertion sort it does not go through the complete list multiple times. | It is slower relative to the next sort calculations for more modest operations. |
| It utilizes different bits with similar times in a stage. | It utilizes extra memory space to reserve the subcomponents of the array's elements. |
| It has a consistent running time. | It also takes O(nlogn) operations whether the array is already sorted. |
| It is useful when the array is not sorted at all. |  |

#### **Proof of correctness:**

When there is only one element, it is already sorted. Merging two arrays takes linear time. The elements are merged and then sorted. So, we have one portion of the array sorted. Similarly, we repeat the steps for the second end. Resultantly, the merge sort algorithm sorts the array.

## Insertion Sort:

It is a sorting technique where the parts are moved one by one to the left position. Insertion sort helps in arranging the last arranged element, everything thus, with the improvement of higher-situated parts.

### **Pseudocode:**

|  |
| --- |
| Procedure InsertionSort(array)  for k <- 2 to length[array]      do key1 <- array[k]         Insert A[j] to array[1 . . k - 1].         index <- k - 1         //While loop for inserting element in sorted order         while index > 0 and array[index] > key1           do array[index + 1] <- array[index]               index <- index - 1        array[index + 1] <- key1        end while  end for |

### **Python Code:**

|  |
| --- |
| def insertionSort(array):      n = len(array)      for index in range(n):          key = array[index]          k = index -1          while k >= 0 and array[k] > key:              array[k+1] = array[k]              k-=1          array[k+1] = key |

### **Analysis of Insertion Sort:**

=

-1) +

Thus, T(n) of is quadratic equation.

**Array:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0  Start | 2 | 9 | 3 | 8 | 7  End |

Input Array: [0,2 ,9 ,3, 8 ,7]                                                    Output: [0, 2, 3, 7, 8,9]

|  |  |
| --- | --- |
| Strengths | Weakness |
| .It is the stable sort. | It is inefficient when arrays are large or reversed sorted. |
| It is the best choice for arrays that have only a few unsorted elements. | The insertion sort does not manage a tremendous and bigger list. |
| It is the best choice when most of the elements are already sorted. | Its worst time complexity is O(n^2) which is not good. |

### **Proof of correctness:**

When there is only one element in the array it is sorted. Here arr[1..i] is sorted. After every iteration, one more element is added to this subarray and placed at a suitable position.  
It also becomes sorted .In the end, when i is equal to n then arr[1..n] i.e. the whole array becomes sorted.

## Bucket Sort:

Bucket Sort is a sorting method that places things in buckets or classifications. These things are then focused on or positioned by significance, first by class and afterward by explicit things inside every classification**.**

### **Pseudocode:**

|  |
| --- |
| Bucket-Sort(A)  Let B[0...n-1] to be new array  N = A.Length  make N buckets  for every one of the buckets  for i = 0 to n-1  introduce bucket B[i] with 0  for every one of the buckets  for i = 0 to n  drive components into buckets coordinating with the reach  for every one of the buckets  for i = 0 to n-1  Sort-List B[i] with insertion-sort  apply arranging components of each bucket  accumulate the above three List together n |

### **Python Code:**

|  |
| --- |
| def insertionSort(array\_element):      num = len(array\_element)      for index in range(num):          key1 = array\_element[index ]          k= index -1          while k>= 0 and array\_element[k] > key1:              array\_element[k+1] = array\_element[k]              k-=1          array\_element[k+1] = key1    def bucketSort(array\_element):      arr1 = [0] \* 10      output = []      for index in range(0,len(array\_element)):          k=int(array\_element[index]\*len(arr1))-1          arr1[k]=[]          arr1[k].append(array\_element[index])      for index in range(0,len(arr1)):          if(arr1[index]!=0):              insertionSort(arr1[index])              for k in range(0,len(arr1[index ])):                  output.append(arr1[index ][k])      return output    array\_element= [0.879,0.565,0.656,0.1234,0.665,0.343]  print("Output:",bucketSort(array\_element)) |

**Analysis of Bucket Sort:**

Array of bucket

**Array** = **{0.56,0.67,0.22,0.44}**.

Total elements in an array: n=4

So, four buckets require to sort this array

Max element = 0.67

So, by using this equation: Bucket =

The index in each bucket element of array starts from 0.

Before Sorting

|  |  |  |  |
| --- | --- | --- | --- |
| 0.56 | 0.67 | 0.22 | 0.44 |

After Sorting:

We assume each box of array as bucket and apply bucket sort algorithm to this array.

|  |  |  |  |
| --- | --- | --- | --- |
| 0.22 | 0.44 | 0.56 | 0.67 |

|  |  |
| --- | --- |
| **Strengths** | **Weakness** |
| It sorts the array in linear time. | The issue is that if the buckets are circulated erroneously, you might end up spending a great deal of additional work for no or very little increase. |
| It is a stable sort. | It is only efficient when the array is uniformly distributed. |
| It is efficient for larger inputs. | It is only useful for numbers ranging from 0 to 1.  Conversion between numbers and this form may consume much time. |

### **Proof of correctness:**

Bucket sort divides the elements into buckets. some buckets have more than one element, so, each bucket is sorted using insertion sort (any other sorting algorithm can also be used, but, insertion sort is better for smaller inputs). In the end, elements from each bucket are added to the array. Hence, the array becomes sorted

## Counting Sort:

It is a sorting calculation that sorts out the components with the procedure of counting the number of each element how many time they occur in the array. Counting calculation is fundamentally a hashing method with the keys between a particular reach and afterward counting the number of items.

### **Pseudocode:**

|  |
| --- |
| CountingSort(L,M,N)  C(0...N) be new array  for index = 0 to n      C[index] = 0  for k =1 to L.length or N       C[L[key1]= C[L[key1]]] + 1       for index = 1 to key1             C[L[key1]= C[index]+C[L[index-1]]]  for k = n or L.length down to 1  M[C[L[k]]] = L[K]  C[M[k]]=C[M[k]-1] |

### **Python code:**

|  |
| --- |
| def countingSort(arr):      size\_of\_arr = len(arr)      output = [0] \* size\_of\_arr        minimum = min(arr)      k = max(arr) - minimum + 1      counts = [0] \* k        for m in range(0, size\_of\_arr):          counts [arr[m]-minimum] += 1        for m in range(1, k):          counts [m] += counts [m - 1]        m = size\_of\_arr- 1      while m >= 0:          output[counts [arr[m] - minimum] - 1] = arr[m]          counts [arr[m]-minimum] -= 1            m -= 1        for m in range(0, size\_of\_arr):          arr[m] = output[m] |

**Analysis of Counting Sort:**

Array = {6,1,5,8,9,2,1,5,6,2}

Given Array = A

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 1 | 5 | 8 | 9 | 2 | 1 | 5 | 6 | 2 |

New Array = B

It has same size and use for sorted output:

0 1 2 3 4 5 6 7 8 9

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |

Before Counting:

0 1 2 3 4 5 6 7 8 9

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 1 | 5 | 8 | 9 | 2 | 1 | 5 | 6 | 2 |

Start Counting:

0 1 2 3

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 7 | 8 |

After Counting:

Array C is in-between A and B:

0 1 2 3 4 5 6 7 8 9

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 2 | 2 | 5 | 5 | 6 | 6 | 8 | 9 |

|  |  |
| --- | --- |
| Strengths | Weakness |
| It sorts the array in linear time. | When the difference between the maximum and minimum element (range) becomes greater, It becomes inefficient. |
| It is the stable sort. | It is efficient only for discrete numbers of elements. |
| For closely placed inputs, it is efficient. | It is not efficient for keeping string values. |

### **Proof of correctness:**

The counting sort counts the frequency of each element then finds the cumulative frequency (CF). With the help of CF, we can find the position of every element in the sorted array. So, using this technique, the array becomes sorted.

