

JOB SEARCH PORTAL

CS162- Final Project Requirement Document



Project Supervisor

Mr. Samyan Qayyum Wahla

Project Members (G15)

Ahmad Shoaib

2020-CS-130

Muhammad Ehsaan

2020-CS-128

DECLARATION

We, hereby declare that the Project entitled “Job Search Portal” being submitted by us towards to partial evaluation of the course “Data Structures and Algorithms” is a project carried by us under the supervision of Mr. Samyan Qayyum Wahla , and have not been submitted somewhere else.

We will solely be responsible if any kind of plagiarism is found.

ACKNOWLEDGMENT

We would like to share our gratitude to all those who has help us in the completion of the entire project. During the implementation of the project we faced many challenges due to our lack of knowledge and experience but our Project Helpers had help us to get over all the difficulties and in final compilation of our idea to a shaped sculpture.

We would like to thank Mr. Samyan Qayyum Wahla for his guidance and governance, because of which we were able to learn the minute aspects of our project work.

ABSTRACT

The brain blowing proportion of data on the Web is a rich resource for any field of assessment or individual premium. We have sifted the data and has chosen Job Information as a resource for our undertaking. We are expected to effectively gather that data by using a robotized cycle and make Job Search Portal.

Job data is fundamental for getting what employees and employers value. Job sites like Glassdoor, Rozee.pk, Indeed.pk and Zip recruiter have job listings full of useful salary data, job descriptions, employee reviews, and more. Learning to extract and analyses this data helps your organization stay competitive and find the best talent. Web scraping, the automatic extraction of data from a web page, makes it possible to extract job data from websites for analysis. Extracting this data provides organizational insights that help you develop a competitive salary range, understand employee desires and values, and stay updated on changes in the job market. Once this data is collected, it can be directly inputted into your preferred analysis program.

Table of Contents

DECLARATION	i
ACKNOWLEDGMENT	ii
ABSTRACT.....	iii
Chapter 1.....	1
Introduction.....	1
1.1 Executive Summary	6
1.2 Business Case.....	6
1.2.1. Business Need	6
1.2.2. End Users	6
1.2.1.1. Particular Industry Companies	6
1.2.1.2. Online Job Boards and Aggregators	6
1.2.1.3. HR Agencies	6
1.3 Motivation	6
1.4 Impacts and Implications	6
1.4.1. Impacts if project proceeds	6
1.4.2. Implications if project not proceeds.....	6
Chapter 2.....	1
Technical Details.....	1
2.1 Details	6
2.1.1. Names of Entity.....	6
2.1.1. Attributes of Entity	6
2.2 Sample of Scrapping Source	6
2.2.1. Git-hub Repository Link	6
2.3 Sorting Algorithms	6
2.3.1. Bubble Sort	6
2.3.2. Insertion Sort.....	6
2.3.3. Merge Sort.....	6
2.3.4. Selection Sort	6
2.3.5. Quick Sort	6
2.3.6. Heap Sort	6
2.3.7. Radix Sort	6
2.3.8. Bucket Sort.....	6
2.3.9. Comb Sort	6

2.3.10. Counting Sort	6
2.3.11. Brick Sort	6
2.3.12. Tree Sort.....	6
2.3.13. Shell Sort.....	6
2.4 Searching Algorithms	6
2.4.1. Linear Search.....	6
2.4.2. Binary Search	6
2.4.3. Jump Search	6
2.4.4. Interpolation Search.....	6
2.5 Searching Filters	6
2.6 Multi-level Sorting.....	6
2.6.1. Alphabetically	6
2.6.2. Reviews	6
2.6.3. Alphabetical Parallel Sorting	6
2.7 Any other feature	6
2.8 Interface	6
2.8.1. Pencil Tool	6
2.8.1. Details	6
Chapter 3.....	1
Sorting Algorithms	1
3.1 Iterative Sorting Algorithms	6
3.1.1. Insertion Sort.....	6
3.1.2. Selection Sort	6
3.1.2. Bubble Sort	6
3.2 Recursive Sorting Algorithms	6
3.2.1. Merge Sort.....	6
3.2.2. Shell Sort.....	6
3.2.3. Quick Sort	6
3.2.3. Recursive Bubble Sort	6
3.3 Linear Sorting Algorithms	6
3.3.1. Brick Sort	6
3.3.12. Counting Sort	6
3.4 Binary Search Tree Sorting Algorithms	6
3.4.1. Tree Sort.....	6
3.4.2. Heap Sort	6

3.4.3. Comb Sort	6
Chapter 4.....	1
Accomplishment and Integration	1
4.1 Integration.....	6
4.1.1. Source of Scrapping.....	6
4.1.2. Difficulties Faced	6
4.1.2.1 Against scratching techniques that block Job Scrapping	6
4.1.2.2 Shortlisted Job Scrapping Websites.....	6
4.1.3. Ideal Source for Project	6
4.2 Data as an asset	6
4.2.1. Requirements.....	6
4.2.2. CSV	6
4.3 UI Implementation	6
4.3.1. QT Designer	6
4.3.2. PyQt5.....	6
4.3.3. Ideas for UI	6
4.3.4. Collaboration.....	6
4.4 Pace Towards Code Execution.....	6
4.4.1. UI to Py	6
4.4.2. QTable Widget	6
4.4.3. Mouse Events	6
4.4.4. Sorting Ascending/Descending	6
4.4.5. Multi-level Sorting	6
4.5 Unfinished Ideas.....	6
4.5.1. Custom Sorting Algorithm.....	6
4.2.1. Dynamic Scrapping	6
Chapter 5.....	1
User Manual.....	1
5.1 How to Commence scrapping	6
5.2 How to Pause/Stop Scrapping	6

List of Tables

Table 1. Attributes of entity	i
Table 2. Detail of scrapping website source.....	2
Table 3. Detail of type of UI Component	2
Table 4. Detail analysis of insertion sort.....	2
Table 5. Detail analysis of Selection Sort	2
Table 6. Detail analysis of Bubble Sort	2
Table 7. Detail analysis of Merge Sort	2
Table 8. Detail analysis of Shell Sort	2
Table 9. Detail analysis of Quick Sort	2
Table 10. Detail analysis of Recursive Bubble Sort	2
Table 11. Detail analysis of Brick Sort	2
Table 12. Detail analysis of Counting Sort	2
Table 13. Detail analysis of Tree Sort	2
Table 14. Detail analysis of Heap Sort	2
Table 15. Detail analysis of Comb Sort	2
Table 16. Detail analysis of Ideal Source.....	2

Table of Figures

Figure 1. Snapshots of websites ZipRecruiter and Indeed.pk	i
Figure 2. Pencil Tool UI	i
Figure 3. Snapshot of website Indeed.pk	i
Figure 4. Snapshot of website Glassdoor.....	i
Figure 5. Snapshot of website SimplyHired	i
Figure 6. Snapshot of website TimesJob.com.....	i
Figure 7. Snapshot of website ZipRecruiter.....	i
Figure 8. Snapshots of CSV	i
Figure 9. Snapshots of Mock Ui.....	i
Figure 10. Snapshots of Final UI Look	i
Figure 11. Snapshots of QRC File.....	i
Figure 1.2 Snapshots of QtWidgets	i
Figure 13. Snapshots of QtTableWidget.....	i
Figure 14. Snapshots of QtProgressBar	i
Figure 15. Command Prompt of Anaconda	i
Figure 16. Scrapped data into Qt Table widget.....	i
Figure 17. Snapshots of Mouse Event in UI	i
Figure 18. Taking algorithm as input from combo box	i
Figure 19. Outputting time taken by a certain algorithm.....	i
Figure 20. Sorting each algorithm in ascending and descending order	i
Figure 21. Adding custom algorithm.....	i
Figure 22. First screen when application gets launched	i
Figure 23. Commence Scrapping	i
Figure 24. Data plotted in fields	i
Figure 25. How to pause/stop scrapping.....	i
Figure 26. Selection algorithm from combo box.....	i
Figure 27. List of algorithms in combo box.....	i
Figure 28. Snapshots of sorted data.....	i
Figure 29. Sorting your desired column	i
Figure 30. Outputting time taken by a certain algorithm.....	i
Figure 31. Selection of multi-level sorting	i
Figure 32. List of multi-level sorting algorithms in combo box	i

Introduction

1.1. Executive Summary

The mind blowing measure of information on the Web is a rich asset for any field of examination or individual premium. We have filtered the information and has selected Job Information as an asset for our project. We are intended to successfully collect that information by utilizing a robotized cycle and make Job Search Portal.

Job data is fundamental for getting what employees and employers value. Job sites like Glassdoor, Rozee.pk, Indeed.pk and Zip recruiter have job listings full of useful salary data, job descriptions, employee reviews, and more.

Figuring out how to separate and investigate this information helps your association stay serious and track down the best ability. Web scratching, the programmed extraction of information from a site page, makes it conceivable to separate occupation information from sites for investigation. Once this data is collected, it can be directly inputted into your preferred analysis program. Most importantly, how about we start by expressing the target of the task: to make some Job Board with data on propositions for employment that show up on the Web. For the contention, suppose I'm intrigued on "Information Investigation" propositions for employment in Japan and I need to get the information out from two diverse realized places of work.

What sort of information would we like to acquire? We should keep it straightforward: Title, URL, Update date, Location, Salary, and Type of job, Experience and Skills requirement. Subsequent to separating the information we are keen on, suppose the "Title" for instance, we need some place to store it: void records.

We will be storing each data in different empty lists and, once all the extracting is done, they will be used in order to create a Data Frame where the final Job Board will be visualized and we will use PyQt5 to display it with user friendly interface. One would be able to sort each column having a unique entity according to their desired algorithms. Moreover, one could be able to differentiate which sorting algorithm takes how much time. Advanced filters for string columns will also be implemented. One will have the option to sort using multiple columns.

Hence, if we do consider the overall purpose of the project or how this project could benefit users? The answer to the vary question would not be wrong saying that it would help agencies or companies on demand to expedite their hiring process and afterwards they could easily be able to keep a check on what post or titled job is been on trend, or what kind of employees they should and what their basic requirements could. They could be able to apply sorting paradigms on their data to experience fruitful results. Online job aggregators could also benefit themselves from such scrapping tools that could provide them with analysis they desire for.

1.2. Business Case

The internet based occupation market has without a doubt abrogated face to face employing exercises. This is particularly obvious when most urban areas all throughout the planet face rounds of lock-down and more positions shift to a distant mode since the 2020 Coronavirus episode. In this sense, web scratching position postings serve establishments and associations as well as individual occupation searchers. So, if we do talk about the business case of Job Scrapping Tool it would not be wrong saying that many companies would need such scrapping tools to expedite their finance, hiring and accommodation processes.

1.2.1. Business Need

Job competitors need to follow positions using conventional strategies. Managers need to publicize the opportunities and sort all candidate subtleties, lead choice techniques and complete the customs. This approach is dreary and requires a ton of effort. There is need of a work entrance where competitors successfully secure the positions and director can find sensible opportunities for the work. It is the place where web rejecting becomes advantageous. Utilizing a web scrapping tool saves your association extraction time, hence giving you more opportunity to devote to investigation and producing helpful bits of knowledge.

1.2.2. End Users

Using data extraction and web gathering, position data scratching used by a huge load of adventures and associations. We should discuss the top customers that land benefitted from work information rejecting.

1.2.2.1 Particular Industry Companies

Specific industry organizations are potential occupation information scratching clients. They at every step of their entire business process need such kinds of scrapping. In the past era, they were not provided with such tools and undoubtedly it was the main reason their process of flow was slow. Though it would also be teary dealing with such a large amount of data. It is especially valid for organizations which are looking for fast augmentation since they need to get great recruits rapidly. Job Data Scrapping has a vital influence of your enlistment interaction. Special steps and advancements has been added to make the methodology less tedious, work information scratching administrations will get done with a task for you. This will offer you with new information from across the web, consequently you can concentrate on it and invest energy on more huge business perspectives. Utilizing a web scrapping tool saves your association extraction time, hence giving you more opportunity to devote to investigation.

1.2.2.2 Online Job Boards and Job Aggregators

Occupation sheets are diverse web-based occupation stages, which offer two vital types of assistance – one for the work searchers just as for the businesses. Occupation searchers might put their CVs on various occupation board destinations and quest for available internet based bids for employment. Businesses might transfer the work opening for making it observable on a site and audit the CVs of the multitude of up-and-comers. Notwithstanding position postings, the work sheets gather information about representative profiles, organization profiles, just as sets of responsibilities.

Occupation web scratching administrations can offer you the most up to date information to dissect just as post on the site to make your substance more important, helpful, and state-of-the-art for both occupation searchers and businesses.

1.2.2.3 HR Agencies

Job information scrapping has a tremendous imminent in the HR business. You can without much of a stretch use it for a HR organization like some other business. In any case, as HR organizations work particularly with HR enrollment and the board, it turns out to be more imperative to consistently get ideal and legitimate information.

HR organizations might utilize work information scratching for gathering position postings which are both area explicit and industry-explicit. Different organizations will counsel your HR office to satisfy their representatives' enrollment. Your key occupation should be centered on staffing, directing meetings, just as work force situations. Occupation information scratching will give you reasonable new occupation information to assist you with focusing on the vital assignments without investing energy in the web looking for available positions.

