

# Data Structures and Algorithms

CS-261- Mid Project Final Report



## Project Supervisor

Mr. Samyan Qayyum Wahla

## Project Members (Project ID 24)

**Member 1 Khadija Asif**

**Registration Number: 2020-CS-143**

---

# Contents

Projects Summary	<b>3</b>
What are the benefits? .....	3
How about your project's UI? .....	3
BUISNESS CASE	<b>3</b>
Business need for the project.....	3
End User of the product.....	3
Technical Details (Product's Details):	<b>4</b>
Name of Attributes .....	5
Sample of Scrapping Source.....	5
GitHub Repository Link	<b>7</b>
Project Interface	<b>7</b>
UI Component Name.....	7
Type of UI component.....	7
Algorithms Used in Project	<b>9</b>
Algorithms Details.....	9
Analysis of Data Scrapping	<b>19</b>
Website Source .....	19
Python code of Data Scrapping .....	19
Issues Faced in Using PyQt5 (GUI Implementation):	<b>23</b>



## CS261-Data Structure and Algorithms Mid Project Proposal (Fall 2021)



### Projects Summary

When you want any data or information, you google it or something like that. But if you want the data to be save into your device or computer then it'll be the best way if you directly scrape it and use your data whenever you need. This project provides you the same feature that will help you to scrape data and save it in a csv file.

#### What are the benefits?

Let's suppose, you want to run your own small business and you know nothing about the market price and financial ratios. So, scraped data helps you to understand about some pricing strategies in marketing and all other things that one should must know before starting the business for its better development.

#### How about your project's UI?

The way your user interacts with you and your project. So, it must be your first priority to make it more efficient and responsible. This project provides you with more flexible features that will allows user to view the progress bar while scrapping data and also allow them to resume or pause scrapping data or information.

#### What about Data Arrangement?

When you're talking about data then the user must demand- to sort data in an efficient manner then this project also provides you with this feature.

BUISNESS CASE	
Business need for the project	As mentioned above, before you start some business or enter into the marketing field it is compulsory for you to have some knowledge about pricing strategies of marketing. So, you need some or vast amount of data for this process so if you scrape data for any product it'll be easier to analyze. On the other hand, if you are working for some project for which you need a vast amount of data so it'll help you for that research also.
End User of the product	There are many types of end users. They can be students, companies or any person. They use the product's information for their benefits. If they are from some company or agency, they use the information to shape their business. On the other hand, if the end user is student then he may use the product's information to research for their project.
Motivation for project	The motivation or importance for attempting the project is that it provides you the data in more quick and efficient way and business especially academia and IT industries are totally depending on data nowadays.

State the level of impact expected should the project proceed and implications of not proceeding	For any user that want to buy the new shoes online, now need no more to search and spend time on different websites because the data that we scrapped though this process provides data that contains all the information like information about size, cost and about brands. But what if we are talking about <b>implications of not proceeding?</b> then it may make some logic that to go through all the data is somehow tiring and modern technologies now provide more better option to the end user as compared to information in the form of data.

## Technology Stack:

Language Used	Python
Platform	Desktop
Library imports	Pandas / PyQt5 / Selenium/ BeatifulSoap and some others
Files used for scrapped Data	csv
For GUI (IDE)	Designer.exe
IDE's	Jupyter Notebook / Visual Studio Code / Designer.exe

## Technical Details (Product's Details):



## Name of Attributes


### Name of Attributes

Name of Entity	Shoes		
Attributes of Entity	Name	Data Type	Description
	Name of Product	String	This attribute defines the name of all the products
	Color	String	Provides the color of the product
	Model Number	String	Attribute defines the model number of each product and it is unique for each one
	Price	Int	Original price of each product
	Sales Price	Int	Sale's price if the product is on sale
	Size	Int	Size of each product (each product has one or more than one size)
	Brand	String	Defines the brand of each product
	SKU	String	Attribute defines the stock keeping unit of each product
	Sole Type	String	Defines the category of shoes whether it is flat/heel or related to the shoe's category
	Season	String	Defines in which season the product is suitable to wear


## Sample of Scraping Source





Webpage




Search



 30 days for return  
see details

 Shipping in 24h  
see details

 Cart (0)

WOMEN'S

MEN'S

KIDS'

SPORT

ACCESSORIES

PREMIUM

HANDBAGS



SALE

MID SEASON  
SALE

up to  
**-40%**

Seasonal discounts

CHECK IT





Scrapping Source (Website: “[Efootwear.eu](https://efootwear.eu)”)


Attribute Source


↑


↺ 360°












↓




←

→

Trainers PEPE JEANS

Kenton Britt Man PMS30707 White 800

-26%



See more products

☆☆☆☆☆ (0)

Be the first to review this product


€69.90

You save €17.90


**€52.00**

Choose size


Shoe width: Medium

 Size chart

ADD TO BAG



Shipment within 24h. Ready to ship in 24h

 30 days for return

## Project Interface

Interface for your Project

UI Component Name	Type of UI component	Purpose of UI Component/Other details
Pause_Button	Button	Option will be provided to the user if he/she wants to pause the data scrapping.
Resume_Button	Button	Option will be provided to the user to resume the data scrapping
Progressing_perc	Progress Bar	To show the progress while scrapping data
Data_file	Table	for scrapped data
Sorting_selector	Combo box	It enables users to choose any sorting algorithm to sort scrapped data accordingly