## Quick Sort:

Quick Sort sorts the array by selecting an element called pivot and sorts it. The process continues for all elements. Consequently, the array becomes sorted. It consists of partitions of arrays. It uses the concept of divide and conquer approach. It works proficiently for bigger-sized data sets.

### **Pseudocode:**

|  |
| --- |
| Quicksort(Arr,L,H)  if  L< H    k = Partitioning(Arr,L,H)    QuickSort(Arr,L,j-1)    QuickSort(Arr,j+1,H) |

### **Python Code:**

|  |
| --- |
| def quickSort (arr, low, high):      if low < high:          pi = partition(arr, low, high)            quickSort(arr, low, pi -1)          quickSort(arr, pi+1, high)  def partition(arr, low, high):        i = low -1      pivot = arr[high]        for j in range(low, high+1):          if arr[j] < pivot:              i += 1              arr[i], arr[j] = arr[j], arr[i]        arr[i+1], arr[j] = arr[j], arr[i+1]        return i+1       def inputArr():      size = int(input('Enter the size of the array: '))      arr = []      for i in range(size):          n = int(input(f'Enter the number {i+1}: '))          arr.append(n)      return arr  a = inputArr()  quickSort(a, 0, len(a)-1)  print(a) |

**Analysis of Quick Sort:**

Probability of first case of splitting:

Cost of first case of splitting:

Probability of last case of splitting:

Cost of last case of splitting:+

T(n)=

T(n) = = 1.39.

The expected runtime for quick sort is O(nlogn).

Quick sort is faster because it uses less space.

### **Dry Run:**

**Sub-Problem Size**

**m                                                         Time Analysis**

**c.m**

**m/2                        m/2                                               2(c.m/2)=c.m**

**m/4                 m/4                    m/4                 m/4                           4(c.m/4)=c.m**

**m/8           m/8          m/8       m/8       m/8        m/8     m/8           m/8             8(cm/8)=cm**

**< m. c = cm**

**< m**

### **Time Complexity and analysis:**

Its average and worst-case complexity is O(n lgn) and O(n2) respectively.

|  |  |
| --- | --- |
| Strengths | Weakness |
| It is one of the best-performing algorithms. | In the worst case, its time complexity is exceptionally helpless which cannot be easily managed i.e n(log2n). |
| It has the ability to manage bigger data sets easily. | Quicksort is recursive if recursion is not accessible, the execution is very convoluted. |
| It is the linear time sorting algorithm. | It is not steady. |

### **Proof of correctness:**

Quick Sort picks the pivot point and sorts it (i.e places it at the position where it has to come in the sorted order). So, one element becomes sorted. When we repeat this step, multiple times, the whole array becomes sorted.

## Bubble Sort

It is a sorting algorithm in which we compare an array of elements or numbers and then replace them if they are not arranged. Bubble sort is not working for a bigger number and amount of data sets.

### **Pseudocode:**

|  |
| --- |
| Function bubblesort(A:list of sorted elements)defined as do:  Swapped = false:  intlize i,j,K  N= length(A)  for j = 1 to N do      for i = 0 to N-1 do      if A[i] > A[i+1] then      temp = A[i]      A[i] = A[ i+1]      A[i+ 1] = temp      end     end  end   Input array A[]  output sorted array |

### **Python Code:**

|  |
| --- |
| def bubbleSort(array\_element):      num = len(array\_element)      for index in range (0, num-1):          for k in range (num-1, index, -1):              # if second element is smaller (descending order), then swap              if (array\_element[k] < array\_element[k-1]):                  array\_element[k], array\_element[k-1] =   array\_element[k-1], array\_element[k]                  print(array\_element) |

**Time Complexity and analysis:**

Its best, average, and worst-case complexity is O(n), O(n2 ) andO(n2) respectively.

|  |  |
| --- | --- |
| Strengths | Weakness |
| It is very easy to learn and understand. | It is a highly inefficient algorithm since it requires too much swapping. |
| When the array is mostly sorted then it is a good approach. | It requires n number of execution cycles for elements to be arranged in an ordered manner. |
| The space necessity is at the very least in bubble sort. | It is inefficient for real-world applications. |

### **Proof of correctness:**

Bubble sort swaps the 2 consecutive elements if they are in the wrong order. Here, arr[1..i] is sorted which has 0 elements before the loop. one element is inserted in every step in this subarray. After the termination of the loop, the whole array is sorted.

**Analysis of Bubble Sort:**

|  |  |  |
| --- | --- | --- |
| i | 0…...j < i | Loop execution number of times |
| 4 | 0,1,2,3 | 4 |
| 3 | 0,1,2 | 3 |
| 2 | 0,1 | 2 |
| 1 | 0 | 1 |

## Selection Sort:

The selection sort algorithm is a sorting algorithm in which we sort the elements by choosing one by one element from an unordered list of elements and placing them in an orderly manner in a list (placing ith smallest element at ith position). This process continues until all elements are arranged in sorted order.

### **Pseudocode:**

|  |
| --- |
| For index = 0 to n-index do:         array = I         For k = I + 1 to N-1 do:           If A(k) < A(array)             array = J           End-If         End-For         num= A(I)         A(I) = A(array)         A(array) = num       End-For |

**Python Code:**

|  |
| --- |
| def find(array, start, key):      for index1 in range(start, len(array)):          if array[index1] == key:              return index1      return -1    def selectionSort(arr, start, n):      index1 = start      while(index1  < n-1):          minIndex = i            for j in range(index1 +1, n):              if arr[j] < arr[minIndex]:                  minIndex = j                arr[minIndex], arr[index] = arr[index], arr[minIndex]          index1 +=1    def SortOrderly(order, arr):      n = len(arr)        ord = 0      i = 0      while i < n-1:          index = find(arr, i, order[ord])          if index == -1:              ord += 1              if ord == len(order):                  selectionSort(arr, i, n)                  break          else:              arr[index], arr[i] = arr[i], arr[index]              i += 1 |

### **Time Complexity and analysis:**

Its best, average, and worst-case complexity is O(n), O(n2), and O(n2) respectively.

|  |  |
| --- | --- |
| Strengths | Weakness |
| It is efficient for the least number of array lists. | It requires n number of execution cycles for elements to be arranged in an ordered manner. |
| Its presentation is handily impacted by the underlying requesting of the things before the arranging system. | It is inefficient for large data sets. |
| It doesn't rely upon the primary information. | In the worst-case, It requires a much larger time O(n^2) for which it is not stable. |

### **Proof of correctness:**

The selection sort first finds the ith smallest element from the array and then places it at the ith position where i goes from 1 to n. So, in the end, each element is at its correct position. (i.e. first component in the sorted list is the littlest among all, and so on)

**Analysis of Selection Sort:**

## Radix Sort

Radix Sort is a sorting algorithm that sorts by sorting the numbers digit by digit. This algorithm uses counting sort so to sort the array.

### **Pseudocode:**

|  |
| --- |
| Radix-Sort(A,d)  For i = 1 to d  N= A.length  Exponent =10[i-1]  By using Bubble sort to sort an array digit of array  For j = 1 to n  For k = j + 1 to n  If(A[j]/exp mod 10 >A[k]/exp mod 10)  Exchange A[i] with A[j] |

### **Python Code:**

|  |
| --- |
| def countingSort(arr,size, n):      buckets = [0]\*10      output = [0]\* size        # add the positions      for x in arr:          number = getNumber(x, n)          buckets[number] += 1        stop = buckets[0] >= size -1          # find the cumulative sum      for i in range(len(buckets)-1):          buckets[i+1] += buckets[i]          # sort the array      for i in range(size-1, -1, -1):          digit = getNumber(arr[i], n)          pos = buckets[digit]          buckets[digit] -= 1          output[pos-1]= arr[i]        return output, stop  def inputArr():      size = int(input("Enter the size of the array: "))        arr = []      for i in range(size):          element = int(input(f"Enter the number {i+1}: "))          arr.append(element)      return arr  # arr = [.5,.4,.3,.2,.1]  arr1 = inputArr()  print(radixSort(arr1, len(arr1))) |

### **Time Complexity and analysis:**

Its best, average, and worst-case complexity is O(nk), O(nk), O(nk) respectively.

|  |  |
| --- | --- |
| Strengths | Weakness |
| It is a proficient non-correlation based on arranging the elements. | It only sorts the array linearly. |
| It utilizes counting sort initially for their efficient calculation. | It needed more space than quicksort. |
| Its time complexity is O(nk). | It is less adaptable. |

### **Proof of correctness:**

It sorts the numbers on the basis of the unit place, then, applies the stable sort on the basis of the 10th place. The process continues until comparing digit of every number becomes zero.