1.3. Motivation

The Corona virus pandemic has set off one of the most exceedingly awful positions emergencies since the Economic crisis of the early 20s. There is a genuine peril that the emergency will expand neediness and extend disparities, with the effect felt for quite a long time to come. The current circumstance expects applicants to look through print and visual media for job opportunities.

Candidates need to follow positions using customary strategies and appear for meet on a foreordained date at demonstrated region. Employers need to publicize the opportunities and sort all candidate subtleties, direct choice techniques and complete the customs. This

methodology is monotonous and requires a lot of exertion and assets. There is need of a work entryway where candidates effectively secure the positions and manager can discover reasonable possibility for the work. It is where web scrapping becomes beneficial. Candidates need to go after positions utilizing traditional techniques and show up for meet on a predetermined date at indicated area. Businesses need to promote the opportunities and sort all candidate subtleties, lead choice strategies and complete the customs. This methodology is monotonous and requires a lot of exertion and assets.

There is need of a work entryway where candidates effectively secure the positions and manager can discover reasonable possibility for the work. It is where web scrapping becomes beneficial. We are intended to make a Job Scraper Tool a sit will take all the pain of extracting, monitoring, crawling, and refining data from the job posting websites and provide this data in full sorted form to be used by companies.

1.4. Impacts and Implications

Here, we will discuss about impacts or effects of the project if it gets complete or due to some reason it gets rebuked, how it would affect society, how it would affect business need and many such factors.

1.4.1 Impacts if project proceeds

Many organization recruitment strategy plan depends on such scrapping tools. Data scraping is amongst services like companies provide. This service could be extremely efficient and value for different companies

1.4.2 Implications if project not proceeds

As discussed earlier, many organizations HR Agencies, recruitment strategy plans depend on scraper tools, though it would not create such hazardous effect, but may be in the near future if we get out project complete, fast and accurate, many companies will head to it. In case it doesn't proceeds, such companies would demise no further trust.

Technical Details

2.1 Details

2.1.1 Names of entity

Job postings on career sites usually consist of

- Job title
- Salary
- Location
- Time commitment (part-time/full-time)
- Job description
- Number of reviews/rating
- Company Name

2.1.2 Attributes of Entity

Name	Data Type	Description
Job Title	String	It will contain information about one's job position.
Salary	Int	It will exhibit a fixed amount of money or compensation that would be paid to an employee by an employer in return for work performed
Location	String	It will tell one about the workplace where the job is to be performed.

Time Commitment	String	It will state the percent time or the number of hours per week to be applied toward the work of project or activity.
Job Description	String	It will list the main features of a specific job.
Reviews/Rating	String	It will declare a formal assessment of job who have already gone through the same intermediate stop.
Company Name	String	It will entertain the user with the name by which the company is recognized globally.

Table 1. Attributes of Entities

2.2 Sample of Scrapping Source

For reference, snapshots of websites from where we will be going to scrap data has been attached. These are websites that are enriched with vast amount of data for jobs. Although a single website would be enough to scrap data up to our requirements but in order to face redundancies we have consider two of them.

Figure 1. Snapshots of websites ZipRecruiter and Indeed.pk

new

IT - Intern

Sybrid (Pvt) Ltd 3.8 ★

Lahore

Rs 10,000 - Rs 15,000 a month

➤ Easily apply ⚡ Responsive employer

- IT Intern reports to Manager IT, he is responsible for 1st Level Support for Systems related issues, Maintaining Systems & Security Company-wide.

Just posted · More...

- View all [Sybrid \(Pvt\) Ltd jobs - Lahore jobs](#)
- Salary Search: [IT - Intern salaries in Lahore](#)
- See popular [questions & answers about Sybrid \(Pvt\) Ltd](#)

201,425 Full Time Jobs Near Me

Remote Call Center Representative (Full-Time or Part-Time)

BJC Medical Group Chicago, IL

Type Full-Time

Job ID: 1222815 Employment Status: Full-Time More Information BJC Medical Group is a dynamic, multidisciplinary group with locations across the St. Louis metropolitan and outlying areas. The ...

Medical Assistant - Pediatrics, Full-time, Days ^{NEW!}

Lake Forest Hospital Foundation Chicago, IL

Type Full-Time

Certified Patient Care GiverEmployment Type: Full-time




Snapshots	Detail
 IT - Intern	Job Title
Sybrid (Pvt) Ltd 3.8 ★	Company Name
Lahore	Location
Rs 10,000 - Rs 15,000 a month	Salary
<ul style="list-style-type: none"> IT Intern reports to Manager IT, he is responsible for 1st Level Support for Systems related issues, Maintaining Systems & Security Company-wide. 	Job Description
3.8 ★	Ratings
 Easily apply  Responsive employer	About

Table 2. Detail of Scrapping Website Source

2.2.1 Git-hub Repository Link

The link of Git-hub Repository of our project has been attached hereby:

<https://github.com/mehsaandev/CS261F21PID15>

2.3 Sorting Algorithms

In math and software engineering, an algorithm is a limited grouping of obvious directions, commonly used to take care of a class of explicit issues or to play out a calculation. Calculations are utilized as particulars for performing computations, information handling, robotized thinking, and mechanized dynamic and different errands. Conversely, a heuristic is

a strategy utilized in critical thinking that utilizes reasonable strategies or potentially different assessments to create arrangements that may not be ideal yet are adequate given the conditions.

Short description of sorting algorithms studied so far has been listed below.

2.3.1 Bubble Sort

Bubble sort is an arranging calculation that works by over and over again venturing through lists that should be arranged, looking at each pair of contiguous items and sorting them until no further swaps are possible.

2.3.2 Insertion Sort

Here, sorted array is built having one item at a time. The array elements are compared with each other sequentially and then arranged simultaneously in increasing or decreasing order.

2.3.3 Merge Sort

It is a sorting algorithm that divides the list into two groups, recursively sorts each group, and merges them into a final but sorted sequence

2.3.4 Selection Sort

This sorting algorithm simply finds the minimum or maximum element in an unsorted array and then puts it in its correct position in a sorted array.

2.3.5 Quick Sort

It just separate the "big" elements and "small" elements repeatedly. The first step of the algorithm requires choosing a "pivot" value that will be used to divide big and small numbers.

2.3.6 Heap Sort

Heap sort is a comparison-based sorting technique based on Binary Heap data structure. It is similar to selection sort where we first find the minimum element and place the minimum element at the beginning. We repeat the same process for the remaining elements. The heap can be represented by a binary tree or array

2.3.7 Radix Sort

Radix sort is one of the sorting algorithms used to sort a list of integer numbers in order. In the radix sort algorithm, a list of integer numbers will be sorted based on the digits of individual

2.3.8 Bucket Sort

Bucket sort, or bin sort, is a sorting algorithm that works by distributing the elements of an array into a number of buckets. Each bucket is then sorted individually, either using a different sorting algorithm, or by recursively applying the bucket sorting algorithm.

2.3.9 Comb Sort

Comb Sort is an advanced version of Bubble Sort as the latter always compares all values to be removed one by one. Comb Sort improves on Bubble Sort by using a gap of size more than 1.

2.3.10 Counting Sort

Counting sort is an arranging strategy on keys. It works by counting quantity of objects having unmistakable key qualities. Then, it does some calculations for right position of each object in the output.

2.3.11 Brick Sort

This algorithm is divided into 2 phases- Odd and Even. In the odd phase, we perform a bubble sort on odd indexed elements and in the even phase, we perform a bubble sort on even indexed elements.

2.3.12 Tree Sort

Tree sort is a sorting algorithm that is based on Binary Search. It first creates a binary search tree from the elements of the input list or array and then performs an in-order traversal.

2.3.13 Shell Sort

The idea of shell Sort is to allow exchange of far items. In shell Sort, we make the array h-sorted for a large value of h. We keep reducing the value of h until it becomes 1.

2.4 Searching Algorithms

In software engineering, a searching algorithm, is a calculation (regularly including a large number of other, more explicit calculations) which tackles an inquiry issue. Search calculations work to recover data put away inside certain information structure, or determined in the hunt space of an issue area, either with discrete or consistent qualities.

Short description of searching algorithms that we are going to use in our project has been listed below:

2.4.1 Linear Search

A linear search or sequential search is a method for finding an element within a list. It sequentially checks each element of the list until a match is found or the whole list has been searched.

2.4.2 Binary Search

Binary search is a fast search algorithm with run-time complexity of $O(\log n)$. This search algorithm works on the principle of divide and conquers. ... Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of the item is returned.

2.4.3 Jump Search

Jump Search is a searching algorithm for sorted arrays. The basic idea is to check fewer elements (than linear search) by jumping ahead by fixed steps or skipping some elements in place of searching all elements.

2.4.4 Interpolation Search

Search a sorted array by estimating the next position to check based on a linear interpolation of the search key and the values at the ends of the search interval

2.5 Searching Filters

Searching filters that we are going to be entertained through for the following data types has been listed below:

String Data Type

- a) Search by Title**
 - i) Contains
 - ii) Starts with
 - iii) Ends with
 - iv) Equals to

- b) Search by Location**
 - i) Contains
 - ii) Starts with
 - iii) Ends with
 - iv) Equals to
- c) Search by time commitment**
 - i) Contains
 - ii) Starts with
 - iii) Ends with
 - iv) Equals to
- d) Search by Company Name**
 - i) Contains
 - ii) Starts with
 - iii) Ends with
 - iv) Equals to

Integer Data Type

- e) Contains**
- f) Equals to**

2.6 Multi-level Sorting

2.6.1 Alphabetically

We will check each character of the string and sort them by ascending order.

2.6.2 Reviews

On reviews bases we perform sorting as more reviews more salary is assigned and on reviews basis salary is sorted and on salary bases the job title is sorted so total 3 columns are sorted reviews salary and job title.

2.6.3 Parallel Alphabetical Sorting

Here, we will entertain our user through a specific kind of multilevel sorting. It will help one's using that scrapping tool with multilevel ascending order of alphabets. It will sort three columns Job Title, Location and Company Names e.g. you want to

sort and fetch all the Job Titles starting with letter B it will firstly display all the Job Titles starting with B, parallel will sort Locations with B and vice versa for Company Names. It will help organization for better job analysis.

2.7 Any other features

We will try to make an additional feature in which we can add a custom sorting Algorithm at runtime. Users will have the option to add a new algorithm and by giving the function name and definition of code, it will add that function in the application and we can use it the next time we will launch the application.

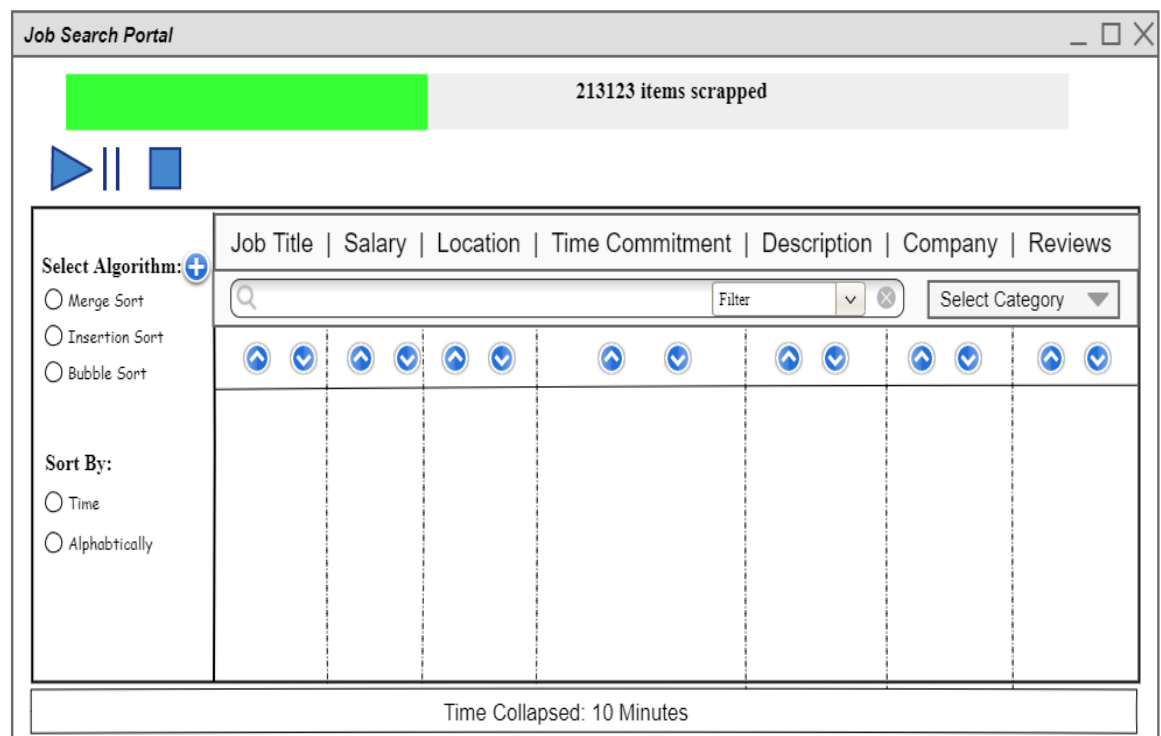
2.8 Interface

The graphical UI is a type of UI that permits clients to cooperate with electronic gadgets through graphical symbols and sound marker like essential documentation, rather than message based UIs, composed order names or message route. GUIs were acquainted in response with the apparent steep expectation to absorb information of order line interfaces (CLIs), which expect orders to be composed on a PC console.