Search\_Button

Button

It'll open another UI screen to search data in a given table.  
**(UI given below)**

Form - [Preview] - Qt Designer

Search Data by Selecting:

	Name	Product ID	Price	Color	Brand	SKU	Sole Type	Sale pe
1								
2								
3								
4								
5								
6								
7								
8								
9								

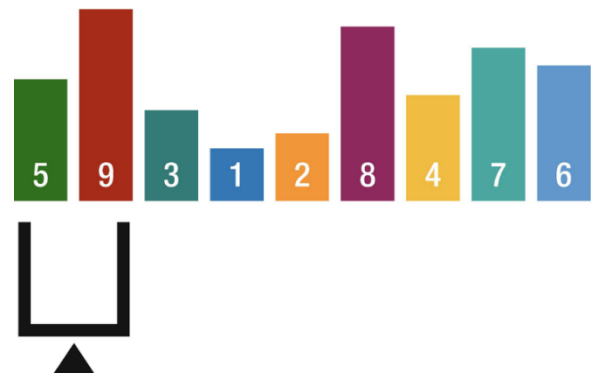
Close

- In the above given UI frame following data (that you searched for) will show in the table.



## Algorithms Used in Project

- ❖ Insertion Sort
- ❖ Selection Sort
- ❖ Merge Sort
- ❖ Bubble Sort
- ❖ Quick Sort
- ❖ Counting Sort
- ❖ Radix Sort
- ❖ Bucket Sort



## Algorithms Details

- **Insertion Sort**

Insertion sort is a sorting algorithm that sort all the given numbers in an array. While sorting numbers using insertion sort. We always assume that the first element is in a sorted form. So, we start comparing the numbers from second element. So set the second number into the key value and start comparing it with the already sorted array or a subarray. This process repeats until all the numbers are in a proper sorted form.

For Example: if we have an array of number  $A = [1, 4, 3, 6, 5]$ . Then we assume 1 to be in a sorted form and start setting the value of key from  $\text{key} = 4$  and compare it with already sorted number or a subarray. If the number is greater than the key value it'll set into its proper position. And at the end we have an output of array  $[1, 3, 4, 5, 6]$ .

- **Pseudo Code:**

1. Outer loop to control the key value starting from the index of an array (**for  $i = 0$  to  $i = \text{max}$** )
2. Set the key value ( **$\text{key} = A[i]$** )
3. Inner loop to compare key value with the already sorted number.

4. If the number is greater than the key value then set it in its correct position (repeats for all the numbers).

- **Code in Python:**

```
#take an array of size n
for i in range (1, size):
    key = list[i]
    j = i - 1
    while (j>=0 and array[j] > key):
        array[j+1] = array[j]
        j = j-1
    array[j+1] = key
```

- **Positive Points of Insertion Sort:**

1. Best for when the given array is already sorted.
2. Efficient for sorting a small number of elements.
3. Insertion sort is the stable sort (it sorts the given element in the same order as it is before)

- **Negative Points of Insertion Sort (or weakness):**

1. Inefficient for sorting large input of Data.
2. Running time problems (because it depends on the size of input)
3. Average time complexity of  $O(n^2)$ .

- **Dry Run using Insertion Sort Algorithm:**

Suppose we have an array  $A = [5, 43, 76, 2]$ . While using the above code when  $i = 1$   $key = 43$ .  $A[j] = 5$

( $j = i - 1$ ). Then it checks condition using inner loop:  $5 > 43$  and  $j \geq 0$  (if it returns true then  $Array[j+1] = Array[j]$  from  $j = 0$  to  $j = 1$  list will be the same.

5, 43, 76, 2

from  $j = 2$  to onward

2, 5, 43, 76 (returns the sorted list)

- **Time Complexity:**

1.  $n$  times
2.  $n - 1$  times
3.  $n - 1$  times
4.  $\sum (t_j)$  from  $i = 1$  to  $n - 1$  time (we use  $t_j$  because it depends on the condition of  $array[i] > key$ )
5.  $\sum (t_j - 1)$  from  $i = 1$  to  $n - 1$  time
6.  $\sum (t_j - 1)$  from  $i = 1$  to  $n - 1$  time
7.  $n - 1$
8.  $n + (n - 1) + \sum (t_j)$  from  $i = 1$  to  $(n - 1)$  times +  $\sum (t_j - 1)$  from  $i = 1$  to  $n - 1$  time +  $\sum (t_j - 1)$  from  $i = 1$  to  $n - 1$  time +  $(n - 1) + (n - 1)$

**Best Case:**  $O(n)$

**Worst Case:**  $O(n^2)$

**Average Case:**  $O(n^2)$

- **Merge Sort**

Merge sort is one of the best sorting algorithms for large input Data. It follows the divide and conquer method or approach (divides the problem into more than one subproblem). It follows the step to call recursive function for each subproblem to sort each subproblem until we reach the subproblem of size 1 and after that combines all the subproblem into one sorted array.

- **Pseudo Code:**

1. Suppose we have an array of size p to r.
2. Divide the array into subparts until  $p < r$ .
3. Recursive call of merge sort when A is from p to q.
4. Step 3 repeats when A is from  $q + 1$  to size.
5. Call function to merge all the sorted subproblems.