**Analysis of Radix Sort:**

Array = {10,15,1,60,5,100}

First, we check the highest number in the array. Now selected number =100.Because there are 3-digits, In first pass: we select least significant digit of array number and apply counting sort on that array and the write in sequence, so array becomes = {010,015,001,060,005,100}.

0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 10**0**  06**0**  01**0** |  |  |  |  |  | 00**5**  01**5** |

In second pass: we select second significant digit of array number and the write in sequence: {010,060,100,001,015,005}

0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0**0**5  0**0**1  1**0**0 | 0**1**5  0**1**0 |  |  |  |  | 0**6**0 |

In third pass, we select highest significant digit of array number and write in sequence: {100,001,005,010,060}

0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **0**60  **0**15  **0**10  **0**05  **0**01 | **1**00 |  |  |  |  |  |

After sorting array:

0 1 2 3 4 5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 5 | 10 | 15 | 60 | 100 |

## Hybrid Sort:

Some algorithms work well for smaller inputs and some for larger inputs. In hybrid sort we make an algorithm efficient by choosing between two algorithms based on the size of the array.

It is the update in the previously built algorithm.

* Insertion Sort
* Merge Sort

# Searching Algorithms

## Linear Search:

### **Pseudocode:**

|  |
| --- |
| LinearSearch(arr, start, end, key)      i = start      while i < end          if (arr[i] == key)              return i          i++      return -1 |

|  |  |
| --- | --- |
| Strengths | Weakness |
| It is the best searching algorithm for the unsorted array | There are too many comparisons in it. |
| It is easy to use not complex loop implementation. | Its worst time complexity is O(n). |
| It’s time complexity is O(1). | It is not much efficient. |

**Analysis of Linear Search:**

In Linear Search we analyze that:

So, Linear Search take O(n) times.

## Binary Search:

## This searching algorithm is used for the sorted arrays. It compares the key with the middle element. If the middle element is the same as the key, return it, else goes to the left or the right half of the array depending on the middle element.

### **Pseudocode:**

|  |
| --- |
| BinarySearch(arr, start, end, key):      if start > end:          return -1      size = end - start      middle = (end-start)/2      if arr[middle] == key:          return middle      if arr[middle] > key          return BinarySearch(arr, start, middle-1, key)      if arr[middle] < key          return BinarySearch(arr, middle+1, end, key) |

|  |  |
| --- | --- |
| Strengths | Weakness |
| It is faster algorithm. | It is only applicable for sorted arrays. |
| It reduces the weakness of both algorithms | not good for searching the extreme left or right elements |
|  |  |

**Analysis of Binary Search:**

Substitute (2) in (1)

Substitute (4) in (3)

So pattern identified:

In binary search algorithm, the problem size is go on reducing until we reach only one element in the array and when we reach only one element in the array then we must check the element we are searching for it and even if last element in our range is not the element we are searching for then we say that the element has not been found and return -1 so at some point, as diminishes, we reach only in an array so

Take log2 on both sides

## Ternary Search:

## It is the same as the binary search except it divides the array into three parts.

### **Pseudocode:**

|  |
| --- |
| TernarySearch(arry, st, ed, key)  if st <= ed then  mid1 := st + (ed - st) /3  mid2 := mid1 + (ed - st) / 3  if array[mid1] == key  return mid1  if array[mid2] == key  return mid2  if key < array[mid1]  TernarySearch(array, st, mid1-1, key)  if key > array[mid2]  TernarySearch(array, mid1+1, ed, key)  else  TernarySearch(array, mid1+1, mid2-1, key)  else  return -1 |

|  |  |
| --- | --- |
| Strengths | Weakness |
| It is good for large-sized inputs. | It is only applicable for sorted arrays. |
| It divides the array into 3 equal parts and results in a faster search than binary. | The numbers which are not multiples of 3 take more time to search. |
|  | When the data is not distributed properly, then it may take more time |

**Exponential Search:**

### **Pseudocode:**

|  |
| --- |
| ExponentialSearch(array, st, ed, key)  if (ed – st) <= 0  return -1  i := 1  while i < (ed - st)  if array[i] < key  i := i \* 2 //increase i as power of 2  else  break  BinarySearch(array, i/2, i, key) |

**Time Complexity:**

* O(1) for the best case. O(log2 i) for the worst or average case. Where i is the location where the element ‘key’ is present.
* Space Complexity: O(1)

|  |  |
| --- | --- |
| **Strengths** | **Weakness** |
| It is good for large sized arrays. | It is only applicable for sorted arrays. |
| It is highly efficient. | It’s inefficient for smaller inputs. |

**Time Complexity of all algorithms:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Algorithm** | **Best time Complexity** | **Average time Complexity** | **Worst time Complexity** | **Worst Space Complexity** |
| Linear Search | O(1) | O(n) | O(n) | O(1) |
| Binary Search | O(1) | O(logn) | O(logn) | O(1) |
| Ternary Search | O(1) | O(log3 i) | O(log3 i) | O(1) |
| Exponential Search | O(1) | O(log2 i) | O(log2 i) | O(1) |
| Bubble Sort | O(n) | O(n2) | O(n2) | O(1) |
| Selection Sort | O(n2) | O(n2) | O(n2) | O(1) |
| Insertion Sort | O(n) | O(n2) | O(n2) | O(1) |
| Merge Sort | O(nlogn) | O(nlogn) | O(nlogn) | O(n) |
| Quick Sort | O(nlogn) | O(nlogn) | O(n2) | O(logn) |
| Radix Sort | O(nlogn) | O(nlogn) | O(n2) | O(logn) |

# Design

Use Cases and design of proposed system:

Diagram

Description automatically generated

Figure for use case of proposed system

Table

Description automatically generated

Figure Pencil Tool figure

**Implementation:**

In implementing phase, first we import different types of libraries like beautiful soap. Beautiful soap is python framework uses for parsing webpages in orderly manner and it helpful for during scraping. Next, we use selenium web driver that allow for testing and help in execution of our script for scraping. Here, we are importing the libraries which we are going to use in our scraping. Some of the libraries are imported for the handling the exceptions.

The function “ScrapeCard” takes the card element and collects all information of that user. The user information is then returned in the tuple. The above code is used to scrape all the pages of the website ‘’freelancer.com’’. In case of any error, it produces sound and tries to resolve the error itself. Since, the data of each user is immediately appended in the file, so, there is no need of lists here.