2.8.1 Pencil Tool

The UI interface for our project has been attached hereby:

Figure 2. Pencil Tool UI



2.8.2 Details

UI Component Name	Type of UI component	Purpose of UI Component/Other details
Progress bar	Output UI Component	It will tell the number of items scrapped till now
Radio Buttons	Input UI Component	It will be used in Sort algorithm selection and sort filter for all tables sorting
Play/Pause/Stop Button	Input UI Component	It is used to pause, play and stop the scrapping process
Search bar	Input UI Component	It is used to search the data according to categories and filters
Up/Down Arrow button	Input UI Component	It will be used to sort the algorithm according to selected Filter
Combo Box	Input UI Component	It is used to select the Category and filters for search.
Add Button	Input UI Component	It is used to show the add new Function Page
Label	Output UI Component	It is used to show the time consumed by scrapping process

Table 3. Detail of type of UI component

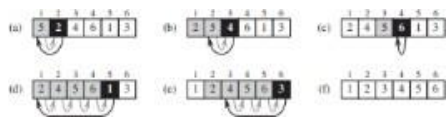
Sorting Algorithms

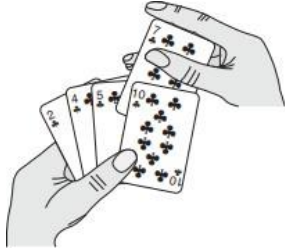
In software engineering, a sorting algorithm is a calculation that places components of a rundown into a request. The most as often as possible utilized requests are mathematical request and lexicographical request, and either rising or plummeting. Proficient arranging is significant for enhancing the effectiveness of different calculations (like inquiry and consolidation calculations) that require input information to be in arranged records. Arranging is additionally regularly helpful for ripping apart information and for creating comprehensible yield.

A detailed discussion of Sorting Algorithms through us will sort our data of job have been listed below one by one:

3.1 Iterative Sorting Algorithms

3.1.1 Insertion Sort

<i>Detailed Analysis</i>	
Description	<p>It is a sorting algorithm in which the whole array is checked out to be sorted array. In this sorting algorithm, we assume array to be divided in two steps; one assumed to be sorted one while the other is supposed to be unsorted one from which elements are being picked and find their appropriate positions in the first part i.e., sorted array. At the very initial point we divide the array with the perspective that a single element in an array would always be sorted, though if it is unsorted when combined with others. When elements get itself inserted in the sorted array, now elements are transferred one at a time to the right position of the array thus giving you with an output of sorted array.</p>  <p>The insertion sort calculation is the sort unconsciously utilized by most players when arranging the cards in their grasp. When holding a hand of cards, players will regularly examine their cards from left to right, searching for the primary card that is awkward. In order to make the concept easier to understand you can think the entire scenario as a game of playing cards. In it, if cards are provided you in an unsorted manner, what will you do?</p> <p>You'll simply start arranging them by picking cards from the unsorted one and placing them in appropriate positions at the sorted side. For</p>

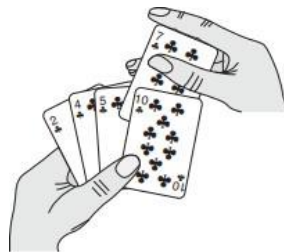
	<p>example, if the player has 3 cards with numbers 6, 7, 3 he would be satisfied with the order of 6, 7 while will hesitate for 3, he'll simple move 6 and 7 one position to right and will insert 3 at the appropriate place such as 3, 6, 7. Same works for the insertion sort.</p>  <p>All the more officially, insertion sort takes each component in turn from an unsorted set and embeds it into an arranged one by checking the arrangement of arranged components to figure out where the new component should be. It is also one of the simple sorting algorithms to be considered.</p>
Pseudo Code	<p>Insertion Sort(Array)</p> <ol style="list-style-type: none"> 1. for j=2 to Array.length 2. key= A[j] 3. i = j-1 4. while i>0 and A[i] > key 5. 5. A[i+1] = A[i] 6. i=i-1 7. A[i+1] = key
Python Code	<pre>def InsertionSort(arr): size= len(arr) for i in range(1, size): key = arr[i] j = i-1 while j >=0 and key < arr[j] : arr[j+1] = arr[j] j =j- 1 arr[j+1] = key</pre>
Time Complexity	Total Running Time

	$(T(n)) = C1 * n + (C2 + C3) * (n - 1) + C4 * \sum^{n-1}_{j=1} 1 = 1(tj) + (C5 + C6) *$ $\sum^{n-1}_{j=1} 1 = 1(tj) + C8 * (n - 1)$ <p>Running Time for Worst case (Array is fully un-sorted) $T(n) = O(n^2)$</p> <p>Running Time for Good case (Array is nearly sorted) For Best Case i.e., $t_j = 1$ After Substitution, $T(n) = C1 * n + (C2 + C3) * (n - 1) + C4 * (n - 1) + (C5 + C6) * (n - 2) + C8 * (n - 1)$ $T(n) = O(n)$</p>
Proof of Correctness	<p>Correctness can be proved through three steps:</p> <ol style="list-style-type: none"> 1. Initialization Actually, loop invariant always holds for the first element before the actual loop iteration gets started. As can be seen from pseudo code, before the first iteration $j=2$ At this point, the sorted array would have only one element $A[1]$ and if an array has one element then, it is assumed to be sorted. So, loop invariant holds prior to the first iteration of the loop. 2. Maintenance Now, I'll try to show that each element of the array maintains the loop invariant. As discussed earlier, the body of the for loop works in a way that it moves element to right side and insert the desired element at the appropriate place i.e., $A[j-1]$, $A[j-2]$ and so on. The sub-array then consists of all the sorted elements. We increment then j to preserve and maintain the flow of for loop for the next iteration. Thus, in this way outer and inner loops are fully maintained. 3. Termination At long last, we inspect what happens when the circle ends. The condition causing the for loop to end is that $j > A.length = n$. Since each circle cycle builds j by 1, we should have $j = n + 1$ around then. Subbing $n + 1$ for j in the phrasing of circle invariant, we have that the subarray $A[1..n]$ comprises of the components initially in $A[1..n]$ yet all at once in arranged request. Seeing that the subarray $A[1..n]$ <p>Is the whole exhibit, we presume that the whole exhibit is arranged. Thus, the calculation is right.</p>

Strengths	<ul style="list-style-type: none"> • Simplicity • Good Performance for lesser elements • Space Requirement is minimal
Weaknesses	<ul style="list-style-type: none"> • Bad Performance for greater elements • Useful for only lesser elements • Requires a large number of shifting

Table 4. Detail Analysis of insertion sort

3.1.2 Selection Sort

<i>Selection Sort</i>	
Description	<p>There are a wide range of ways of arranging the cards. One of them I'll discuss below and through it I'll try to explain the concept of selection sort:</p> <ul style="list-style-type: none"> • Track down the littlest card. Trade it with the main card. • Track down the second-littlest card. Trade it with the subsequent card. • Track down the third-littlest card. Trade it with the third card. • Rehash tracking down the following littlest card, and trading it into the right situation until the exhibit is arranged.  <p>One of the means in selection sort is to track down the smallest card to place into its right area. For instance, if the cluster at first has values [12, 18, 20, 6, 10, 22].</p> <p>Now what we need to do is to first the smallest card among the bunch of cards and place it into its right position by swapping it with that particular element where it is the right position of the that card.</p>

	<p>Here, it can be seen that 6 is smallest card among them or you can assume it to be the smallest element of the array.</p> <p>Selection sort would swap the worth at position 6 with the index at list 0,</p> <p>Giving [6, 12, 18, 20, 10, 22]. Now we will find the second smallest among them and place it right in the index 1.</p> <p>Notice that since the littlest element has as of now been traded into record 0, what we truly need is to track down the littlest element in the piece of the cluster of cards that beginnings at index 1. We consider a segment of a cluster a subarray, so that for this</p> <div data-bbox="893 756 1339 1323"><p>93 is largest</p><p>77 is largest</p><p>55 is largest</p><p>54 is largest</p><p>44 is largest stays in place</p><p>31 is largest</p><p>26 is largest</p><p>20 is largest</p><p>17 ok list is sorted</p></div> <p>Situation, we need the index of the littlest worth in the subarray that beginnings at record 1. For our model, on the off chance that the full exhibit is [6, 12, 10, 18, 20,, 22] , the littlest worth in the subarray beginning at record 1 is 10, and it has file 4 in the first cluster. So record 4 is the area of the second-littlest component of the full exhibit.</p>
Pseudo Code	<p>SelectionSort(A, s)</p> <ol style="list-style-type: none">1. N=length.A2. For j in N3. min = j4. for k = j+1 to N5. If (A[k] < A[min]

	6. Min = 1 7. Swap (A [i] , A [min])																
Python Code	<pre>def SelectionSort(Array, s) N= len(A) for i in range (N): min = j for k in range(j+1, N): if (A[i] < A[min]): min= i A[i], A[min] = A[min], A[i]</pre>																
Time Complexity Analysis	<table border="1"> <thead> <tr> <th>Cost</th><th>No of time it executes</th></tr> </thead> <tbody> <tr> <td>C1</td><td>1</td></tr> <tr> <td>C2</td><td>n-1</td></tr> <tr> <td>C3</td><td>n-1</td></tr> <tr> <td>C4</td><td>$\sum_{j=1}^{n-1} (n-j+1)$</td></tr> <tr> <td>C5</td><td>$\sum_{j=1}^{n-1} (n-j)$</td></tr> <tr> <td>C6</td><td>$\sum_{j=1}^{n-1} (n-j)$</td></tr> <tr> <td>C7</td><td>n-1</td></tr> </tbody> </table> <p>Total Running Time $(T(n)) = C_1 * 1 + (C_2 + C_3) * (n - 1) + C_4 * \sum_{j=1}^{n-1} (n-j+1) + (C_5 + C_6) * \sum_{j=1}^{n-1} (n-j) + C_7 * (n-1)$</p> <p>Running Time for Worst case (Array is fully un-sorted) T (n) = O (n²)</p> <p>Running Time for Good case (Array is nearly sorted) T (n) = O (n)</p>	Cost	No of time it executes	C1	1	C2	n-1	C3	n-1	C4	$\sum_{j=1}^{n-1} (n-j+1)$	C5	$\sum_{j=1}^{n-1} (n-j)$	C6	$\sum_{j=1}^{n-1} (n-j)$	C7	n-1
Cost	No of time it executes																
C1	1																
C2	n-1																
C3	n-1																
C4	$\sum_{j=1}^{n-1} (n-j+1)$																
C5	$\sum_{j=1}^{n-1} (n-j)$																
C6	$\sum_{j=1}^{n-1} (n-j)$																
C7	n-1																
Proof of Correctness	<p>Correctness can be proved through three steps:</p> <ol style="list-style-type: none"> 1. Initialization Selection sort start loop over each index of the array. Prior to the first iteration of the loop, for the outer loop we what we do is select and element from the array and consider it to be the smallest element from the entire array. And if we assume the entire array to be consisting of a single element that it a= is already sorted and for the first index, array is wholly sorted. 2. Maintenance Now what we will do is to prove that loop variation is get 																

	<p>maintained for each iteration of the loop. In the first call of min, it has to look for every element present in the array and so the body of the loop will run n times. N here represents the number of elements in the entire array. Now first elements get sorted and now loop will iterate from 1 to n index of the array and thus making swapping if appropriate. For the further iteration loop will iterate and will go through n-1 to n elements of the array.</p> <p>3. Termination At the point of termination, loop iteration get equal to n. Now, the array contains smallest elements of after comparing it with minimum which is thus the sorted array. It'll get terminated when the whole array is sorted by going though each iteration of the loop invariant and thus algorithm is proved to correct.</p>
Strengths	<ul style="list-style-type: none"> • Good performance on lesser elements • No temporary storage is required • Simplicity
Weaknesses	<ul style="list-style-type: none"> • Bad performance for greater elements • Worst Time Complexity is entirely bad

Table 5. Detail Analysis of Selection Sort

3.1.3 Bubble Sort

<i>Bubble Sort</i>	
Description	<p>One of the simplest sorting algorithms that in may get it easy to understand is the bubble sort. It basically form the basis of sorting in course and DSA. What it actually do is t go through entire elements of array and rearrange them either in ascending order or in descending order,</p> <p>To do this, the algorithm compares number X to the adjacent number Y. If X is higher than Y, the two are swapped and the algorithm starts over.</p>