- **Code in Python:**

```
#take an array of size n (function for merge to combine all the subproblem in a sorted form)
def merge (A, p, q, size):
    B = []
    p = 0
    K = 0
    j = q
    while (p < q and j < size):
        if(A[p] < A[j]):
            B.append(A[p])
            p = p + 1
        else:
            B.append(A[j])
            j = j + 1
    while(p < q):
        B.append(A[p])
        p = p + 1
    while(j < size):
        B.append(A[j])
        j = j + 1
    for i in range (len(B)):
        A[i] = B[i]
#function for merge sort
def merge_sort (A, p, r):
    if r > p:
        q = (p+1)//2
        merge_sort (A, p, q)
        merge_sort (A, q+1, r)
        merge (A, p, q, r)
```

```
merge_sort (A, p, size)
```

- **Positive Points of Merge Sort:**

1. Works correctly even when the number of inputs is not even.
2. Better than the insertion and other sorting algorithms when the huge elements of an array are given.
3. Running time is  $O(n * \log n)$  for all the cases (running time is consistent).

- **Negative Points of Merge Sort (or Weakness):**

1. Merge Sort follows divide and conquer approach (divides problem into subproblems). So, it takes more space.
2. Not best for already sorted array.
3. Not suitable for small inputs.

- **Dry Run using Merge Sort:**

Let suppose we have an array  $A = [1, 3, 54, 2, 72, 23, 12, 32, 76]$  of size = 9. Then by taking its mid-point we have  $q = r // 2$   $q = 4$ . So, we sort recursively by sorting subarray from 0 to  $q$  and subarray from  $q + 1$  to size and then combine all the sorted subarray. So, output will become  $[1, 2, 3, 12, 23, 32, 54, 72, 76]$ .

- **Time Complexity:**

$O(n * \log n)$  Constant for all case

- **Selection Sort:**

Selection sort is a type of sorting algorithm in which we set minimum value equals to 0 index of array according to the outer loop which then compare it with all other elements by using inner loop if any other minimum value exists. Then swap the positions of both minimum values. This process will repeat until all the elements of an array is sorted.

- **Pseudo Code:**

1. Take an array  $A$  of array of size  $n$ .
2. Store value of 0 index into variable name minimum value using outer loop (from  $i = 0$  to size).
3. Inner loop (for  $j = i+1$  to size) to find value if any smaller than the minimum value (if  $A[j] < A[i]$ ) then update the minimum value and swap their positions. ( $array[i], array[min] = array[min], array[i]$ ).

- **Code in Python:**

```
#take an array of size n  
min = 0
```

```
for i in range (0, max_size - 1):
    min = i
    for j in range (i+1, max_size):
        if(array[min] > array[j]):
            min = j
    array[i], array[min] = array[min], array[i]
```

- **Positive Points of Selection Sort:**

1. Best for using small input of Data.
2. It doesn't require extra space.
3. Best performance for already sorted array.

- **Negative Point of Selection Sort:**

1. It performs worst as compare to other algorithms when the size of input is large or huge.

- **Dry Run Using Selection Sort:**

if we have an array A = [2, 5, 9, 7]. As written in above code when i = 2 we have j = i+1 so A[j] = 7 and A[i] > A[j] so we swap them both and the array is in correct order. So, we have A = [2, 5, 7, 9].

- **Time Complexity:**

O(n<sup>2</sup>) same for all three cases.

- **Bubble Sort:**

Bubble sort is the type of sorting algorithm that compares the elements and simply swap them according to their right positions. It is the simplest but not efficient way of sorting element of an array.

- **Pseudo Code:**

2. Take an array A of size n.
3. Outer for loop (for i = 0 to size -1)
4. Inner for loop (for j = 0 to i -1). In this loop check if A[j] > A[j+1] if it's true then swap them according to their position.
5. This process repeats until the outer for loop terminates and return sorted array.

- **Code in Python:**

```
for j in range (0, max_size - 1)
    swap = false
    for i in range (0, max_size - 1):
        if(list_array[i] > list_array[i+1]):
            list_array[i], list_array[i+1] = list_array[i+1], list_array[i]
            swap = True
    if (swap == False):
        break
```

- **Positive Points of Bubble Sort:**

1. Easy to implement (we just have to compare two values and swap them).
2. Doesn't require much space.

- **Negative Points of Bubble Sort:**

1. It performs worst for the condition when the given input array is huge.

- **Dry Run using Bubble Sort:**

If we have an array  $A = [2, 5, 9, 7]$ . When  $i = 0$  we compare it with all the other elements of an array starting from  $A[i+1]$  if there exist an element greater than  $A[0]$ , we just swap them. When  $j = 0$  and  $i = 0$  check  $A[0] < A[i+1]$  ( $2 < 5$ ) return false. When  $i = 2$ ,  $A[i] > A[i+1]$  (so we simply swap them and we have an array  $A[2, 5, 7, 9]$ . But still this process repeats until the outer loop terminates.