**Scraping code details:**

|  |
| --- |
| from bs4 import BeautifulSoup as bs  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  from selenium.webdriver.common.by import By  from selenium.common.exceptions import TimeoutException  from selenium.common.exceptions import NoSuchElementException  def scrapeCard(card):      try:          #name          name = card.h3.a.text.strip()          #tagline          tagline = card.find('div', 'user-tagline').text            #country          country = card.find('img', 'flag-icon')['title']              #rating          rating = card.find('span', 'Rating Rating--labeled')['data-star\_rating']            #reviews          rev = card.find('span', 'Rating-review').a.text.strip()          #hourly rate          rate = card.find('span', 'user-hourly-rate freelancer-hourlyrate').text                #earningLabel          earningLabel = card.find('span', 'Earnings-label freelancer-user-earnings').text.strip()            #description          description = card.find('span', 'profile\_text').text.strip()              #Top Skills          skills = card.find('div', 'top-skills').text.strip()            #Profile Picture          imgSrc = card.img['src']        except Exception as e:          print(type(e))          name = ''          tagline = ''          country = ''          cardStats = ''          span = ''          rating = ''          rev = ''          rate = ''          earningLabel = ''          description = ''          skills = ''          imgSrc = ''        return (name, tagline, country, rating, earningLabel, rev, rate, description, skills, imgSrc)  # oldUrl = driver.current\_url  while (True):      try:  #         d = randint(1,3)  #         print(f'sleeping for {d} seconds')  #         time.sleep(d)  #         print('sleep done')          delay = 15          try:  #             myElem = WebDriverWait(driver, delay).until(EC.presence\_of\_element\_located((By.CLASS\_NAME, 'ns\_result')))    #             myElem = WebDriverWait(driver, delay).until(EC.url\_changes))              myElem = WebDriverWait(driver, delay).until(EC.invisibility\_of\_element\_located((By.CSS\_SELECTOR, 'div.Loader.Loader--contained')))              print("Page is ready!")              print(driver.current\_url)                  soup = bs(driver.page\_source, 'html.parser')              cards = soup.find\_all('li', 'ns\_result')              print(len(cards))              with open('headless\_\_freelancer.csv', 'a+', encoding='utf-8', newline='') as f:                  spamWriter = csv.writer(f)                  for c in cards:                      data = (scrapeCard(c))                      spamWriter.writerow(data)              driver.execute\_script('moveToNextPage()')              print('moved to the next page')            except TimeoutException:              print("Loading took too much time!")              print(driver.current\_url)              winsound.Beep(100, 1000)          print()        except Exception as e:          print(e)          winsound.Beep(440, 1000)          print('resolving the issue')            driver.execute\_script(previous)          myElem = WebDriverWait(driver, 20).until(EC.invisibility\_of\_element\_located((By.CSS\_SELECTOR, 'div.Loader.Loader--contained')))          print(driver.current\_url)            driver.execute\_script('moveToNextPage()')          myElem = WebDriverWait(driver, 20).until(EC.invisibility\_of\_element\_located((By.CSS\_SELECTOR, 'div.Loader.Loader--contained')))          print(driver.current\_url)            print(type(e))          winsound.Beep(440, 1000)          print('\a')  # ctrl + enter  winsound.Beep(140, 500)  print('\a') |

# Testing:

# Scrapelancer has testing screens are following:

Graphical user interface, text, application

Description automatically generated Figure Main Window of Scrapelancer

, Graphical user interface, text, application

Description automatically generated

Figure Dialog box of Sorting Filtering and Searching data

A screenshot of a computer

Description automatically generated with medium confidence Figure Loading data window of Scrapelancer

Graphical user interface, text, application, email

Description automatically generated Figure Time Calculation window of Scrapelancer

A screenshot of a computer

Description automatically generated with low confidence Figure Storing data in csv file of Scrapelancer

# Exceptions:

### In this section, there are exceptions that can occur while scraping the data.

|  |  |  |
| --- | --- | --- |
| **Type of Exception** | **Why this exception will occur** | **How you will handle the exception** |
| No internet | When the user is offline | Using the try catch |
| Timeout exception | When the webpage takes too much time to load | Using the try catch statement |
| Element Not found | When some of the attributes are not present on the webpage | Using the try catch block |

# Data Storage:

In these sections, we store scraped data in the following format:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Username | Title | Rating | Amount Earned | Total Reviews | Price | Profile Description | Category | Profile Picture |
| Sharma | Graphic Designer | 5 | 3.8 | 5 reviews | $12 | Hello, my name is Sharma…… | Graphic Designing | Link |

Table

Description automatically generated

*Fig 3.0 for scraped data storage*

# User Manual:

# Scrapelancer is web scraping software application. It is easy to use. The principal objective is to oversee and control gigantic measures of information utilizing the ideas of plan and investigation of calculations. We proficiently work on such arranging calculations to sort all the scraping information and apply various kinds of sorting algorithms in this application.

# Following are the instructions how to use Scrapelancer:

# Step1: Open and Run the Scrapelancer application

# Step2: After open Scrapelancer then click on Load button to Load data in this application.

# Step3: In between Loading of data, there is a button of Pausing and Resuming you want to pause and resume in between data loading and scraping you can do it easily without any restrictions and there is also progress bar to show you how much percentage of data load or scrape so far.

# Step4: After Loading data in Scrapelancer click on Sort Button. There is a dialog window box is open where you can sort your data by choosing different options like: Sort by different sorting algorithms which you want.

# Step5: In dialog box window, there is a Radio button of search where you search your scraped data as your need.

# Step6: In dialog box window, there is a radio button of Filter where you filter your scraped data as your need like apply different filters only data of freelancers having high reviews shows marked.

# Project Code:

Complete Python script for Scrapping data from a website

|  |
| --- |
| from bs4 import BeautifulSoup as bs  from selenium.webdriver.support.ui import WebDriverWait  from selenium.webdriver.support import expected\_conditions as EC  from selenium.webdriver.common.by import By  from selenium.common.exceptions import TimeoutException  from selenium.common.exceptions import NoSuchElementException  def scrapeCard(card):      try:          #name          name = card.h3.a.text.strip()          #tagline          tagline = card.find('div', 'user-tagline').text            #country          country = card.find('img', 'flag-icon')['title']              #rating          rating = card.find('span', 'Rating Rating--labeled')['data-star\_rating']            #reviews          rev = card.find('span', 'Rating-review').a.text.strip()          #hourly rate          rate = card.find('span', 'user-hourly-rate freelancer-hourlyrate').text                #earningLabel          earningLabel = card.find('span', 'Earnings-label freelancer-user-earnings').text.strip()            #description          description = card.find('span', 'profile\_text').text.strip()              #Top Skills          skills = card.find('div', 'top-skills').text.strip()            #Profile Picture          imgSrc = card.img['src']        except Exception as e:          print(type(e))          name = ''          tagline = ''          country = ''          cardStats = ''          span = ''          rating = ''          rev = ''          rate = ''          earningLabel = ''          description = ''          skills = ''          imgSrc = ''        return (name, tagline, country, rating, earningLabel, rev, rate, description, skills, imgSrc)  # oldUrl = driver.current\_url  while (True):      try:  #         d = randint(1,3)  #         print(f'sleeping for {d} seconds')  #         time.sleep(d)  #         print('sleep done')          delay = 15          try:  #             myElem = WebDriverWait(driver, delay).until(EC.presence\_of\_element\_located((By.CLASS\_NAME, 'ns\_result')))    #             myElem = WebDriverWait(driver, delay).until(EC.url\_changes))              myElem = WebDriverWait(driver, delay).until(EC.invisibility\_of\_element\_located((By.CSS\_SELECTOR, 'div.Loader.Loader--contained')))              print("Page is ready!")              print(driver.current\_url)                  soup = bs(driver.page\_source, 'html.parser')              cards = soup.find\_all('li', 'ns\_result')              print(len(cards))              with open('headless\_\_freelancer.csv', 'a+', encoding='utf-8', newline='') as f:                  spamWriter = csv.writer(f)                  for c in cards:                      data = (scrapeCard(c))                      spamWriter.writerow(data)              driver.execute\_script('moveToNextPage()')              print('moved to the next page')            except TimeoutException:              print("Loading took too much time!")              print(driver.current\_url)              winsound.Beep(100, 1000)          print()        except Exception as e:          print(e)          winsound.Beep(440, 1000)          print('resolving the issue')            driver.execute\_script(previous)          myElem = WebDriverWait(driver, 20).until(EC.invisibility\_of\_element\_located((By.CSS\_SELECTOR, 'div.Loader.Loader--contained')))          print(driver.current\_url)            driver.execute\_script('moveToNextPage()')          myElem = WebDriverWait(driver, 20).until(EC.invisibility\_of\_element\_located((By.CSS\_SELECTOR, 'div.Loader.Loader--contained')))          print(driver.current\_url)            print(type(e))          winsound.Beep(440, 1000)          print('\a')      # ctrl + enter    winsound.Beep(140, 500)  print('\a') |