	<div><div><div>First pass</div><div><div><div>54</div><div>26</div><div>93</div><div>17</div><div>77</div><div>31</div><div>44</div><div>55</div><div>20</div></div><div>Exchange</div></div><div><div><div>26</div><div>54</div><div>93</div><div>17</div><div>77</div><div>31</div><div>44</div><div>55</div><div>20</div></div><div>No Exchange</div></div><div><div><div>26</div><div>54</div><div>93</div><div>17</div><div>77</div><div>31</div><div>44</div><div>55</div><div>20</div></div><div>Exchange</div></div><div><div><div>26</div><div>54</div><div>17</div><div>93</div><div>77</div><div>31</div><div>44</div><div>55</div><div>20</div></div><div>Exchange</div></div><div><div><div>26</div><div>54</div><div>17</div><div>77</div><div>93</div><div>31</div><div>44</div><div>55</div><div>20</div></div><div>Exchange</div></div><div><div><div>26</div><div>54</div><div>17</div><div>77</div><div>31</div><div>93</div><div>44</div><div>55</div><div>20</div></div><div>Exchange</div></div><div><div><div>26</div><div>54</div><div>17</div><div>77</div><div>31</div><div>44</div><div>93</div><div>55</div><div>20</div></div><div>Exchange</div></div><div><div><div>26</div><div>54</div><div>17</div><div>77</div><div>31</div><div>44</div><div>55</div><div>93</div><div>20</div></div><div>Exchange</div></div><div><div><div>26</div><div>54</div><div>17</div><div>77</div><div>31</div><div>44</div><div>55</div><div>20</div><div>93</div></div><div>93 in place after first pass</div></div></div></div> <div><p>In this algorithm what we actually do is we make preferences of calls.</p><p>What I mean by this is it simply takes each element of on array and compares it with the entire array and swap it if appropriated. It will continue swapping unless and until the entire array is sorted. Key points to notice are that it takes the greater elements to the end of the array thus providing the fully sorted array.</p></div>
<p>Pseudo Code</p>	<p>Bubble Sort (A)</p> <div><div>1.</div><div>For i=0 to len(A)-1</div></div> <div><div>2.</div><div>For j=0 to len(A) – i -1</div></div> <div><div>3.</div><div>If A[j] > A[j+1]</div></div> <div><div>4.</div><div>Swap A[j] and A[j+1]</div></div>
<p>Python Code</p>	<pre>def BubbleSort(arr): for i in range (0,size-1): for j in range (0,size-1-i): if (arr[j]>arr[j+1]): arr[j],arr[j+1]= arr[j+1], arr[j]</pre>

<p>Time Complexity Analysis</p>	<table border="1" data-bbox="750 327 1546 575"> <thead> <tr> <th>Cost</th><th>No of Time it executes</th></tr> </thead> <tbody> <tr> <td>C1</td><td>N</td></tr> <tr> <td>C2</td><td>$\sum_{j=1}^n (j)$</td></tr> <tr> <td>C3</td><td>$\sum_{j=1}^n (j-1)$</td></tr> <tr> <td>C4</td><td>$\sum_{j=1}^n (j-1)$</td></tr> </tbody> </table> <p>Running Time for Worst case (Array is fully un-sorted) $T(n) = O(n^2)$</p> <p>Running Time for Good case (Array is nearly sorted) For Best Case i.e., $j = 1$ After Substitution, $T(n) = C1 * n + (C2 + C3) * (n - 1) + C4 * (n - 1)$ $T(n) = O(1)$</p>	Cost	No of Time it executes	C1	N	C2	$\sum_{j=1}^n (j)$	C3	$\sum_{j=1}^n (j-1)$	C4	$\sum_{j=1}^n (j-1)$
Cost	No of Time it executes										
C1	N										
C2	$\sum_{j=1}^n (j)$										
C3	$\sum_{j=1}^n (j-1)$										
C4	$\sum_{j=1}^n (j-1)$										
<p>Proof of Correctness</p>	<p>Algorithms can be proved through three steps:</p> <ol style="list-style-type: none"> <p>1. Initialization</p> <p>Prior to the first iteration of the loop, when not a single has been called after, it can be seen from pseudo code that size-i-1 which means if the length of an array is 8 then the last element of an array that would be of the index 7 would be the smallest element of the array. In general A [n-1] would be the smallest element of an array.</p> <p>2. Maintenance</p> <p>Now we have to show how our algorithm maintains the loop invariants and prove if it is true for k then it should also satisfy all the k+1 steps. In every step of the loop what we do is compare A [j+1] and A[j] and check if A [j+1] is smaller than A[j] if it gets true, then A[j] and A [j+1] gets swap. In this entire scenario, after each iteration the length of the sorted array starts increasing and after first iteration, the first element is the smallest element of the sub-array.</p> 										

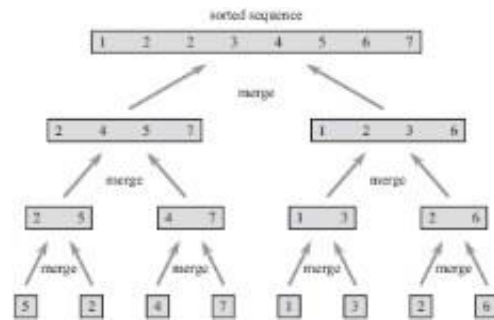
	3. Termination The loop will terminate when i of the outer loop becomes equal to the j of the inner loop. According to the variant, A[i] is the least element of sub-array and it originally consists of elements in A [i...n] before iteration of the loop.
Strengths	<ul style="list-style-type: none"> • Understandable • Simplicity • No temporary memory is required • Perform at constant time if array is sorted
Weaknesses	<ul style="list-style-type: none"> • It is inefficient for greater elements • Lack of efficiency • Takes greater time to sort elements

Table 6. Detail analysis of Bubble Sort

3.2 Recursive Sorting Algorithms

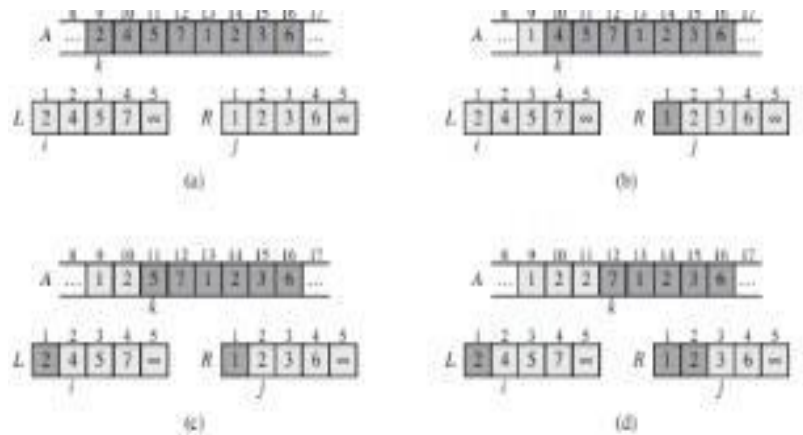
3.2.1 Merge Sort

<i>Bubble Sort</i>	
Description	<p>The merge sort calculation intently follows the divide and conquer approach. It works as follows.</p> <p>Divide: Divide the n-component succession to be arranged into two aftereffects of $n=2$ components each.</p> <p>Conquer: Sort the two aftereffects recursively utilizing consolidate sort.</p> <p>Approach: Merge the two arranged aftereffects to deliver the arranged reply</p>



The critical activity of the consolidation sort calculation is the converging of two arranged arrangements in the "join" step. We converge by calling a helper methodology MERGE(A,p,q,r) where A is an array and p, q, and r are indexes.

Our MERGE strategy sets aside time $O(n)$ and its function



Follows. Getting back to our card playing theme, assume we have two heaps of cards face up on a table. Each heap is arranged, with the littlest cards on top. We wish to combine the two heaps into a solitary arranged yield heap, which is to be face down on the table.

Our fundamental advance comprises of picking the more modest of the two cards on top of the face-up heaps, eliminating it from its heap (which uncovered another top card), and putting this card face down onto the yield heap. We rehash this progression until one info heap is unfilled, at which time we simply take the leftover info heap and spot it face down onto the yield heap.

<p>Pseudo Code</p>	<pre> Merge(A, p, q, r) 1. n₁ = q - p + 1 2. n₂ = r - q 3. L₁[n₁] 4. R[n₂] 5. For i = 1 to n₁ 6. L[i] = A[p + i - 1] 7. For j = 1 to n₂ 8. R[j] = A[q + j] 9. i = j = 1 10. For k = p to r 11. If L[i] < M[j] 12. A[k] = L[i] 13. i = i + 1 14. Else: 15. A[k] = R[j] 16. j = j + 1 17. While(i < and = q) 18. A[k] = L [i] i = i + 1 , k = k + 1 Merge Sort(A, p ,r) 1. If p < r 2. Q = (p + r)/2 3. Merge Sort(A ,p ,q) 4. Merge Sort(A ,q+1 ,r) 5. Merge(A, p , q, r) </pre>
<p>Python Code</p>	<pre> import math def Merge(Arr1, Arr2, A): B=[] i=0 j=0 k=0 s1=len(Arr1) s2=len(Arr2) print(s1) for i in range(s1): if(Arr1[i]<0): A[k] = Arr1[i] k=k+1 </pre>

	<pre> for j in range(s2): if(Arr2[j]<0): A[k] = Arr2[j] k=k+1 for i in range(s1): if(Arr1[i]>=0): A[k] = Arr1[i] k=k+1 for j in range(s2): if(Arr2[j]>=0): A[k] = Arr2[j] k=k+1 def MergeSort(A): s=len(A) if (s > 1): r = s//2 Arr1 = A[:r] Arr2 = A[r:] MergeSort(Arr1) MergeSort(Arr2) Merge(Arr1,Arr2,A) </pre>																
Time Complexity Analysis	<p>Merge Function</p> <table> <tr> <th>Cost</th><th>No of time it executes</th></tr> <tr> <td>C1</td><td>1</td></tr> <tr> <td>C2</td><td>1</td></tr> <tr> <td>C3</td><td>1</td></tr> <tr> <td>C4</td><td>1</td></tr> <tr> <td>C5</td><td>N</td></tr> <tr> <td>C6</td><td>n-1</td></tr> <tr> <td>C7</td><td>N</td></tr> </table>	Cost	No of time it executes	C1	1	C2	1	C3	1	C4	1	C5	N	C6	n-1	C7	N
Cost	No of time it executes																
C1	1																
C2	1																
C3	1																
C4	1																
C5	N																
C6	n-1																
C7	N																

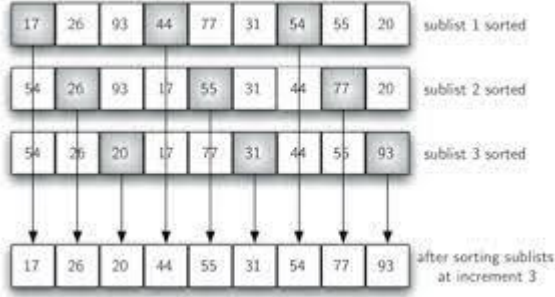
	<table border="1" data-bbox="786 195 1542 632"> <tr><td>C8</td><td>n-1</td></tr> <tr><td>C9</td><td>1</td></tr> <tr><td>C10</td><td>N</td></tr> <tr><td>C11</td><td>n-1</td></tr> <tr><td>C12</td><td>n-1</td></tr> <tr><td>C13</td><td>n-1</td></tr> <tr><td>C14</td><td>n-1</td></tr> <tr><td>C15</td><td>n-1</td></tr> <tr><td>C16</td><td>n-1</td></tr> <tr><td>C17</td><td>n</td></tr> <tr><td>C18</td><td>n-1</td></tr> <tr><td>C19</td><td>n-1</td></tr> </table> <p>Total Running Time $(T(n)) = O(n)$</p> <p>Merge Sort Function</p> <table border="1" data-bbox="792 852 1528 1089"> <tr> <th>Cost</th><th>No of time it executes</th></tr> <tr><td>C1</td><td>1</td></tr> <tr><td>C2</td><td>1</td></tr> <tr><td>C3</td><td>$2T(n/2)$</td></tr> <tr><td>C4</td><td>$2T(n/2)$</td></tr> <tr><td>C5</td><td>$O(n)$</td></tr> </table> <p>Running Time for Worst case (Array is fully un-sorted) $T(n) = O(n \log n)$</p> <p>Running Time for Good case and Average case (Array is nearly sorted) $T(n) = O(n \log n)$</p>	C8	n-1	C9	1	C10	N	C11	n-1	C12	n-1	C13	n-1	C14	n-1	C15	n-1	C16	n-1	C17	n	C18	n-1	C19	n-1	Cost	No of time it executes	C1	1	C2	1	C3	$2T(n/2)$	C4	$2T(n/2)$	C5	$O(n)$
C8	n-1																																				
C9	1																																				
C10	N																																				
C11	n-1																																				
C12	n-1																																				
C13	n-1																																				
C14	n-1																																				
C15	n-1																																				
C16	n-1																																				
C17	n																																				
C18	n-1																																				
C19	n-1																																				
Cost	No of time it executes																																				
C1	1																																				
C2	1																																				
C3	$2T(n/2)$																																				
C4	$2T(n/2)$																																				
C5	$O(n)$																																				
<p>Proof of Correctness</p>	<p>Correctness can be proved through three steps:</p> <ol style="list-style-type: none"> 1. Initialization Before the first iteration, the value of k is equal to p, in that case subarray A [p...k-1] is null. This subarray has k-p=0, thus smallest elements of L and R. Also, i=j=1 at the point L [1] and R[j] have been considered as the smallest elements of array and has not yet been copied back to array A which is actually the final sorted array. 2. Maintenance Now we have to see that either our algorithm maintain the loop variations, in order to get it justified let us assume that 																																				