- **Time Complexity:**

Best Case:  $O(n)$

Average Case:  $O(n^2)$

Worst Case:  $O(n^2)$

- **Quick Sort:**

Quick sort is a type of sort algorithm that follows divide and conquer approach (just like merge sort). This sorting algorithm consist of two function (that calls recursively) the first one is partition and another is Quick sort function. In this sorting algorithm we set pivot value (usually we take the high index value as a pivot) according to its right position. (the right side of pivot the elements of an array are greater than the pivot's value and for left side it must be smaller and then we apply quick sort on both sides).

- **Pseudo Code:**

1. Take an array  $A$  of size  $n$ .
2. Quicksort ( $A, p, r$ )
3.  $q = \text{partition}(A, p, r)$  (return the value of pivot).
4. Calls function Quicksort. (first from  $p = 0$  to  $q - 1$  and then from  $q+1$  to  $r$ ).

- **Code in Python:**

```
def partition(A, low, high):
    pivot = A[high]
    j = low - 1
    for i in range(low, high):
        if A[i] < pivot:
            j = j + 1
            (A[i], A[j]) = (A[j], A[i])

    X = A[j+1]
    A[j+1] = pivot
    A[high] = x
    return (j+1)

def Quicksort(A, low, high):
```

```
if (low < high):
    X = partition (A, low, high)
    Quicksort (A, low, X-1)
    Quicksort (A, X+1, high)
Return (A)
```

- **Positive Points of Quick Sort:**

1. Faster than the merge sort when the given input array is small.

- **Negative Points of Quick Sort (or Weakness):**

1. Its time complexity is  $O(n^2)$  which not better than the merge sort (time complexity is  $n * \log n$ ). So, it is not better than the merge sort in this case.

- **Dry Run using Quick Sort:**

If we have an array  $A = [4, 1, 3, 9, 7]$  then according to the above code firstly we have pivot value of 7 which is the 4 index of  $i$  after calling partition function the value of pivot will be 3 and we check whether  $A[i] < \text{pivot}$  when  $i = 0$  ( $4 < 3$ ) which returns false and when  $i = 1$  ( $1 < 3$ ) which returns true and swap  $A[i] = 1$  and  $A[j] = 4$  (because  $j = 0$ ) positions so the subarray will be in the form of  $[1, 4, 3]$ . In this way all the element of array sorted (by calling partition and quick sort function recursively). And it returns  $A = [1, 3, 4, 7, 9]$ .

- **Time Complexity:**

Best Case:  $O(n^2)$

Average Case:  $O(n * \log n)$

Worst Case:  $O(n * \log n)$

- **Counting Sort:**

Counting sort is not a comparison sort it sorts element just by counting the number of existences of all elements and store them in an array.

- **Pseudo Code:**

1. Suppose we have an array of size  $n$ .
2. Create an array  $C$  of size  $(n+1)$  and initialize zero to all the elements of an array.
3. Now increment value by one on each index of an array  $C$  according to the value of an array  $A$  such as  $C[A[i]] = C[A[i]] + 1$
4. Now add each value of an array  $C$  starting from index 1 with the previous index value  $C[i] = C[i] + C[i-1]$
5.  $C[A[i]] = C[A[i]] - 1$
6.  $\text{var} = C[A[i]]$
7. Create an array of  $B$  for storing elements and assign  $B[\text{var}] = A[i]$ .

- **Code in Python:**

```
Size = len(A) #suppose A is an array of given elements
max = np.Max(A)
```

```

min = np.min (A)
C = np. Array ([0 for each element in range (min, (max+2))])
Output = [0 for each element in range (size)]
for i in range (0, size):
    j = A[i]
    C[j] = C[j] + 1
for i in range (min, max+1):
    if (i == min):
        continue
    else:
        C[i] = C[i] + C[i-1]
for i in range (0, size):
    j = A[i]
    C[j] = C[j] - 1
    K = C[j]
    Output[K] = j

```

- **Positive Points of Counting Sort:**

1. The most important point of using counting sort is that it is stable.
2. Can also be used to handle negative inputs.
3. Not perform any comparison operations.

- **Negative Points of Counting Sort:**

1. Not suitable for large input of data.
2. Need more than one array to sort data using Counting Sort.

- **Dry Run Using Counting Sort:**

Suppose we have an array  $A = [0, -3, 8, 5, -1, 10]$ . Then according to the above given code  $C = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$  if  $A[i] = 0$ , increment zero index of  $C$  by 1. Similarly, if  $A[i] = 10$  then  $C[10] = C[10] + 1$ . Then  $C = [1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1]$ . Add  $C[i] = C[i] + C[i-1]$  then  $C = [1, 1, 2, 3, 3, 3, 3, 4, 4, 5, 5, 6]$  if  $A[i] = 0$  and  $C[0] = 3$ ,  $C[0] = C[0] - 1$  (then  $C[0] = 2$ )  $B[C[0]] = A[i]$  ( $B[2] = 0$ ). if  $A[i] = -3$  and  $C[-3] = 1$ ,  $C[-3] = C[-3] - 1$  (then  $C[-3] = 0$ )  $B[C[-3]] = A[i]$  ( $B[0] = -3$ ), this process repeats until we have an array  $B = [3, -1, 0, 5, 8, 10]$ .

- **Time Complexity:**

Best Case:  $\Omega(n+k)$

Average Case:  $\theta(n+k)$

Worst Case:  $O(n+k)$

- **Radix Sort:**

In radix sort, we use stable sort (counting sort) for sorting purpose. In this sorting algorithm, we sort digits starting from least significant digit to most significant one.