**All GUI Files**

**Gui.py**

|  |
| --- |
| # from test import Ui\_Dialog  from loader2 import Ui\_MainWindow  from SelectAlgorithm import Ui\_Dialog  import splashScreen  from PyQt5 import QtCore, QtGui, QtWidgets  '''from PyQt5.QtWidgets import (      QApplication,      QLabel,      QMainWindow,      QPushButton,      QVBoxLayout,      QWidget,  )'''  from PyQt5.QtGui import \*  from PyQt5.QtWidgets import \*  from PyQt5.QtCore import \*  #from PyQt5.QtCore import QThreadPool, QRunnable, pyqtSlot    import time  import csv    # SPLASH SCREEN  class SplashScreen(QMainWindow):      def \_\_init\_\_(self):          QMainWindow.\_\_init\_\_(self)          self.ui = splashScreen.Ui\_MainWindow()          self.ui.setupUi(self)          ## UI ==> INTERFACE CODES          ########################################################################          ## REMOVE TITLE BAR          self.setWindowFlag(QtCore.Qt.FramelessWindowHint)          #self.setWindowFlag(Qt.SplashScreen)          self.setAttribute(QtCore.Qt.WA\_TranslucentBackground)          ## DROP SHADOW EFFECT          self.shadow = QGraphicsDropShadowEffect(self)          self.shadow.setBlurRadius(20)          self.shadow.setXOffset(0)          self.shadow.setYOffset(0)          self.shadow.setColor(QColor(0, 0, 0, 60))          self.ui.frame.setGraphicsEffect(self.shadow)          #self.show()          #self.progress()          ## QTIMER ==> START          self.timer = QtCore.QTimer()          ####################################################################################################################################################################################################          self.counter = 95          self.timer.timeout.connect(self.progress)          # TIMER IN MILLISECONDS          self.timer.start(35)      ## ==> APP FUNCTIONS      ########################################################################      def progress(self):            # SET VALUE TO PROGRESS BAR          self.ui.progressBar.setValue(self.counter)          # CLOSE SPLASH SCREE AND OPEN APP          if self.counter > 100:              # STOP TIMER              self.timer.stop()              # SHOW MAIN WINDOW              self.main = Window()              self.main.show()              # CLOSE SPLASH SCREEN              self.close()          # INCREASE COUNTER          self.counter += 1  '''class AlgorithmSelecionDialog(QDialog):      """Employee dialog."""      def \_\_init\_\_(self, parent=None):          super().\_\_init\_\_(parent)          # Create an instance of the GUI          self.ui = Ui\_Dialog()          # Run the .setupUi() method to show the GUI          self.ui.setupUi(self)'''    # dialog  class AlgorithmSelecionDialog(QDialog, Ui\_Dialog):      def \_\_init\_\_(self, parent=None):          super().\_\_init\_\_(parent)          # Create an instance of the GUI          #self.ui = Ui\_Dialog()          # Run the .setupUi() method to show the GUI          self.setupUi(self)              self.addEvents()          def addEvents(self):          self.radioButton\_3.clicked.connect(self.sortClicked)          self.radioButton\_2.clicked.connect(self.searchClicked)          self.radioButton.clicked.connect(self.filterClicked)          def sortClicked(self):          alg = [              'Merge Sort',              'Insertion Sort',              'Counting Sort'          ]          self.comboBox.clear()          for a in alg:              self.comboBox.addItem(a)        def searchClicked(self):          print('radio button 3 clicked')          self.comboBox.clear()          alg = [              'Linear Search',              'Binary Search',              'Ternary Search',              'Exponential Search',          ]            for a in alg:              self.comboBox.addItem(a)        def filterClicked(self):          alg = [              'Filter',          ]            self.comboBox.clear()          for a in alg:              self.comboBox.addItem(a)      # our class  class Window(QMainWindow, Ui\_MainWindow):      def \_\_init\_\_(self, parent = None):            super().\_\_init\_\_(parent)          self.setupUi(self)          self.addEvents()              self.threadpool = QThreadPool()          print("Multithreading with maximum %d threads" % self.threadpool.maxThreadCount())          self.continueLoading = True          self.progressBarValue = 0              def startLoading(self, start):          self.continueLoading = True          print('starting the progress bar')          i = start          while i <= 100 and self.continueLoading:              print(self.continueLoading)              self.progressBarValue = i                self.progressBar.setValue(i)              time.sleep(.05)              i+=1          print('done')            def startLoading2(self, start = 0):          worker = Worker(self.startLoading, start)          self.threadpool.start(worker)          def loadData(self):            startingTime = time.time()          print('loading the data')            with open('data.csv', 'r', encoding='utf-8') as f:              csvReader = csv.reader(f)                  row = 0              for item in csvReader:                  self.tableWidget.setRowCount(row+1)                  i = 0                  for attribute in item:                      self.tableWidget.setItem(row, i, QTableWidgetItem(attribute))                      i += 1                        row +=1            endingTime = time.time()          print('data loaded successfully')          print(f'Loading took {endingTime - startingTime} second(s) to complete')          print(self.tableWidget.rowCount())              '''people = [{"name": "Khuzaima", "age": 18}, {"name": "Musab", "age": 11}]            r = self.tableWidget.rowCount()          self.tableWidget.setRowCount(r+len(people))            row = r          for p in people:              self.tableWidget.setItem(row, 0, QTableWidgetItem(p["name"]))              a = str(p["age"])              self.tableWidget.setItem(row, 1, QTableWidgetItem(a))              row +=1'''          def loadButtonClicked(self):          loadDataWorker = Worker(self.loadData)          self.threadpool.start(loadDataWorker)        def selectDialogButton\_clicked(self):            dlg = AlgorithmSelecionDialog()              if dlg.exec():              print("Success!")          else:              print("Cancel!")            def pauseLoading(self):          self.continueLoading = False        def resumeLoading(self):          print('i\'ll resume loading ')          self.startLoading2(self.progressBarValue)        def addEvents(self):          self.pushButton\_2.clicked.connect(self.startLoading2)          self.pushButton.clicked.connect(self.loadButtonClicked)          self.pushButton\_3.clicked.connect(self.selectDialogButton\_clicked)          self.pushButton\_4.clicked.connect(self.pauseLoading)          self.pushButton\_5.clicked.connect(self.resumeLoading)              #self.timer = QTimer()          #self.timer.setInterval(5)          #self.timer.timeout.connect(self.checkPainter)          #self.timer.start()            def checkPainter(self):            #print(painter)          if (self.painter.isActive()):              print('painter is active')          #print(self.painter.isActive()) |