	<p>$L[i] < \text{and} = R[j]$. Here, $L[i]$ is the smallest element and has not yet been copied back to A. Now thus incrementing k in the for loop and i gets incremented by our own selves just in order to move to the right of the subarray, thus it reestablishes the loop invariant for the next iteration. Same works for j. And for the else case where $L[i] > R[i]$ then the respective lines will perform the accurate and useful loop invariants.</p> <p>3. Termination At the point of termination, k is equal r plus 1. Now, the sub array contains smallest elements of $L[i]$ and $R[i]$ which is thus the sorted array. It'll get terminated when the whole array is sorted by going through each iteration of the loop invariant and thus algorithm is proved to correct.</p>
Strengths	<ul style="list-style-type: none"> • Good performance for greater elements • It is a stable sort and has consistent running time • It breaks the entire array into two parts and lead towards the solution parallel
Weaknesses	<ul style="list-style-type: none"> • Not commendable for lesser elements • Flows through the entire elements of array • Worst Space Complexity

Table 7. Detail analysis of Merge Sort

3.2.2 Shell Sort

<i>Shell Sort</i>	
Description	<p>Shell sort is an improvement of inclusion sort that permits the trading of things that are far separated. The thought is to orchestrate the rundown of components so that, beginning anyplace, taking each h^{th} component delivers an arranged rundown. Such a rundown is supposed to be h-arranged. It can likewise be considered as h interleaved records, each independently arranged. Starting with huge upsides of h permits components to move significant distances in the first rundown, decreasing a lot of turmoil rapidly, and leaving less work for more modest h-sort steps to do. Assuming the rundown is, k-arranged for some more modest number k, then, at that point, the rundown remains h-arranged. Following this thought for a diminishing grouping of h esteems finishing in 1 is ensured to leave an arranged rundown eventually.</p> <p>Let us consider array {38, 16, 34, 20, 52, 28, 12, 45}</p>

	<p>Allow us to consider the accompanying guide to have a thought of how shell sort functions. For our model and simplicity of Comprehension, we take 4 intervals of array. Make a virtual sub-rundown of all values situated at the timespan positions. Here these values are {38, 16}, {34, 20}, {52, 28} and {12, 45}</p>  <p>Now, what we will do is we will compare each value of the sub array and will swap it if necessary into our original array. Now, what we will do is that we will take interval and this gap will generate two sub-lists {38, 16, 34, and 20} and {52, 28, 12, 45}. We will now compare and swap the values, if required. After this array could would now be in a sorted order.</p>
<p>Pseudo Code</p>	<pre> ShellSort(A) 1. G= len (A) // 2 2. While G > 0 3. i=0 4. J= G 5. While j < len(A) 6. If A[i] > A[j] 7. Swap A[i] and A[j] 8. I=i+1 9. j=j+1 10. k=1 11. while k – G > -1 12. if A[k-G] > A[k] 13. Swap A[k-G] and A[k] 14. K=k-1 </pre>
<p>Python Code</p>	<pre> def ShellSort(A, len(A)): gap = len (A) // 2 i=0 j= G </pre>

	<pre> while gap > 0: if A[i] > A[j] temp = array[i] var2 = var i= i+1 j= j+1 while A[var2 - gap] > temp and var2 >= gap : A[var2] = A[var2 - interval] j = j+ gap A[var2] = temp gap //= 2 </pre>
Time Complexity Analysis	<p>It is a comparison based sorting. So, the time complexity of the algorithm depends upon the gap sequence, G in the pseudo code.</p> <p>Running Time for Worst case (Array is fully un-sorted) $T(n) = O(n * \log^2 n)$</p> <p>Running Time for Good case (Array is nearly sorted) $T(n) = O(n * \log n)$</p> <p>In total, it provides you the complexity that lies between $O(n)$ and $O(n^2)$</p>
Proof of Correctness	<p>Algorithm correctness can be proved using three steps:</p> <ol style="list-style-type: none"> 1. Initialization The initialization of the shell sort is somehow similar to insertion sort. The first loop of the shell sort is $j < \text{len}(A)$ and swapping the prior settled indexes with their partial gaps. Thus, at the first settled index according to the unbiased gap the first element would be sorted 2. Maintenance Now, I'll try to show that each element of the array maintains the loop invariant. As discussed earlier, the body of the loop iterates in a way that it moves element to right side and insert the desired element at the appropriate place i.e., $A[j-1]$, $A[j-2]$ and so on. The sub-array then consists of all the sorted elements. We increment then j to preserve and maintain the flow of loop for the next iteration. Thus, in this way outer and inner loops are fully maintained. And gets swap when find appropriate to do so. 3. Termination

	The loop will terminate when i of the outer loop becomes equal to the j of the inner loop. According to the variant, A[i] is the least element of sub-array and it originally consists of elements in A [i...n] before iteration of the loop.
Strengths	<ul style="list-style-type: none"> • Very efficient for average elements • Faster than Insertion Sort • 5 times faster than Bubble Sort
Weaknesses	<ul style="list-style-type: none"> • Complex Algorithm • Less faster than merge sort • Slower for greater elements

Table 8. Detail analysis of Shell Sort

3.3.3 Quick Sort

Quick Sort	
Description	<p>Quick sort is an efficient algorithm to which works on the principle of divide and conquer rule. Quick sort sets the pivot and split the array in two part one is left part of pivot and other is right part of pivot.</p> <p>The diagram illustrates the steps of the Quick Sort algorithm. It shows an array [54, 26, 20, 17, 77, 31, 44, 55, 23] being partitioned around a pivot. The diagram shows the movement of left and right pointers, the exchange of elements, and the final sorted array [17, 20, 23, 26, 31, 44, 54, 55, 77].</p> <p>The vital interaction in quickSort is segment (). Focus of parts is, given an exhibit and a component x of cluster as turn, put x at its Right situation in arranged exhibit and put every more modest component (more modest than x) before x, and put every more prominent component (more noteworthy than x) after x. This ought to be done in straight time.</p> <p>And then again split the array and place the elements on the right position. And by setting the pivot to right position the array will be sorted.</p>
Pseudo Code	<ol style="list-style-type: none"> 1. QUICKSORT(A,p,r) 2. If $p < r$ 3. $Q = \text{PARTITION}(A,p,r)$ 4. $Q = \text{PARTITION}(A,p,r)$

	5. QUICKSORT(A,p,q-1) 6. QUICKSORT(A,q+1,r) 7. Partition(A,p,r) 8. x = A[r] 9. i = p - 1 10. for j = p to r -1 11. if A[j] <= x 12. i = i +1 13. Exchange A[i] with A[j] 14. Exchange A[i + 1] with A[r] 15. Return
Python Code	<pre> from functools import reduce def digitWithMaximumLength(Array): max=Array[0] for i in range(0, len(Array)): if(max< Array[i]): max=Array[i] return len(str(max)) def ReduceBucketList(A): return reduce(lambda x, y: x + y, A) def radixSort(A): noOfBuckets = digitWithMaximumLength(A) for k in range(0, noOfBuckets): B = [[] for i in range(10)] for i in A: p = i // 10 ** (k) % 10 B[p].append(i) A = ReduceBucketList(B) return A n= int(input("Enter how many numbers you want to enter :")) A=[] for i in range (n): num= int(input("Enter Numbers :")) try: A.append(num) except: </pre>

	<pre>print("Not possible! Reconsider!")</pre> <pre>B= radixSort(A)</pre> <pre>print(B)</pre>																
Time Complexity Analysis	<table border="1"> <thead> <tr> <th>Cost</th><th>No of time it executes</th></tr> </thead> <tbody> <tr><td>C1</td><td>cn</td></tr> <tr><td>C2</td><td>cn</td></tr> <tr><td>C3</td><td>cn</td></tr> <tr><td>C4</td><td>cn</td></tr> <tr><td>C5</td><td>cn</td></tr> <tr><td>C6</td><td>cn</td></tr> <tr><td></td><td>O(nlogn)</td></tr> </tbody> </table>	Cost	No of time it executes	C1	cn	C2	cn	C3	cn	C4	cn	C5	cn	C6	cn		O(nlogn)
Cost	No of time it executes																
C1	cn																
C2	cn																
C3	cn																
C4	cn																
C5	cn																
C6	cn																
	O(nlogn)																
Proof of Correctness	<p>Initialization:</p> <p>Prior to the first iteration of the loop, $i = p - 1$ and $j = p$. Because no values lie between p and i and no values lie between $i + 1$ and $j - 1$, the first two conditions of the loop invariant are trivially satisfied. The assignment in line 1 satisfies the third condition.</p> <p>Maintenance:</p> <p>We consider two cases, depending on the outcome of the test in line 4. Figure 7.3(a) shows what happens when $A[j] > x$; the only action in the loop is to increment j. After j is incremented, condition 2 holds for $A[j - 1]$ and all other entries remain unchanged. Figure 7.3(b) shows what happens when $A[j] \leq x$; the loop increments i, swaps $A[i]$ and $A[j]$, and then increments j. Because of the swap, we now have that $A[i] \leq x$, and condition 1 is satisfied. Similarly, we also have that $A[j - 1] > x$, since the item that was swapped into $A[j - 1]$ is, by the loop invariant, greater than</p>																
Strengths	<ul style="list-style-type: none"> • Average time complexity of Quick sort is $O(n \log n)$ • Average case of Quick sort performs faster than merge sort • It does not require any additional memory to sort data 																
Weaknesses	<ul style="list-style-type: none"> • Worst case time complexity of Quick sort is $O(n^2)$ which is worse than insertion sort. • Quick sort is not a stable sort • Quick sort time can vary according to the inputs of array 																

Table 9. Detail analysis of Quick Sort

3.3.4 Recursive Bubble Sort

Recursive Bubble Sort																															
Description	<p>One of the simplest sorting algorithms that in may get it easy to understand is the bubble sort.</p> <p>Now we will try to implement it through recursive approach and will see if it makes any difference through recursive approach. Through what we have studied far enough we have come to know that through levels it get divides when divide them recursively its complexity get reduces but there w might face a problem as our loop works somehow similar to the one recursive approach. We here by iterative method face two for loops when may get reduce by a single base case approach. But what will we do when we consider our second loop?</p> <div><p>Original Array</p><table><tr><td>-5</td><td>2</td><td>33</td><td>10</td><td>-7</td></tr></table><p>Iteration 1</p><table><tr><td>-5</td><td>2</td><td>10</td><td>33</td><td>-7</td></tr><tr><td>-5</td><td>2</td><td>10</td><td>-7</td><td>33</td></tr></table><p>Iteration 2</p><table><tr><td>-5</td><td>2</td><td>-7</td><td>10</td><td>33</td></tr></table><p>Iteration 3</p><table><tr><td>-5</td><td>-7</td><td>2</td><td>10</td><td>33</td></tr></table><p>Iteration 4</p><table><tr><td>-7</td><td>-5</td><td>2</td><td>10</td><td>33</td></tr></table></div> <p>Here, is the problem. So the recursive approach to bubble give us no as such benefits. But we also not be wrong to explore something new.</p> <p>Also it basically form the basis of sorting in course and DSA. What it actually do is that we go through entire elements of array and rearrange them either in ascending order or in descending order.</p> <p>In this algorithm what we actually do is we make preferences of calls.</p> <p>What I mean by this is it simply takes each element of on array and compares it with the entire array and swap it if appropriated.</p> <p>It will continue swapping unless and until the entire array is sorted. Key points to notice are that it takes the greater elements to the end of the array thus providing the fully sorted</p>	-5	2	33	10	-7	-5	2	10	33	-7	-5	2	10	-7	33	-5	2	-7	10	33	-5	-7	2	10	33	-7	-5	2	10	33
-5	2	33	10	-7																											
-5	2	10	33	-7																											
-5	2	10	-7	33																											
-5	2	-7	10	33																											
-5	-7	2	10	33																											
-7	-5	2	10	33																											

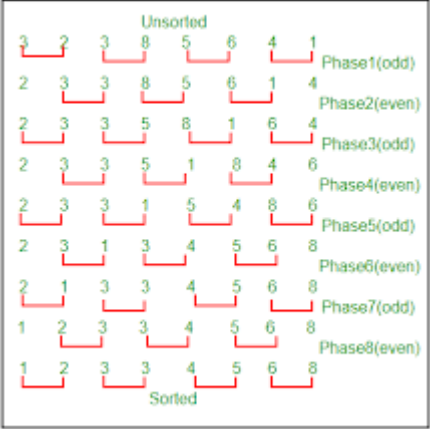
	array.														
Pseudo Code	RecursiveBubbleSort (A, n) <ol style="list-style-type: none"> 1. If n==1 2. Return 3. For i= 0 to len(A)-1 4. If A[i] > A[i+1] 5. Swap (A[i] and A[i+1]) 6. bubbleSortRecursive(A, n-1) 7. 														
Python Code	<pre>def RecursiveBubbleSort (A, ,size= len(A)): if(size == 1) return A; for k in range (size): if(A[size] > A[size + 1]){ A[size], A[size+1]] = [A[size+1], A[size] return RecursiveBubbleSort(A, n-1);</pre>														
Time Complexity Analysis	<table border="1"> <thead> <tr> <th>Cost</th><th>No of Time it executes</th></tr> </thead> <tbody> <tr> <td>C1</td><td>1</td></tr> <tr> <td>C2</td><td>1</td></tr> <tr> <td>C3</td><td>N</td></tr> <tr> <td>C4</td><td>n-1</td></tr> <tr> <td>C5</td><td>n-1</td></tr> <tr> <td>C6</td><td>2T(n/2)</td></tr> </tbody> </table> <p>Total Running Time $(T(n)) = C_1 * 1 + C_2 * 1 + (C_3) * (n) + (C_4) * (n-1) + (C_5) * (n-1) + (C_6) * 2T(n/2)$</p> <p>Running Time for Worst case (Array is fully un-sorted) $T(n) = O(n^2)$</p> <p>Running Time for Good case (Array is nearly sorted) For Best Case i.e., $t_j = 1$ After Substitution, $T(n) = C_1 * n + (C_2 + C_3) * (n - 1) + C_4 * (n - 1) + (C_5 + C_6) * (n - 2) + C_8 * (n - 1)$ $T(n) = O(n)$</p>	Cost	No of Time it executes	C1	1	C2	1	C3	N	C4	n-1	C5	n-1	C6	2T(n/2)
Cost	No of Time it executes														
C1	1														
C2	1														
C3	N														
C4	n-1														
C5	n-1														
C6	2T(n/2)														