- **Pseudo Code:**

1. Take an array  $A$  of size  $n$ .



2. Radix-sort (A, d)
3. Find Maximum element of A
4. Var = 1 and using while loop check max / var > 0 if it returns true then call counting sort algorithm.
5. And then multiply var by 10.
6. Return A

- **Code in Python:**

```
def Radix-Sort(A):
    max = np.max(A)
    place = 1
    while ((max. place)>0):
        Counting_sort (A, place)
        Place = place * 10
    Return A

A = np. Array ([110, 45, 65, 50, 65, 90, 602, 24, 2, 66])
x = Radix_Sort(A)
print(x)
```

- **Positive Point of Radix Sort:**

1. Fast sorting algorithm but still depends on the number of operations performed.
2. Not perform any comparison operations.

- **Negative Point of Radix Sort:**

1. Take more space (using more than one array for sorting using radix sort).
2. If more complex operations performed, then it is slower than the other sorting algorithm (such as merge sort).
3. Needs to be rewritten if the type of data is changed such as different technique use for digits and alphabets.

- **Dry Run using Radix Sort:**

If we have an array A = [6, 345, 11, 0, 4, 22, 987], then max = 987,  $987 / 1 = 987$  so while loop returns true and calls for counting sort function. In radix sort, some changes that were made in the counting sort is given below:

```
Def Counting_sort (A, place):
    Size = len(A)
    max = np. Max(A)
    C = np. Array ([0 for each element in range (0, 10)])
    Output = [0 for each element in range (size)]
    for i in range (0, size):
```

```

        j = (A[i] // place) % 10
        C[j] = C[j] + 1
    for i in range(0, len(C)):
        if (i == 0):
            continue
        else:
            C[i] = C[i] + C[i-1]
    for i in range(size - 1, -1, -1):
        j = (A[i] // place) % 10
        C[j] = C[j] - 1
        K = C[j]
        Output[K] = j

```

So, we have an array  $C = [0,0,0,0,0,0,0,0,0]$  (size must be 10 because in this case we have digits from 0 to 9)

$J = (6/\text{place}) \% 10$  which return 6.  $C[6] = 1$ . Similarly, when  $A[i] = 345, 11, 0, 4, 22, 9, 87, 366$  then  $C[5] = 1, C[1] = 1, C[0] = 1, C[4] = 1, C[2] = 1, C[9] = 1, C[7] = 1, C[6] = 2$ .

When each index value of  $C$  adds with the previous value of index (starting from  $C[1]$  then we have  $C = [1,2,3,3,4,5,6,7,8]$  then  $A = [0,11,22,4,345,6,366,87,9]$  then  $\text{place} * 10$  and the same process will repeat until  $\text{max}/\text{pace} > 0$ .

- **Time Complexity:**

Best Case:  $\Omega(nk)$

Average Case:  $\theta(nk)$

Worst Case:  $O(nk)$

- **Bucket Sort:**

Bucket sort is a sorting algorithm in which we use  $n$  buckets to sort elements of an array. (by distributing elements into the number of buckets which we use then sort individually).

- **Pseudo Code:**

1. Suppose we have an array  $A$ .
2. We create number of buckets according to the size of given array then we multiply size by each element of an array and create bucket's size accordingly.
3.  $\text{Bucket}[\text{int}(\text{size} * i)]. \text{append}(i)$  (**case similar to 2d array**)
4. We apply insertion sort to each bucket individually.

- **Code in Python:**

```

def bucketSort(arr):
    size = len(arr)
    bucket = []
    for i in range(size):
        bucket.append([])
    for i in arr:
        bucket[int(size*i)].append(i)

```

```
for i in range (0, size):
    bucket[i] = Insertion_sort(bucket[i])
```

- **Positive Point of Bucket Sort:**

1. Not even a single comparison performs in this sorting algorithm.
2. It is stable (also require any stable algorithm to perform sorting on buckets).
3. It is fast.

- **Negative Point of Bucket Sort:**

1. Need extra spaces as n buckets use in this sorting algorithm.

- **Time Complexity:**

Best Case:  $\Omega(n+k)$

Average Case:  $\theta(n+k)$

Worst Case:  $O(n^2)$

## Analysis of Data Scrapping

- Data scrapped from webpage using selenium and beautiful soap. Here we describe some analysis of data that scrapped in this project and issues that faced while scrapping data from website.