**Gui2.py**

|  |
| --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_MainWindow(object):      def setupUi(self, MainWindow):          MainWindow.setObjectName("MainWindow")          MainWindow.resize(800, 600)          MainWindow.setStyleSheet("")          self.centralwidget = QtWidgets.QWidget(MainWindow)          self.centralwidget.setObjectName("centralwidget")          self.label = QtWidgets.QLabel(self.centralwidget)          self.label.setGeometry(QtCore.QRect(120, 50, 211, 31))          self.label.setStyleSheet("font: 16pt \"MS Shell Dlg 2\";")          self.label.setObjectName("label")          self.label\_2 = QtWidgets.QLabel(self.centralwidget)          self.label\_2.setGeometry(QtCore.QRect(36, 120, 91, 20))          self.label\_2.setObjectName("label\_2")          self.label\_3 = QtWidgets.QLabel(self.centralwidget)          self.label\_3.setGeometry(QtCore.QRect(36, 170, 51, 20))          self.label\_3.setObjectName("label\_3")          self.pushButton = QtWidgets.QPushButton(self.centralwidget)          self.pushButton.setGeometry(QtCore.QRect(190, 240, 75, 23))          self.pushButton.setStyleSheet("font: 75 14pt \"MS Shell Dlg 2\";")          self.pushButton.setObjectName("pushButton")          self.comboBox = QtWidgets.QComboBox(self.centralwidget)          self.comboBox.setGeometry(QtCore.QRect(160, 170, 181, 22))          self.comboBox.setObjectName("comboBox")          self.comboBox.addItem("")          self.comboBox.addItem("")          self.comboBox.addItem("")          self.comboBox.addItem("")          self.comboBox\_2 = QtWidgets.QComboBox(self.centralwidget)          self.comboBox\_2.setGeometry(QtCore.QRect(160, 120, 181, 22))          self.comboBox\_2.setObjectName("comboBox\_2")          self.comboBox\_2.addItem("")          self.comboBox\_2.addItem("")          self.comboBox\_2.addItem("")          self.comboBox\_2.addItem("")          self.comboBox\_2.addItem("")          MainWindow.setCentralWidget(self.centralwidget)          self.menubar = QtWidgets.QMenuBar(MainWindow)          self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 21))          self.menubar.setObjectName("menubar")          self.menuScrapelancer = QtWidgets.QMenu(self.menubar)          self.menuScrapelancer.setObjectName("menuScrapelancer")          MainWindow.setMenuBar(self.menubar)          self.statusbar = QtWidgets.QStatusBar(MainWindow)          self.statusbar.setObjectName("statusbar")          MainWindow.setStatusBar(self.statusbar)          self.menubar.addAction(self.menuScrapelancer.menuAction())          self.retranslateUi(MainWindow)          QtCore.QMetaObject.connectSlotsByName(MainWindow)      def retranslateUi(self, MainWindow):          \_translate = QtCore.QCoreApplication.translate          MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))          self.label.setText(\_translate("MainWindow", "Sorting Specifications"))          self.label\_2.setText(\_translate("MainWindow", "Sorting Algorithm"))          self.label\_3.setText(\_translate("MainWindow", "Sort By"))          self.pushButton.setText(\_translate("MainWindow", "Done"))          self.comboBox.setItemText(0, \_translate("MainWindow", "Select Column Name"))          self.comboBox.setItemText(1, \_translate("MainWindow", "Select Row Name"))          self.comboBox.setItemText(2, \_translate("MainWindow", "Select Seller\'s Name"))          self.comboBox.setItemText(3, \_translate("MainWindow", "Select Seller\'s Service Price"))          self.comboBox\_2.setItemText(0, \_translate("MainWindow", "Merge Sort"))          self.comboBox\_2.setItemText(1, \_translate("MainWindow", "Selction Sort"))          self.comboBox\_2.setItemText(2, \_translate("MainWindow", "Bubble Sort"))          self.comboBox\_2.setItemText(3, \_translate("MainWindow", "Quick Sort"))          self.comboBox\_2.setItemText(4, \_translate("MainWindow", "Insertion sort"))          self.menuScrapelancer.setTitle(\_translate("MainWindow", "Scrapelancer")) |

**Gui3.py**

|  |
| --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_MainWindow(object):      def setupUi(self, MainWindow):          MainWindow.setObjectName("MainWindow")          MainWindow.resize(800, 600)          MainWindow.setStyleSheet("font: 75 12pt \"MS Shell Dlg 2\";")          self.centralwidget = QtWidgets.QWidget(MainWindow)          self.centralwidget.setObjectName("centralwidget")          self.label = QtWidgets.QLabel(self.centralwidget)          self.label.setGeometry(QtCore.QRect(160, 60, 211, 41))          self.label.setStyleSheet("font: 75 16pt \"MS Shell Dlg 2\";")          self.label.setObjectName("label")          self.line = QtWidgets.QFrame(self.centralwidget)          self.line.setGeometry(QtCore.QRect(160, 110, 171, 20))          self.line.setFrameShape(QtWidgets.QFrame.HLine)          self.line.setFrameShadow(QtWidgets.QFrame.Sunken)          self.line.setObjectName("line")          self.label\_2 = QtWidgets.QLabel(self.centralwidget)          self.label\_2.setGeometry(QtCore.QRect(20, 150, 541, 51))          self.label\_2.setStyleSheet("font: 12pt \"MS Shell Dlg 2\";")          self.label\_2.setObjectName("label\_2")          self.label\_3 = QtWidgets.QLabel(self.centralwidget)          self.label\_3.setGeometry(QtCore.QRect(130, 230, 311, 51))          self.label\_3.setStyleSheet("font: 12pt \"MS Shell Dlg 2\";")          self.label\_3.setObjectName("label\_3")          self.pushButton = QtWidgets.QPushButton(self.centralwidget)          self.pushButton.setGeometry(QtCore.QRect(230, 310, 75, 23))          self.pushButton.setObjectName("pushButton")          MainWindow.setCentralWidget(self.centralwidget)          self.menubar = QtWidgets.QMenuBar(MainWindow)          self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 25))          self.menubar.setObjectName("menubar")          self.menuScrapelancer = QtWidgets.QMenu(self.menubar)          self.menuScrapelancer.setObjectName("menuScrapelancer")          MainWindow.setMenuBar(self.menubar)          self.statusbar = QtWidgets.QStatusBar(MainWindow)          self.statusbar.setObjectName("statusbar")          MainWindow.setStatusBar(self.statusbar)          self.menubar.addAction(self.menuScrapelancer.menuAction())          self.retranslateUi(MainWindow)          QtCore.QMetaObject.connectSlotsByName(MainWindow)      def retranslateUi(self, MainWindow):          \_translate = QtCore.QCoreApplication.translate          MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))          self.label.setText(\_translate("MainWindow", "Process Complete "))          self.label\_2.setText(\_translate("MainWindow", "Congratulations! The  Process of Sorting has been  completed Successfully."))          self.label\_3.setText(\_translate("MainWindow", "Total Time Taken:  2 minutes 5 seconds"))          self.pushButton.setText(\_translate("MainWindow", "OK"))          self.menuScrapelancer.setTitle(\_translate("MainWindow", "Scrapelancer")) |