Proof of Correctness	<p>Proof of correctness for an algorithm can be proved using three steps: Let us consider them one by one</p> <ol style="list-style-type: none"> 1. Initialization Before the first iteration of the recursive step when it starts from the bottom the array gets divided into portions and recursively divides them at the smaller spots. At the very bottom of the recursive tree we have one element of the array at the left or right side of the array. And we fully know that if an array consists of a single element we can easily deduce that array is fully sorted. 2. Maintenance Now we will see how our code approaches towards the greater loop invariants when we go from bottom to the top when Base Case of $A[n]$ becomes equal to n equal to 1. Inside it we are also making them go through for loop where it compares $A[n]$ and checks if $A[n + \text{iteration of the loop}]$ get greater whenever, this condition gets true it will swap the greater elements and gradually the size of array will get increase. 3. Termination Our function for BubbleSort will get terminate when base case comes into consideration. It will basically start from the end of array will recursively call them when size of array becomes equal to one and hence entire array will get sorted. Thus our algorithm is correct.
Strengths	<ul style="list-style-type: none"> • Understandable and easy to write • Simplicity and a bit efficient • Perform at constant time if array is sorted
Weaknesses	<ul style="list-style-type: none"> • It is inefficient for greater elements • Takes more space than normal bubble sort • Takes greater time to sort elements

Table 10. Detail analysis of Recursive Bubble Sort

3.3 Linear Sorting Algorithms

3.3.1 Brick Sort

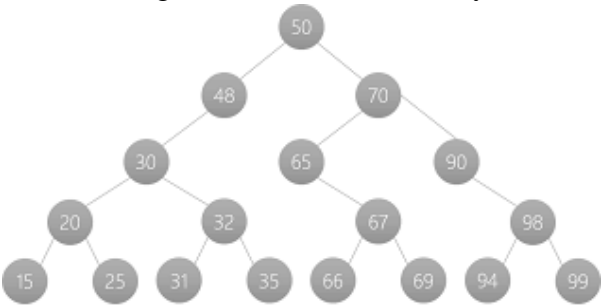
<i>Brick Sort</i>	
Description	<p>This is fundamentally a variety of air pocket sort. This calculation is isolated into two stages Odd and Even Phase. The calculation runs until the exhibit components are arranged and in every cycle two stages happens Odd and Even Phases.</p> <p>In the odd stage, we play out an air pocket sort on odd ordered components and in the even stage, we play out an air pocket sort on even recorded components.</p> 
Pseudo Code	<pre> procedure ODD-EVEN_PAR (n) begin id := process's label for i := 1 to n do begin if i is odd and id is odd then compare-exchange_min(id + 1); else compare-exchange_max(id - 1); if i is even and id is even then </pre>

	<pre> compare-exchange_min(id + 1); else compare-exchange_max(id - 1); end for end ODD-EVEN_PAR </pre>
Python Code	<pre> procedure ODD-EVEN_PAR (n) : begin id := process's label for i := 1 to n do begin if i is odd and id is odd then compare-exchange_min(id + 1); else compare-exchange_max(id - 1); if i is even and id is even then compare-exchange_min(id + 1); else compare-exchange_max(id - 1); end for end ODD-EVEN_PAR end for </pre>
Time Complexity Analysis	<ul style="list-style-type: none"> • Worst Case Complexity : $O(N^2)$ • Average Case Time Complexity : $\Omega(N^2/2^p)$
Strengths	<ul style="list-style-type: none"> • Brick sort is a stable sort • Best case time complexity is lower than other • It doesn't require additional space to sort
Weaknesses	<ul style="list-style-type: none"> • Worst and Average case time complexity is $\Theta(n^2)$

Table 11. Detail analysis of Brick Sort

3.4 Binary Search Tree Sorting Algorithms

3.4.1 Tree Sort

Tree Sort	
Description	<p>Tree sort basically is a sorting algorithm that works on the concept of BST. What exactly BST is? BST stand for Binary Search Tree where it makes searching much more efficient with lesser loss.</p> <p>We need to give it more discussion in order to understand the concept of Tree Sort. It basically works with nodes. What exactly are nodes? Nodes are links through which it attaches one entity with another. It allows the user to search, insert and delete them with greater ease. BST (Binary Search Tree) has</p>  <p>Following properties that should be in account to understand the concept. BST starts from the roots and endorses children just as natural human process works. Our ancestor can be considered our roots. Just like that nodes at the end can considered as grandchildren or further of the roots. It contain two nodes or in other words two children. The tree gets divide into further sub-trees. The left sub-tree of the node will let the elements get linked only which are having data or element that</p> <p>is less than node is key. Moreover, left sub-tree of the node will let the elements get linked only which are having data or element that is greater than node's key.</p> <p>Now, we will see how Tree Sort actually gets worked through the concept of Binary Search Tree. What we do is we first create BST from the elements that we receive them as an input and then we transverse each input element to put them in sorted order.</p>

Pseudo Code	<pre> Class Node 1. Def init 2. E.left= none 3. E.right= none 4. E.value= val 5. Def Combine(E, Val) 6. If val <E.value 7. If E.left= None 8. E.left == Node(val) 9. Else 10. E.left.add(Val) 11. Else 12. If E.right== None 13. E.right= Node(Val) 14. Def TreeSort(S) 15. N= Node(S[0]) 16. For value in S[1:] 17. Node.add(value) 18. Return node.sort() </pre>
Python Code	<pre> class Node: def __init__(element, value): element.left = Null element.right = Null element.val = value def Combine(element, val): if val < element.val: if element.left == None: element.left = Node(val) else: element.left.add(val) else: if element.right == None: element.right = Node(val) else: element.right.add(val) def Treesort(element): array1 = [] array2 = [element.val] array3 = [] if element.left: a = element.left.sort() if element.right: </pre>

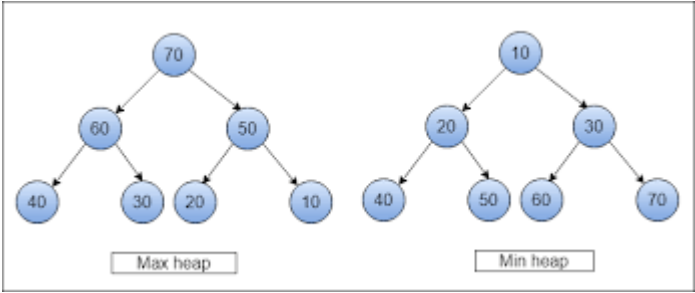
	<pre> array3 = element.right.sort() return array1+array2+array3 def str(element): array1 = "" array2 = "" if element.left: a = element.left.str() if element.right: b = element.right.str() return "{ {}, {}, {} }".format(element.value, array1, array2) def tree_sort(s): node = Node(s[0]) for value in s[1:]: node.add(value) return node.sort() </pre>
Time Complexity Analysis	<p>Running Time for Worst case (Search Tree is balanced) $T(n) = O(n \log n)$</p> <p>Running Time for Worst case (Search Tree is un-balanced) $T(n) = O(n^2)$</p> <p>Running Time for Average case (Search Tree is balanced) $T(n) = O(n \log n)$</p> <p>Running Time for Best Case $T(n) = O(n \log n)$</p> <p>Running Time for Good case (Array is nearly sorted) For Best Case i.e., $t_j = 1$ After Substitution, $T(n) = C_1 * n + (C_2 + C_3) * (n - 1) + C_4 * (n - 1) + (C_5 + C_6) * (n - 2) + C_8 * (n - 1)$</p>
Proof of Correctness	<p>Correctness can be proved through three steps:</p> <p>Initialization</p> <p>It contain two nodes or in other words two children. The tree gets divide into further sub-trees. The left sub-tree of the node will let the elements get linked only which are having data or</p>

	<p>element that is less than node is key. Moreover, left sub-tree of the node will let the elements get linked only which are having data or element that is greater than node's key.</p> <p>Maintenance</p> <p>Now we have to see that either our algorithm maintain the loop variations, in order to get it justified let us assume that element.Left < and = Element.Right. Here, Element.Left is the smallest element and has not yet been copied back to sub-tree. Now thus incrementing k in the for loop and i gets incremented by our own selves just in order to move to the right of the subarray,</p> <p>Tree Sort actually gets worked through the concept of Binary Search Tree. What we do is we first create BST from the elements that we receive them as an input and then we transverse each input element to put them in sorted order thus it reestablishes the loop invariant for the next iteration. Same works for j. And for the else case where Element.Left > Element.right then the respective lines will perform the accurate and useful nodes invariants.</p>
Strengths	<ul style="list-style-type: none"> • Extends capability of array • Can do sorting in logarithmic time
Weaknesses	<ul style="list-style-type: none"> • Tree may get out of balance • It becomes slower at an array access • Can get unregulated if not self-checked

Table 13. Detail analysis of Tree Sort

3.4.2 Heap Sort

Heap Sort	
Description	<p>Heap sort is a tree searching sorting algorithm. It uses a binary tree to sort it out. First of all we make a binary tree of the data by taking the 1st element as root node and by making the rest of the nodes with consecutive elements of the data.</p> <p>e.g. [4,1,8,3,11,6] Take 4 as root node</p> <p>And make other nodes from left to right</p>

	 <p>After making the nodes we will compare each node with the children we will apply the heapify function.</p> <p>Heapify Function: Heapify compares the children nodes of a parent node and swap the greater child node with parent node until the max node will be find out to top So at the end of it we will have a Sorted heap by eliminating the max nodes from the tree.</p>
<p>Pseudo Code</p>	<pre> heapify(arr , i) leftChild = arr [2*i + 1] rightChild = arr [2*i + 2] maxIndex = max(arr[i], leftChild, rightChild) if(i != maxIndex) swap(arr[i], arr[maxIndex]) buildMaxHeap(arr) for n / 2 - 1 to 0 heapify(arr, i) Heapsort(arr) buildMaxHeap(arr) for n-1 to 0 swap(&arr[0], &arr[i]) heapsize-- maxHeapify(arr,0) </pre>
<p>Python Code</p>	<pre> def heapify(arr , i): leftChild = arr [2*i + 1] rightChild = arr [2*i + 2] maxIndex = max(arr[i], leftChild, rightChild) if(i != maxIndex): swap(arr[i], arr[maxIndex]) </pre>

	<pre> def buildMaxHeap(arr) for j in range (n / 2 - 1 to 0): heapify(arr, i) def Heapsort(arr): buildMaxHeap(arr) for i in range n-1 to 0 : arr[0], &arr[i] = arr[i] , arr[0] heapsize = heapsize -1 maxHeapify(arr,0) </pre>
Time Complexity Analysis	<p>Running Time for Worst case $T(n) = O(n \log n)$</p> <p>Running Time for Average case $T(n) = O(n \log n)$</p> <p>Running Time for Best Case $T(n) = O(n \log n)$</p>
Proof of Correctness	<p>Proof of Correctness can be demonstrated utilizing three stages:</p> <p>1. Initialization</p> <p>The introduction of the shell sort is by one way or another like addition sort. The primary circle of the heap sort is $j < \text{len}(A)$ and trading the earlier settled files with their fractional holes. In this way, at the primary settled list as per the unprejudiced hole the main component would be arranged</p> <p>2. Maintenance</p> <p>Presently, I'll attempt to show that every component of the cluster keeps up with the circle invariant. As examined before, the body of the circle emphasizes such that it moves component to right side and addition the ideal component at the suitable spot i.e., $A[j-1]$, $A[j-2]$, etc. The sub-exhibit then, at that point, comprises of the relative multitude of arranged components. We increase then j to safeguard and keep up with the progression of circle for the following emphasis. Along these lines, in this way external and inward circles are completely kept up with. What's more, gets trade when see as fitting to do as such.</p>

	<p>3. Termination</p> <p>The circle will end when I of the external circle becomes equivalent to the j of the inward circle. As indicated by the variation, A[i] is minimal component of sub-cluster and it initially comprises of components in A [i... n] before emphasis of the circle</p> <p>.</p>
Strengths	<ul style="list-style-type: none"> • Efficiency Since heap sort works in $n \log n$, so the time increases in a logarithmic order which makes it efficient because other use the power on n. • Memory usage Heap sort does not require any additional memory to save data, it used the memory equal to the input list. • Simply City: While other sorting algorithms are a bit difficult to understand, it is such an easy algorithm because it does not use the advance concepts of Computer.
Weaknesses	<ul style="list-style-type: none"> • Slow in Procedure: It is a slower in often slower for real numbers,