Website Source	Scrapping Source (Website: " <a href="https://www.efootwear.eu">Efootwear.eu</a> ")
<p>Python code of Data Scrapping (only for one webpage is given as an example)</p> <pre>import requests import re from selenium import webdriver from bs4 import BeautifulSoup import pandas as pd from selenium.webdriver.chrome.options import Options from selenium import webdriver from bs4 import BeautifulSoup from selenium.webdriver.common.action_chains import ActionChains chrome_options=Options() prefs = {"profile.managed_default_content_settings.images":2} chrome_options.add_experimental_option("prefs",prefs) driver = webdriver.Chrome(executable_path=r"C:\Users\khadija_precious\Downloads\chromedriver_win 32\chromedriver.exe",options=chrome_options) driver.get("https://www.efootwear.eu/womens-shoes.html?limit=50")  btn=driver.find_element_by_css_selector('.button:nth-child(2)') clic = ActionChains(driver).move_to_element(btn) clic.click().perform()  content = driver.page_source</pre>	

```

soup = BeautifulSoup(content)
names=[]
products_ID = []
price = []
color = []
Brand = []
SKU = []
Season = []
Type = []
sale_per = []
sale_price = []
size = []
urls=[]
total_pages = soup.find('div',attrs={'class':'toolbar-
bottom_pager'}).findChildren()[6].text
count = 1

for page in range(72,int(total_pages)):
    driver.get("https://www.efootwear.eu/womens-shoes.html?p=" + str(page) +
"&limit=200")
    main=driver.find_elements_by_css_selector('.products-list__item')
    for i in main:
        hover=ActionChains(driver).move_to_element(i)
        hover.perform()
        content = driver.page_source
        soup = BeautifulSoup(content)

        for a in soup.findAll('div',attrs={'class':'products-list__item-wrapper'}):
            url=a.find('a',attrs={'class':'products-list__link'}).get('href')
            if(url):
                name = a.find('span',attrs = {'class':'products-list__name-
first'}).text.strip()
                product_ID = a.find('span',attrs = {'class':'products-list__name-
second'}).text.strip()
                proPrice = (a.find('div' , attrs = {'class':'products-list__price-
box'})).findChildren()[0].text.strip())
                pr = re.sub(r"[\u20ac]", "", proPrice)
                sizes = a.find('div',attrs={'class': 'products-list__item-sizes'})
                link = "https:"+url
                print("link " + link)
                driver.get(link)
                content_2 = driver.page_source
                soup = BeautifulSoup(content_2)
                for b in soup.findAll('div',attrs={'class':'one-col-wrapper'}):
                    names.append(name)
                    products_ID.append(product_ID)
                    price.append(pr)
                    try:
                        li=[]
                        for i in sizes.findAll('span'):
                            li.append(i.text)

```

```

        makeitastring = ' , '.join(map(str, li))
        size.append(makeitastring)
    except:
        size.append("N/A")
    if(b.find('li' , attrs = {'class' : 'e-list-item e-product-attributes__attribute-item' , 'data-attribute': 'kolor'})):
        colorPro = b.find('li' , attrs = {'class' : 'e-list-item e-product-attributes__attribute-item' , 'data-attribute': 'kolor'}).findChildren()[1].text.strip()
        color.append(colorPro)

    else:
        colorPro = "not defined"
        color.append(colorPro)
    if(b.find('li' , attrs = {'class' : 'e-list-item e-product-attributes__attribute-item' , 'data-attribute': 'manufacturer_with_collection'})):
        brand = b.find('li' , attrs = {'class' : 'e-list-item e-product-attributes__attribute-item' , 'data-attribute': 'manufacturer_with_collection'}).findChildren()[1].text.strip()
        Brand.append(brand)
    else:
        brand = "not defined"
        Brand.append(brand)
    if( b.find('li' , attrs = {'class' : 'e-list-item e-product-attributes__attribute-item' , 'data-attribute': 'sku'})):
        sku = b.find('li' , attrs = {'class' : 'e-list-item e-product-attributes__attribute-item' , 'data-attribute': 'sku'}).findChildren()[1].text.strip()
        SKU.append(sku)
    else:
        sku = brand = "not defined"
        SKU.append(sku)
    if(b.find('li' , attrs = {'class': 'e-list-item e-product-attributes__attribute-item' , 'data-attribute': 'rodzaj_obcasa' })):
        sole = b.find('li' , attrs = {'class': 'e-list-item e-product-attributes__attribute-item' , 'data-attribute': 'rodzaj_obcasa' }).findChildren()[1].text.strip()
        t = re.sub(r"[\,\\n]", "", sole)
        m = re.sub(' +', ' ', t)
        Type.append(m)
    else:
        sole = "not defined"
        Type.append(sole)
    if(b.find('div' , attrs = {'class': 'product-left__badges'})):
        sale = b.find('div' , attrs = {'class': 'product-left__badges'}).findChildren()[0].text.strip()
        sale_per.append(sale)
    else:
        sale = "-0%"
        sale_per.append(sale)
    if(b.find('div' , attrs = {'class': 'e-product-price__special'})):
        salePrice = b.find('div' , attrs = {'class': 'e-product-price__special'}).text.strip()

```

```

        Spr = re.sub(r"[€]", "", salePrice)
        sale_price.append(Spr)
    else:
        salePrice = "product not in sale"
        sale_price.append(salePrice)

    if(b.find('li' , attrs = {'class' : 'e-list-item e-product-attributes__attribute-item' , 'data-attribute': 'sezon'})):
        season = b.find('li' , attrs = {'class' : 'e-list-item e-product-attributes__attribute-item' , 'data-attribute': 'sezon'}).findChildren()[1].text.strip()
        Season.append(season)
    else:
        season = "not found"
        Season.append(season)
    try:
        df = pd.DataFrame({'Product Name':names, 'Product ID':products_ID, 'Price (in Euro)':price, 'Color':color, 'Brand':Brand , 'SKU':SKU , 'Sole-Type':Type , 'Sale Percentage': sale_per , 'Sale Price' :sale_price , 'Season': Season , 'Size': size})
        df.to_csv('DJ_4.csv', index=False, encoding='utf-8')
        print(count)
        count+= 1
    except:
        print(f'url = {urls}')
    else:
        continue

driver.close()