**Loader2.py**

|  |
| --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_MainWindow(object):      def setupUi(self, MainWindow):          MainWindow.setObjectName("MainWindow")          MainWindow.resize(512, 426)          self.centralwidget = QtWidgets.QWidget(MainWindow)          self.centralwidget.setObjectName("centralwidget")          self.verticalLayout = QtWidgets.QVBoxLayout(self.centralwidget)          self.verticalLayout.setObjectName("verticalLayout")          self.label = QtWidgets.QLabel(self.centralwidget)          self.label.setEnabled(True)          font = QtGui.QFont()          font.setPointSize(36)          self.label.setFont(font)          self.label.setObjectName("label")          self.verticalLayout.addWidget(self.label)          self.tableWidget = QtWidgets.QTableWidget(self.centralwidget)          self.tableWidget.setEnabled(True)          self.tableWidget.setStyleSheet("")          self.tableWidget.setAutoScroll(False)          self.tableWidget.setCornerButtonEnabled(False)          self.tableWidget.setObjectName("tableWidget")          self.tableWidget.setColumnCount(10)          self.tableWidget.setRowCount(0)          item = QtWidgets.QTableWidgetItem()          self.tableWidget.setHorizontalHeaderItem(0, item)          item = QtWidgets.QTableWidgetItem()          self.tableWidget.setHorizontalHeaderItem(1, item)          item = QtWidgets.QTableWidgetItem()          self.tableWidget.setHorizontalHeaderItem(2, item)          item = QtWidgets.QTableWidgetItem()          self.tableWidget.setHorizontalHeaderItem(3, item)          item = QtWidgets.QTableWidgetItem()          self.tableWidget.setHorizontalHeaderItem(4, item)          item = QtWidgets.QTableWidgetItem()          self.tableWidget.setHorizontalHeaderItem(5, item)          item = QtWidgets.QTableWidgetItem()          self.tableWidget.setHorizontalHeaderItem(6, item)          item = QtWidgets.QTableWidgetItem()          self.tableWidget.setHorizontalHeaderItem(7, item)          item = QtWidgets.QTableWidgetItem()          self.tableWidget.setHorizontalHeaderItem(8, item)          item = QtWidgets.QTableWidgetItem()          self.tableWidget.setHorizontalHeaderItem(9, item)          self.verticalLayout.addWidget(self.tableWidget)          self.progressBar = QtWidgets.QProgressBar(self.centralwidget)          self.progressBar.setEnabled(True)          self.progressBar.setStyleSheet("QProgressBar{\n"  "            background-color: rgb(98,114,164);\n"  "            color: rgb(200, 200, 200);\n"  "            border-style: none;\n"  "            border-radius: 10px;\n"  "           text-align: center;\n"  "           \n"  "}\n"  "\n"  "QProgressBar::chunk{    \n"  "            background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:0, stop:0.0170455 rgba(226, 0, 185, 255), stop:1 rgba(21, 25, 255, 255));\n"  "            border-radius: 10px;\n"  "            \n"  "}")          self.progressBar.setProperty("value", 24)          self.progressBar.setObjectName("progressBar")          self.verticalLayout.addWidget(self.progressBar)          self.horizontalLayout = QtWidgets.QHBoxLayout()          self.horizontalLayout.setObjectName("horizontalLayout")          self.pushButton\_3 = QtWidgets.QPushButton(self.centralwidget)          self.pushButton\_3.setObjectName("pushButton\_3")          self.horizontalLayout.addWidget(self.pushButton\_3)          spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)          self.horizontalLayout.addItem(spacerItem)          self.pushButton\_2 = QtWidgets.QPushButton(self.centralwidget)          self.pushButton\_2.setEnabled(True)          self.pushButton\_2.setObjectName("pushButton\_2")          self.horizontalLayout.addWidget(self.pushButton\_2)          self.pushButton\_4 = QtWidgets.QPushButton(self.centralwidget)          self.pushButton\_4.setObjectName("pushButton\_4")          self.horizontalLayout.addWidget(self.pushButton\_4)          self.pushButton\_5 = QtWidgets.QPushButton(self.centralwidget)          self.pushButton\_5.setObjectName("pushButton\_5")          self.horizontalLayout.addWidget(self.pushButton\_5)          self.pushButton = QtWidgets.QPushButton(self.centralwidget)          self.pushButton.setEnabled(True)          self.pushButton.setObjectName("pushButton")          self.horizontalLayout.addWidget(self.pushButton)          self.verticalLayout.addLayout(self.horizontalLayout)          MainWindow.setCentralWidget(self.centralwidget)          self.menubar = QtWidgets.QMenuBar(MainWindow)          self.menubar.setGeometry(QtCore.QRect(0, 0, 512, 21))          self.menubar.setObjectName("menubar")          MainWindow.setMenuBar(self.menubar)          self.statusbar = QtWidgets.QStatusBar(MainWindow)          self.statusbar.setObjectName("statusbar")          MainWindow.setStatusBar(self.statusbar)          self.retranslateUi(MainWindow)          QtCore.QMetaObject.connectSlotsByName(MainWindow)      def retranslateUi(self, MainWindow):          \_translate = QtCore.QCoreApplication.translate          MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))          self.label.setText(\_translate("MainWindow", "Scrapelancer"))          self.tableWidget.setSortingEnabled(True)          item = self.tableWidget.horizontalHeaderItem(0)          item.setText(\_translate("MainWindow", "Username"))          item = self.tableWidget.horizontalHeaderItem(1)          item.setText(\_translate("MainWindow", "Tagline"))          item = self.tableWidget.horizontalHeaderItem(2)          item.setText(\_translate("MainWindow", "Country"))          item = self.tableWidget.horizontalHeaderItem(3)          item.setText(\_translate("MainWindow", "Ratings"))          item = self.tableWidget.horizontalHeaderItem(4)          item.setText(\_translate("MainWindow", "EarningLabel"))          item = self.tableWidget.horizontalHeaderItem(5)          item.setText(\_translate("MainWindow", "Reviews"))          item = self.tableWidget.horizontalHeaderItem(6)          item.setText(\_translate("MainWindow", "Rate"))          item = self.tableWidget.horizontalHeaderItem(7)          item.setText(\_translate("MainWindow", "description"))          item = self.tableWidget.horizontalHeaderItem(8)          item.setText(\_translate("MainWindow", "skills"))          item = self.tableWidget.horizontalHeaderItem(9)          item.setText(\_translate("MainWindow", "imgSrc"))          self.pushButton\_3.setText(\_translate("MainWindow", "Sort"))          self.pushButton\_2.setText(\_translate("MainWindow", "OK"))          self.pushButton\_4.setText(\_translate("MainWindow", "Pause"))          self.pushButton\_5.setText(\_translate("MainWindow", "Resume"))          self.pushButton.setText(\_translate("MainWindow", "Load Data")) |

**SelectAlgorithm.py**

|  |
| --- |
| # -\*- coding: utf-8 -\*-  from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_Dialog(object):      def setupUi(self, Dialog):          Dialog.setObjectName("Dialog")          Dialog.resize(413, 300)          self.verticalLayout = QtWidgets.QVBoxLayout(Dialog)          self.verticalLayout.setObjectName("verticalLayout")          self.label = QtWidgets.QLabel(Dialog)          font = QtGui.QFont()          font.setPointSize(24)          self.label.setFont(font)          self.label.setObjectName("label")          self.verticalLayout.addWidget(self.label)          self.horizontalLayout = QtWidgets.QHBoxLayout()          self.horizontalLayout.setObjectName("horizontalLayout")          self.label\_2 = QtWidgets.QLabel(Dialog)          self.label\_2.setObjectName("label\_2")          self.horizontalLayout.addWidget(self.label\_2)          self.horizontalLayout\_3 = QtWidgets.QHBoxLayout()          self.horizontalLayout\_3.setObjectName("horizontalLayout\_3")          self.verticalLayout\_2 = QtWidgets.QVBoxLayout()          self.verticalLayout\_2.setObjectName("verticalLayout\_2")          spacerItem = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)          self.verticalLayout\_2.addItem(spacerItem)          self.horizontalLayout\_5 = QtWidgets.QHBoxLayout()          self.horizontalLayout\_5.setContentsMargins(20, -1, 20, -1)          self.horizontalLayout\_5.setObjectName("horizontalLayout\_5")          self.radioButton\_3 = QtWidgets.QRadioButton(Dialog)          self.radioButton\_3.setObjectName("radioButton\_3")          self.horizontalLayout\_5.addWidget(self.radioButton\_3)          self.radioButton\_2 = QtWidgets.QRadioButton(Dialog)          self.radioButton\_2.setObjectName("radioButton\_2")          self.horizontalLayout\_5.addWidget(self.radioButton\_2)          self.radioButton = QtWidgets.QRadioButton(Dialog)          self.radioButton.setObjectName("radioButton")          self.horizontalLayout\_5.addWidget(self.radioButton)          self.verticalLayout\_2.addLayout(self.horizontalLayout\_5)          spacerItem1 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)          self.verticalLayout\_2.addItem(spacerItem1)          self.horizontalLayout\_3.addLayout(self.verticalLayout\_2)          self.horizontalLayout.addLayout(self.horizontalLayout\_3)          self.verticalLayout.addLayout(self.horizontalLayout)          self.horizontalLayout\_2 = QtWidgets.QHBoxLayout()          self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")          self.label\_3 = QtWidgets.QLabel(Dialog)          self.label\_3.setObjectName("label\_3")          self.horizontalLayout\_2.addWidget(self.label\_3)          self.horizontalLayout\_4 = QtWidgets.QHBoxLayout()          self.horizontalLayout\_4.setObjectName("horizontalLayout\_4")          self.verticalLayout\_4 = QtWidgets.QVBoxLayout()          self.verticalLayout\_4.setObjectName("verticalLayout\_4")          spacerItem2 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)          self.verticalLayout\_4.addItem(spacerItem2)          self.horizontalLayout\_6 = QtWidgets.QHBoxLayout()          self.horizontalLayout\_6.setContentsMargins(20, -1, 20, -1)          self.horizontalLayout\_6.setObjectName("horizontalLayout\_6")          self.comboBox = QtWidgets.QComboBox(Dialog)          self.comboBox.setObjectName("comboBox")          self.comboBox.addItem("")          self.horizontalLayout\_6.addWidget(self.comboBox)          self.verticalLayout\_4.addLayout(self.horizontalLayout\_6)          spacerItem3 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)          self.verticalLayout\_4.addItem(spacerItem3)          self.horizontalLayout\_4.addLayout(self.verticalLayout\_4)          self.horizontalLayout\_2.addLayout(self.horizontalLayout\_4)          self.verticalLayout.addLayout(self.horizontalLayout\_2)          self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)          self.buttonBox.setOrientation(QtCore.Qt.Horizontal)          self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)          self.buttonBox.setObjectName("buttonBox")          self.verticalLayout.addWidget(self.buttonBox)          self.retranslateUi(Dialog)          self.buttonBox.accepted.connect(Dialog.accept)          self.buttonBox.rejected.connect(Dialog.reject)          self.radioButton\_3.clicked.connect(self.comboBox.update)          QtCore.QMetaObject.connectSlotsByName(Dialog)      def retranslateUi(self, Dialog):          \_translate = QtCore.QCoreApplication.translate          Dialog.setWindowTitle(\_translate("Dialog", "Dialog"))          self.label.setText(\_translate("Dialog", "Choose the algorithm"))          self.label\_2.setText(\_translate("Dialog", "Type of Algorithm"))          self.radioButton\_3.setText(\_translate("Dialog", "Sort"))          self.radioButton\_2.setText(\_translate("Dialog", "Search"))          self.radioButton.setText(\_translate("Dialog", "Filter"))          self.label\_3.setText(\_translate("Dialog", "Algorithm Name   "))          self.comboBox.setItemText(0, \_translate("Dialog", "Algorithm")) |