Table 14. Detail analysis of Heap Sort

3.4.3 Comb Sort

Comb Sort	
Description	<p>Look around Sort is chiefly an improvement Bubble Sort. Air pocket sort consistently looks at adjoining values. So all reversals are taken out individually. Brush Sort enhances Bubble Sort by utilizing hole of size mutiple. The hole begins with an enormous worth and psychologists by a factor of 1.3 in each cycle until it arrives at the worth 1. Accordingly Comb Sort eliminates more than one reversal counts with one trade and performs better compared to Bubble Sort.</p>
Pseudo Code	<p>function combsort(array input) is</p> <p>gap := input.size</p> <p>shrink := 1.3</p>

	<pre> sorted := false loop while sorted = false gap := floor(gap / shrink) if gap ≤ 1 then gap := 1 sorted := true end if i := 0 loop while i + gap < input.size if input[i] > input[i+gap] then swap(input[i], input[i+gap]) sorted := false end if i := i + 1 end loop end loop end function </pre>
Python Code	<pre> def NextGap(diff): diff = (diff * 10)/13 if diff < 1: return 1 return gap def CombSort(A): diff = len(A) diff = NextGap(diff) for i in range(0, n- diff): if A[i] > A[i + diff]: A[i], A[i + diff]=A[i + diff], A[i] </pre>
Time Complexity Analysis	<ul style="list-style-type: none"> • Worst Case Complexity : $O(N^2)$ • Average Case Time Complexity : $\Omega(N^2/2^p)$ • Best Case time Complexity: $\Theta(N \log N)$
Weaknesses	<ul style="list-style-type: none"> • It is not a stable sorting algorithms • Worst case time complexity of Comb sort is $O(n^2)$
Strengths	<ul style="list-style-type: none"> • It is a stable sort • It is efficient • Works best for greater elements

Table 15. Detail analysis of Comb Sort

Accomplishment & Integration

4.1 Integration

Job scratching is the method involved with get-together (rather scratching) position information across the web (model, organization's professions pages) and sharing it in an organized arrangement.

With the assistance of occupation information creeping, high volume of occupation postings is gathered across the web and in the arrangement that accommodates your particular necessities. Here, we will discuss how we go through our integration process of scrapping, what sources were hand-picked as source of data for our project, what kind of difficulties was being faced and what were the reasons if any, we change our source of data.

4.1.1 Source of Scrapping

Though, we had decided at the very start that these websites would be used as a source of data for our project. We had picked such websites that fully fulfill the requirements of our project. If we talk about requirements, what were they? The requirements were though quite simple. We had to choose such a website that should be enriched with data. The website should have data that could fall under our 7 or more attributes.

We were asked to scrap data having seven attributes. We had to gather data giving solid occupation information to work sheets, work aggregators, staffing organizations and progressed recruiting stages, should build the right substance stock dependent on the work market interest, for ed-tech stages, should follow work market patterns with work market information, for statistical surveying and counseling firms. We were destined to gather such data that should follow contenders' employing plan and necessities so clean and fully prepared data should be available to all our clients.

Following websites were selected out from all the Job enriched websites:

- Indeed
- ZipRecruiter
- Glassdoor
- Jobster
- Job Times
- LinkedIn
- SimplyHired

For reference snapshots of the above mentioned websites has been attached hereby.

- **Indeed**

Indeed, is an American overall work site for work postings dispatched in November 2004. It is an auxiliary of Japan's Recruit Co. Ltd. furthermore, is co-settled in Austin, Texas and Stamford, Connecticut with extra workplaces around the world. As a solitary subject web index, it is likewise an illustration of vertical hunt. Without a doubt is at present accessible in more than 60 nations and 28 dialects. In October 2010, Indeed.com passed Monster.com to turn into the most elevated traffic work site in the United States.

The screenshot displays the Indeed website's job search interface. At the top, the Indeed logo is on the left, and links for 'Find jobs', 'Company reviews', 'Submit your CV', 'Sign in', and 'Employers / Post Job' are on the right. Below this is a search bar with 'What' (job title, keywords, or company) and 'Where' (Lahore) fields, a 'Find jobs' button, and an 'Advanced Job Search' link. Below the search bar are several filter buttons: 'Date Posted', 'within 25 kilometers', 'Salary Estimate', 'Job Type', 'Location', 'Company', and 'Job Language'. The main content area is divided into two columns. The left column is titled 'Post your CV - Let employers find you' and shows 'Jobs in Lahore' with a 'Sort by: relevance - date' and 'Page 1 of 5,020 jobs' indicator. A job listing for 'Accounting & Billing Clerk' at 'Compliance Wizard Inc' is highlighted, showing a salary of 'Rs 55,000 - Rs 65,000 a month' and a status of 'Easily apply' and 'Urgently hiring'. The right column is titled 'Get new jobs for this search by email' and contains an 'Email address' input field, a 'Send me new jobs' button, and a disclaimer about creating a job alert.

Figure 3. Snapshot of website Indeed.pk

Following data attributes were in this website:

- Job Title
- Job Company Name
- Job Location
- Job Salary
- Responsiveness
- Description
- Time when job was posted

- **Glassdoor**

Glassdoor is an American site where momentum and previous workers secretly audit companies. Glassdoor additionally permits clients to namelessly submit and see pay rates just as look and go after positions on its foundation.

In 2018, the organization was obtained by the Japanese Firm Recruit Holdings for US\$1.2 billion. It keeps on working independently. The organization is settled in San Francisco, California, with extra workplaces in Chicago, Dublin, London, and São Paulo.

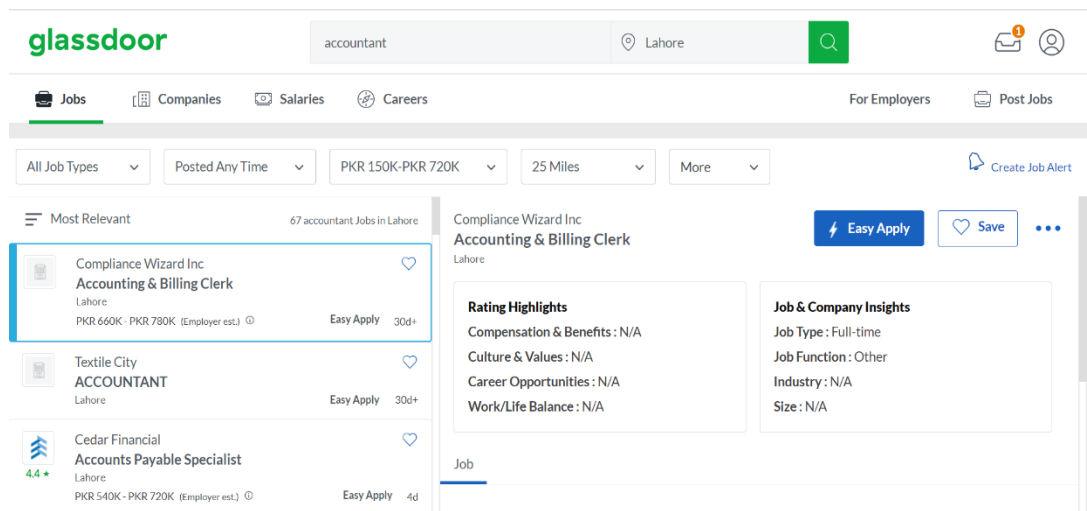


Figure 4. Snapshot of website Glassdoor

Following data attributes were in this website:

- Company Name
- Job Type
- Job Location
- Salary
- Time when job was posted
- Ratings
- Highlights

- **Simply Hired**

Simply Hired is a work site and portable application and an internet based enlistment promoting network situated in Sunnyvale, California. The organization was dispatched in 2003. In 2016, Recruit Holdings Co., Ltd. (proprietor of Indeed.com, a Simply Hired contender), bought Simply Hired. The organization totals work postings from large number of sites and occupation sheets. It then, at that point, promotes those positions on its site and portable application. Occupation searchers search work postings on Simply Hired by catchphrase and area to secure positions of interest.

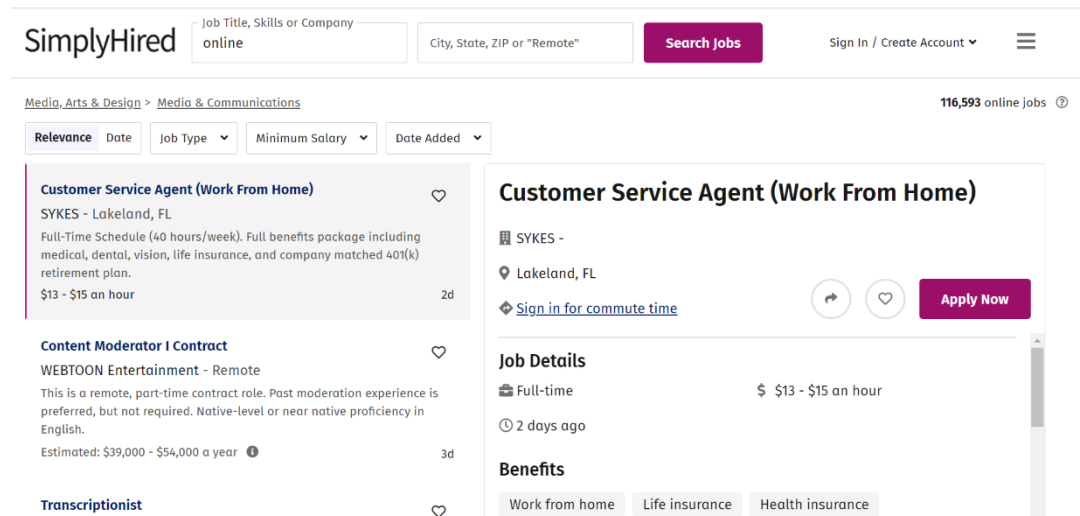


Figure 5. Snapshot of website Simply Hired

Following data attributes were in this website:

- Company Name
- Job Type
- Job Location
- Salary
- Time when job was posted

- **Job Times**

Job Times was initially evolved to help a fruitful vocation counseling office to rapidly and successfully coordinate with quality individuals with extraordinary positions.

Utilizing the force of man-made consciousness, Jobster.io prevailed with regards to taking care of this issue and in doing as such, perceived the requirement for this arrangement in the commercial center.

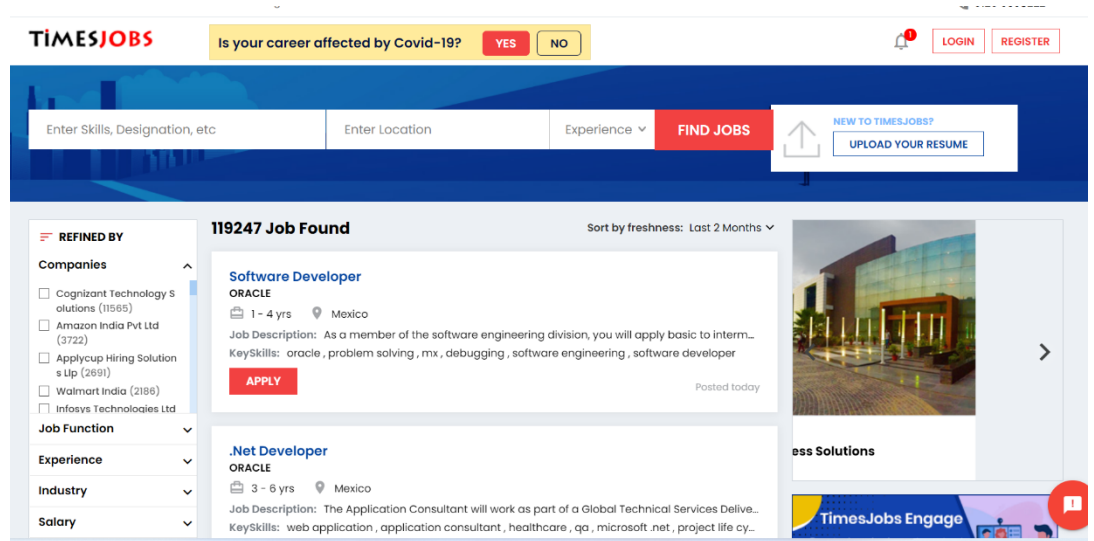


Figure 6. Snapshot of website TimeJobs.com

Following data attributes were in this website:

- Job Title
- Job Company Name
- Job Location
- Job Salary
- Experience
- Description
- Time when job was posted

- Zip Recruiter

ZipRecruiter is an American business commercial center for work searchers and managers. The organization is settled in Santa Monica, California with workplaces in Tempe, AZ; London, UK and Tel Aviv, Israel.