```

## • Issues Faced during Data Scrapping:

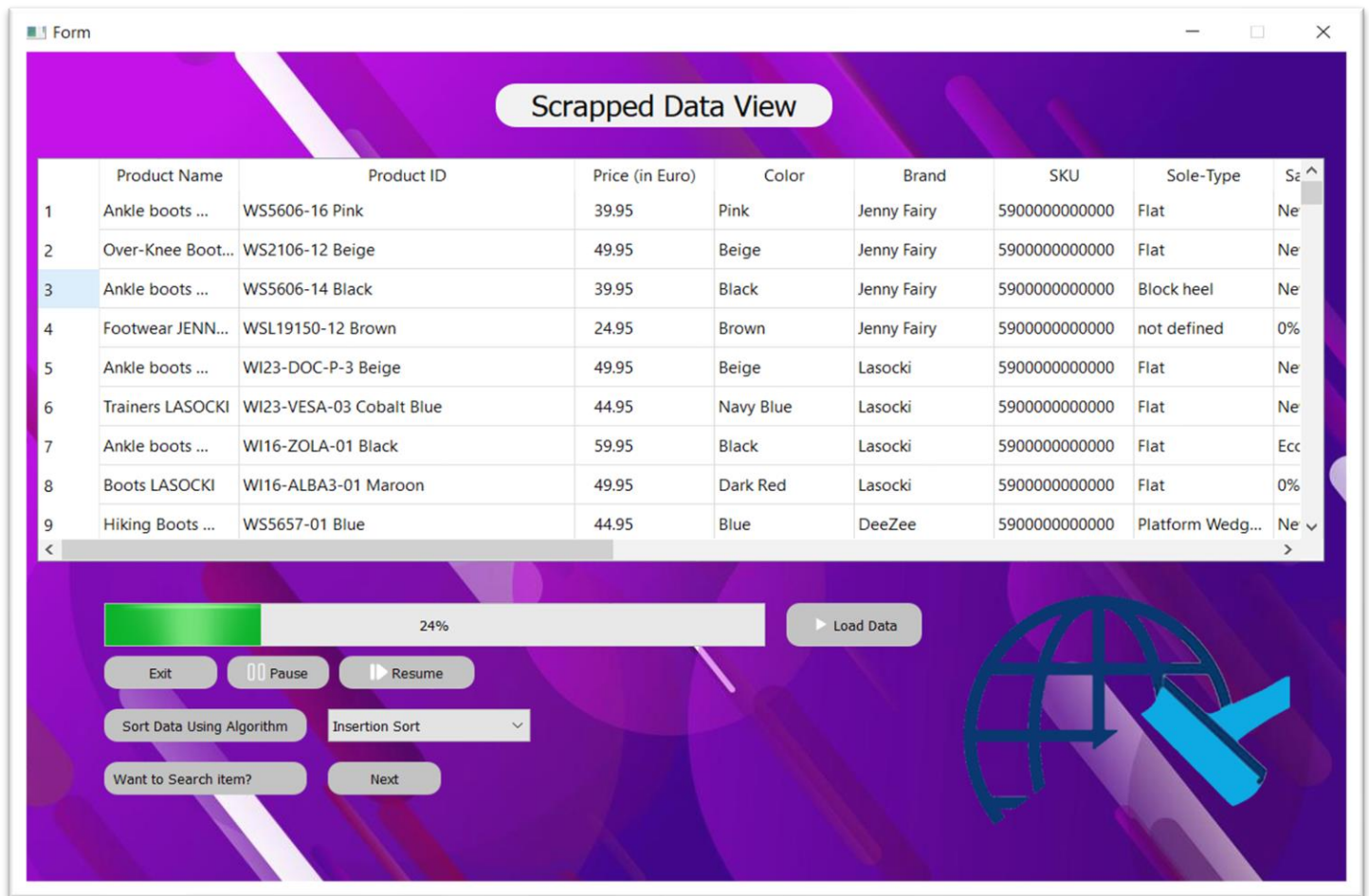
As we mentioned above, data scrapping done in this project using selenium and beautiful soap. Here we'll discuss some problems that we faced during this process.

Issues	How did you resolve these issues?
Getting URL of the page using selenium web driver	While getting URL of different pages during web Scrapping there were a lot of problems that occurred such as “ <b>404 error</b> ”. While scrapping if there are problems like internet issues or some other issue while getting URL from web driver, we resolve them by import request or using conditions (if the condition returns true then we successfully get URL of webpage but if it returns false then instead of stop scrapping or return some exception it'll continuously load another web pages and ignore the previous one.

Getting null pointer exception	While getting different attributes from webpage we often receive a null pointer exception if there exist such attribute or object that have no value or if it doesn't even exist so, we again handle it by using if else conditions. So, if there exist such attribute with some specific value then we successfully append it into the list and if the webpage doesn't have that attribute or have attribute but with none value then handle it in else part.
Arrays must be the same length	If we are facing the above-mentioned problem then also facing this type of error and we can resolve them by using the above-mentioned technique.
Module not found	By just installing or importing the module or package.