**SplashScreen.py**

|  |
| --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_MainWindow(object):      def setupUi(self, MainWindow):          MainWindow.setObjectName("MainWindow")          MainWindow.resize(499, 350)          MainWindow.setStyleSheet("\*{\n"  "    alternate-background-color: rgb(255, 0, 0);\n"  "}")          self.centralwidget = QtWidgets.QWidget(MainWindow)          self.centralwidget.setStyleSheet("alternate-background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:0, stop:0.0170455 rgba(185, 38, 38, 255), stop:0.380682 rgba(16, 72, 106, 255), stop:1 rgba(87, 23, 106, 255));")          self.centralwidget.setObjectName("centralwidget")          self.verticalLayout = QtWidgets.QVBoxLayout(self.centralwidget)          self.verticalLayout.setObjectName("verticalLayout")          self.frame = QtWidgets.QFrame(self.centralwidget)          self.frame.setStyleSheet("#frame{\n"  "\n"  "    background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:0, stop:0.0170455 rgba(185, 38, 38, 255), stop:0.380682 rgba(16, 72, 106, 255), stop:1 rgba(87, 23, 106, 255));\n"  "\n"  "background-color: rgb(56, 58, 89);\n"  "    border-radius: 15px;\n"  "}\n"  "\n"  "\* {\n"  "color: rgb(98, 114, 164)\n"  "}")          self.frame.setFrameShape(QtWidgets.QFrame.StyledPanel)          self.frame.setFrameShadow(QtWidgets.QFrame.Raised)          self.frame.setObjectName("frame")          self.verticalLayout\_2 = QtWidgets.QVBoxLayout(self.frame)          self.verticalLayout\_2.setObjectName("verticalLayout\_2")          self.label = QtWidgets.QLabel(self.frame)          font = QtGui.QFont()          font.setPointSize(24)          self.label.setFont(font)          self.label.setStyleSheet("color: rgb(99, 105, 209);\n"  "color: qlineargradient(spread:pad, x1:1, y1:0.834773, x2:0.0170455, y2:1, stop:0.0170455 rgba(185, 38, 38, 255), stop:0.474308 rgba(16, 72, 106, 255), stop:1 rgba(87, 23, 106, 255));\n"  "color: rgb(254, 121, 199);\n"  "")          self.label.setAlignment(QtCore.Qt.AlignCenter)          self.label.setObjectName("label")          self.verticalLayout\_2.addWidget(self.label)          self.label\_2 = QtWidgets.QLabel(self.frame)          font = QtGui.QFont()          font.setPointSize(14)          self.label\_2.setFont(font)          self.label\_2.setAlignment(QtCore.Qt.AlignCenter)          self.label\_2.setObjectName("label\_2")          self.verticalLayout\_2.addWidget(self.label\_2)          self.label\_4 = QtWidgets.QLabel(self.frame)          self.label\_4.setObjectName("label\_4")          self.verticalLayout\_2.addWidget(self.label\_4)          self.progressBar = QtWidgets.QProgressBar(self.frame)          self.progressBar.setStyleSheet("QProgressBar{\n"  "            background-color: rgb(98,114,164);\n"  "            color: rgb(200, 200, 200);\n"  "            border-style: none;\n"  "            border-radius: 10px;\n"  "           text-align: center;\n"  "           \n"  "}\n"  "\n"  "QProgressBar::chunk{    \n"  "            background-color: qlineargradient(spread:pad, x1:0, y1:0, x2:1, y2:0, stop:0.0170455 rgba(226, 0, 185, 255), stop:1 rgba(21, 25, 255, 255));\n"  "            border-radius: 10px;\n"  "\n"  "    background-color: qlineargradient(spread:pad, x1:0, y1:0.511364, x2:1, y2:0.523, stop:0 rgba(254, 121, 199, 255), stop:1 rgba(170, 85, 255, 255));\n"  "            \n"  "}")          self.progressBar.setProperty("value", 24)          self.progressBar.setObjectName("progressBar")          self.verticalLayout\_2.addWidget(self.progressBar)          self.label\_3 = QtWidgets.QLabel(self.frame)          font = QtGui.QFont()          font.setPointSize(11)          self.label\_3.setFont(font)          self.label\_3.setStyleSheet("padding-right: 40px;")          self.label\_3.setAlignment(QtCore.Qt.AlignRight|QtCore.Qt.AlignTrailing|QtCore.Qt.AlignVCenter)          self.label\_3.setObjectName("label\_3")          self.verticalLayout\_2.addWidget(self.label\_3)          self.verticalLayout.addWidget(self.frame)          MainWindow.setCentralWidget(self.centralwidget)          self.statusbar = QtWidgets.QStatusBar(MainWindow)          self.statusbar.setObjectName("statusbar")          MainWindow.setStatusBar(self.statusbar)          self.retranslateUi(MainWindow)          QtCore.QMetaObject.connectSlotsByName(MainWindow)      def retranslateUi(self, MainWindow):          \_translate = QtCore.QCoreApplication.translate          MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))          self.label.setText(\_translate("MainWindow", "Welcome to <strong>Scrapelancer</strong>"))          self.label\_2.setText(\_translate("MainWindow", "Scraping made easy"))          self.label\_4.setText(\_translate("MainWindow", "Loading..."))          self.label\_3.setText(\_translate("MainWindow", "© Made by Group: 39")) |

# Project Plan

This section should include the implementation plan and work division.

Graphical user interface, application

Description automatically generated

Figure 8 Gannt Chart for Scrapelancer planning

# Conclusion:

While this project may not be pretty much as complex as web scrapers made by huge companies, there is sufficient degree in this application to have a huge effect in the world. By utilizing Scrapelancer user can scrape data from any freelancing website and easy to manipulate them.