- **Issues Faced in Using PyQt5 (GUI Implementation):**

Issues	How did you resolve these issues?
Qrc file (for style sheet)	Applied some properties on buttons, labels or set background properties using qrc file. An error occurred while importing qrc file in GUI with .py extension file. So, we resolve this by converting qrc file into py and then import it in GUI file by using Command in cmd <b>pyrcc5 file_name.qrc -o finew_name.py</b>
Unable to get UI file in py file	Simply run command <b>pyuic5 -x file_name.ui -o new_filename.py</b> in cmd by opening it in the same path of ui file
Error in loading CSV file into Qtwidget Table	An error occurred "UnicodeDecodeError: 'utf-8' codec can't decode byte 0xff in position 155: invalid start byte" while loading data from csv file so we simply resolved this by-passing encoding= 'unicode_escape' in function panda.read_csv (). So, after that table with loaded data given below as an example:



### Issues Faced during implementation of Sorting Algorithm:

Issues	How did you resolve these issues?
Key error while Loading Data from file for sorting	An error Key error occurred while load data from CSV file before sorting and the reason is that the column name of CSV file having some spaces so we just set it according to the CSV file.
Problem in algorithm while sorting	We simply resolved them by changing some steps of our code. The snapshot of GUI screen is given below as a example of sorting data.



## Scrapped Data View

	Product Name	Product ID	Price	Color	Brand	SKU	Sole-Type	Sale P
1	Ankle boots ...	WS5606-16 Pink	39	Pink	Jenny Fairy	5,900,000,000,000	Flat	New
2	Over-Knee Boot...	WS2106-12 Beige	50	Beige	Jenny Fairy	5,900,000,000,000	Flat	New
3	Ankle boots ...	WS5606-14 Black	40	Black	Jenny Fairy	5,900,000,000,000	Block heel	New
4	Footwear JENN...	WSL19150-12 Brown	25	Brown	Jenny Fairy	5,900,000,000,000	not defined	0
5	Ankle boots ...	WI23-DOC-P-3 Beige	50	Beige	Lasocki	5,900,000,000,000	Flat	New
6	Trainers LASOCKI	WI23-VESA-03 Cobalt Blue	45	Navy Blue	Lasocki	5,900,000,000,000	Flat	New
7	Ankle boots ...	WI16-ZOLA-01 Black	60	Black	Lasocki	5,900,000,000,000	Flat	Eco frie
8	Boots LASOCKI	WI16-ALBA3-01 Maroon	50	Dark Red	Lasocki	5,900,000,000,000	Flat	0
9	Hiking Boots ...	WS5657-01 Blue	45	Blue	DeeZee	5,900,000,000,000	Platform Wedg...	New

24%

Start Loading Scrapped Data

Exit Pause Resume

Insertion Sort ☐ Name ☐ Price ☐ Product ID Sort Data Using Algorithm

Want to Search item? Next

Before Sorting

## Scrapped Data View

	Product Name	Product ID	Price	Color	Brand	SKU	Sole-Type	Sale P
1	Ankle boots ...	WS2702-09 Black	35	Black	Clara Barson	5,900,000,000,000	Flat	0
2	Ankle boots ...	WYL1877-5 Bluemarin	30	Navy Blue	Clara Barson	5,900,000,000,000	Wedge heel	0
3	Ankle boots ...	WS5200-10 Brown	35	Brown	Clara Barson	5,900,000,000,000	Flat	0
4	Ankle boots ...	WS19289-02 Black	40	Black	DeeZee	5,900,000,000,000	Block heel	0
5	Ankle boots ...	V802-101 Beige	90	Beige	Gino Rossi	5,900,000,000,000	Flat	New
6	Ankle boots ...	RST-GALLIE-02 Black	85	Black	Gino Rossi	5,900,000,000,000	Block heel ...	New
7	Ankle boots GO...	WI23-REBECA-06 Red	50	Red	Go Soft	5,900,000,000,000	Flat	New
8	Ankle boots GO...	WI16-SAMSON-02 Black	50	Black	Go Soft	5,900,000,000,000	not defined	0
9	Ankle boots ...	WS5606-16 Pink	39	Pink	Jenny Fairy	5,900,000,000,000	Flat	New

98%

Start Loading Scrapped Data

Exit Pause Resume

Insertion Sort ☒ Name ☐ Price ☐ Product ID Sort Data Using Algorithm

Want to Search item? Next

After Sorting

As we given below this sorting has been done according to specific column name and using Insertion sort as a Sorting Algorithm.

- **Improvements Require in the Project:**

1. Data Scrapping must contain more than 1 million products or entities. But this project contains only 21000 to 25000 data or products.
2. There should be many improvements in GUI integration according to the requirement.
3. should provide better option for sorting data.
4. Multi-threading should be used to make the project more efficient to perform multi-tasking so that data should be load into the table and perform sorting while scrapping data.

- **Things that we learn from this project:**

1. Learn how to handle different cases to get Data and URL from different Web pages.
2. Learn Python GUI toolkit PyQt5. Also learn about qrc file and how to convert UI and qrc file into py file.
3. Qtwidget and how to work with Qtwidget table. (load data into the widget Tables).
4. Have understanding of different python Libraries and many other things.