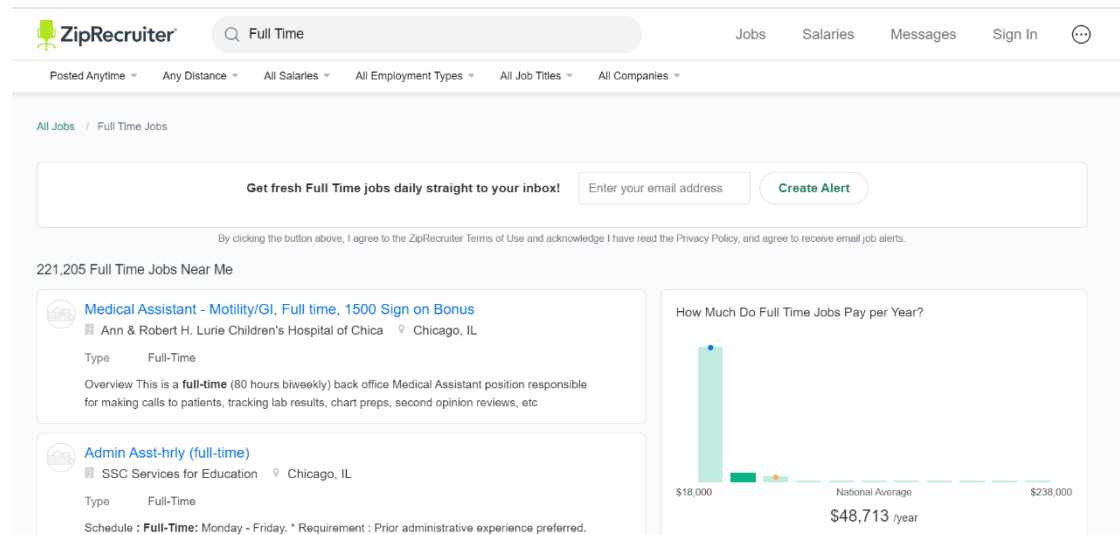


Figure 7. Snapshot of website ZipRecruiter

Following data attributes were in this website:

- Job Title
- Job Company Name
- Job Location
- Job Salary
- Responsiveness
- Description
- Time when job was posted

4.1.2 Difficulties Faced

As a matter of first importance, you'll need to choose where to separate this data. There are two primary kinds of hotspots for work information:

Significant occupation aggregator destinations like Indeed, Monster, Naukri, ZipRecruiter, Glassdoor, LinkedIn, Simply Hired, reed.co.uk, Jobster, JobTimes.com and so forth

Each organization, enormous or little, has a lifelong segment on their sites. Scratching those pages consistently can provide you with the most refreshed rundown of employment opportunities.

Specialty selecting stages in case you are searching for occupations in a specific specialty, similar to occupations for the debilitated, positions in the green business, and so on

4.1.2.1 Against scratching Techniques That Block Job Scraping

Then, you'll need a web scrubber for any of the sites referenced previously. Huge occupation entries can be amazingly interesting to scratch since they will quite often carry out enemy of scratching procedures to keep scratching bots from gathering data off of them. A portion of the more normal squares incorporate IP blocks, following for dubious perusing exercises, honeypot traps, or utilizing Captcha to forestall extreme page visits.

Indeed, there are still ways of bypassing against scratching methods and fix the thing.

4.1.2.2 Shortlisted Job Searching Websites

Now when we have get through a detailed analysis of each website, it was now the perfect time to decide which website should be get selected that could work our purpose. We first check which websites are fulfilling our requirement of having all the seven attributes. So, keeping this convention we shortlisted websites having all the seven attributes.

Now we checked for which website less complexity has. By complexity we mean to say, all the seven attributes should be on the first page, rather than having partial information about job of that particular website in button casing. So, out of all those websites, three websites were picked out.

Those three are listed below:

- ZipRecruiter
- Indeed.pk
- TimesJob.com

Many other problems were faced while dealing with code for these websites, we were avoiding using driver for accessing data as it gives a major drawback of slow speed. And two of these websites, were not allowing us to scrap their data. So, the last website we was left with was

TimesJob.com. The website proved to be quite fruitful as it was totally enriched with data of Jobs, all our seven categories was also there.

4.1.3 Ideal Source for Project

When selecting an ideal source, following domains should be in consideration:

- Consider all the Portals you are willing to scrape
- Consider your search criteria
- Consider all the fields, entities or attributes you willing to extract
- Decide, if you require a Custom Jobs Portal Scraper or Data Services

4.1.3.1 Detail Analysis of Ideal Source

Following table shows detail analysis of jobs on the website TimesJob.com which we have assumed to be ideal for our respective project.

Job Category	No of Jobs
Information Technology	119238
Manufacturing/Engineering	266428
Banking & Finance	54987
SCM & Operations/BPO	5642
Sales & Marketing Jobs	32783
Work from Home	201597

Table 16. Detail analysis of Ideal Source

4.2 Data as an asset

Data is a resource for any association when it's perceived and connected successfully to distinguish the hidden business issues and immediately assembling an answer for address it. Here and there information saw mistakenly can prompt serious issues as business wind up centering in off-base ways. Data is major source for any organizations. And so for our project data is also an asset.

4.2.1 Requirements

According to our project requirements as instructed by our Project Supervisor, we had to scrap 1 Million Data out of different websites relating to the domain of your project, having all 7 seven entities as discussed earlier.

4.2.2 CSV

In order to store data safely somewhere, so it could not get exploit we had to save it in CSV/Excel so such large amount of data couldn't get misplaced and be easily considered if we had to use it somewhere else in our project.

For your reference, screenshot of scrapped data in CSV has been attached hereby.

Job Title	Company Name	Job Location	Job Salary	Job Description	Key Skills	Experience
Workfromhomejobs	GoldenPlacementDivision	Nazik, Sangli, Satara, Solapur, Vasai	None	Hello all, One of the E	Computeroperation	0 - 3 yrs
Workfromhomejobs	Apptof Solutions	Ahmedabad, Bengaluru / Bangalore, Gurgoan	₹ 2.50 - 3.50 Lacs p.a.	Job DescriptionRoles	computerbasic,inter	3 yrs
ContentStrategistworkfromhome	BusiusdolutionsPvt.Ltd	Gurgoan	₹ 3.00 - 5.00 Lacs p.a.	1-3 yrs experienceWe	Editing,Reverting,Fc1	3 yrs
ContentStrategistworkfromhome	BusiusdolutionsPvt.Ltd	Bengaluru / Bangalore	₹ 3.00 - 5.00 Lacs p.a.	1-3 yrs experienceWe	Seo,Englishprofeser	1 - 3 yrs
Workfromhomezeroinvestment	GoldenPlacementDivision	Bangagaoon, Dibrugarh, Guwahati, Gurgoan	None	Online Ad Posters Rec	Computeroperation	0 - 3 yrs
TESTERCUMSCURMASTER	Work BusiusdolutionsPvt.Ltd	Gurgoan	₹ 4.00 - 6.00 Lacs p.a.	We're a fast-growing s	FunctionalTesting,S-2	4 yrs
TESTERCUMSCURMASTER	Work BusiusdolutionsPvt.Ltd	Bengaluru / Bangalore	₹ 4.00 - 6.00 Lacs p.a.	We're a fast-growing s	ApplicationTesting,S-2	4 yrs
Sr.ReactiveDeveloper	Work BusiusdolutionsPvt.Ltd	Chandigarh	₹ 7.00 - 8.00 Lacs p.a.	We're a fast-growing s	SQLAdministration,I3	5 yrs
FullStackDeveloper	WorkfromhomeBusiusdolutionsPvt.Ltd	Bengaluru / Bangalore	₹ 4.80 - 6.00 Lacs p.a.	2-4 yrs experiencePH	Git,HTTP,MySQL,Data2	4 yrs
Sr.FullStackDeveloper	WorkfromhomeBusiusdolutionsPvt.Ltd	Gurgoan	₹ 7.20 - 8.40 Lacs p.a.	4-8 yrs experiencePH	Webarchitecture,ba4	8 yrs
FresherFullStackDeveloper	WorkfromhomeBusiusdolutionsPvt.Ltd	Bengaluru / Bangalore	₹ 3.00 - 4.00 Lacs p.a.	We're a fast growing s	basicsdesignability,5.0	2 yrs
Sr.FullStackDeveloper	WorkfromhomeBusiusdolutionsPvt.Ltd	Bengaluru / Bangalore	₹ 7.20 - 8.40 Lacs p.a.	4-8 yrs experiencePH	Ums,BackendLangs4	8 yrs
FullStackDeveloper	WorkfromhomeBusiusdolutionsPvt.Ltd	Gurgoan	₹ 4.80 - 6.00 Lacs p.a.	2-4 yrs experiencePH	MySQLDatabase,MTT2	4 yrs
FresherFullStackDeveloper	WorkfromhomeBusiusdolutionsPvt.Ltd	Gurgoan	₹ 3.00 - 4.00 Lacs p.a.	We're a fast growing s	jQuery,HTML,CSS,SC0	2 yrs
Workfromhomejobs,DataEntryO	Apptof Solutions	Nagpur, Nanded, Sangli, Satara, Thir	₹ 2.50 - 3.50 Lacs p.a.	Job DescriptionRoles	computerbasic,inter	0 - 3 yrs
Workfromhomejobs,DataEntryO	Apptof Solutions	Ahmednagar, Aurangabad, Kolhapur	₹ 2.50 - 3.50 Lacs p.a.	Job DescriptionRoles	computerbasic,inter	0 - 3 yrs
BPOCallCenterCustomerSupport	SANCTUS	Bengaluru / Bangalore	None	Immediate Openings f	customersupport,hi	0 - 3 yrs
OnlineDataEntryOperatorrequire	MinanshikaSoftTechSolutionPvt.Ltd	Chandigarh, Guwahati, Patna, Raipur	₹ 1.20 - 3.60 Lacs p.a.	Note : It is informedto	MicrosoftExcel,Back0	3 yrs
OnlineDataEntryOperatorrequire	MinanshikaSoftTechSolutionPvt.Ltd	Vijayawada, Visakhapatnam, Cochin	₹ 1.20 - 3.60 Lacs p.a.	Note : It is informedto	MicrosoftExcel,Back0	3 yrs
workfromhome	freelancer	Hyderabad/Secunderabad	₹ 1.05 - 2.10 Lacs p.a.	urgent opening for ho	PackingMaterials	1 - 2 yrs
workfromhome	GoldenPlacementDivision	Junagarh, Jind, Karthal, Cuttack, Ero	₹ 3.00 - 4.00 Lacs p.a.	Position: Fresher / Co	workfromhome	0 - 3 yrs

Figure 7. Snapshot of CSV

4.3 UI Implementation

The (UI) is where human clients communicate with a PC, site or application. The objective of viable UI is to make the client's experience simple and instinctive, requiring least exertion on the client's part to get most extreme wanted result.

UI is made in layers of communication that allure for the human detects (sight, contact, hearable and that's just the beginning). They incorporate both information gadgets like console,

mouse, trackpad, receiver, contact screen, unique finger impression scanner, e-pen and camera and yield gadgets like screens, speakers and printers. Gadgets that collaborate with various faculties are classified "mixed media UIs".

In order to create UI for our respective project, guidelines were given by our project supervisor to get it integrate. We were asked to make maximum screens up to 3 or minimum to 1. The main screen should consist all the information. If we talk about main screen in bit detail, it would consist of Table in which we have to show all the seven attributes that we have scrapped. Moreover, progress bar showing how much data has been scrapped from a certain website. With the optionality, of stopping and pausing it wherever the user wants and starting it again exact from the same place. Moreover, we had to provide combo boxes that would contain all the sorting algorithms that user would be entertained with. Buttons that would facilitate the users in sorting each column either up or down. Also, a searching facility. User would be able to facilitate any search according to its job need. Also, how much time an algorithm takes to sort large amount of data would also be displayed on the screen.

In order to full these requirements, we had to use certain IDE's, libraries and other such things. Detail of these tools has been described in the Chapter ahead.

4.3.1 Qt Designer

Qt Designer is the Qt tool for planning and building graphical UIs (GUIs) with Qt Widgets. You can create and tweak your windows or exchanges in a what-you-see-is-the thing that you-get (WYSIWYG) way, and test them utilizing various styles and goals.

Gadgets and structures made with Qt Designer incorporate consistently with customized code, utilizing Qt's signs and spaces component, so you can without much of a stretch appoint conduct to graphical components. All properties set in Qt Designer can be changed powerfully inside the code. Moreover, highlights like gadget advancement and custom modules permit you to utilize your own parts with Qt Designer.

We were asked to us Qt designer Tool to integrate forms or screens of our UI. As it provides the functionality of Drag and Drop and will generate code for the certain language you want. In our case, it was python so how do we convert it will be discussed in the Chapter later.

4.3.2 PyQt5

PyQt5 is a complete arrangement of Python ties for Qt v5. It is executed as in excess of 35 expansion modules and empowers Python to be utilized as an elective application improvement language to C++ on completely upheld stages including iOS and Android. We were asked to use this library to make GUI in python for our project entitled “Job Search Portal”

4.3.3 Ideas for UI

In this specific section, we will discuss what our idea was for UI and how we proceed to the final look. There is a tool name Pencil Tool. Pencil is worked to give a free and open-source GUI prototyping instrument that individuals can without much of a stretch introduce and use to make mockups in famous work area stages. So, users just like us after having a rough idea for our UI implement in the Qt Designer.

The mock UI for our project created on pencil tool has been attached hereby for reference:

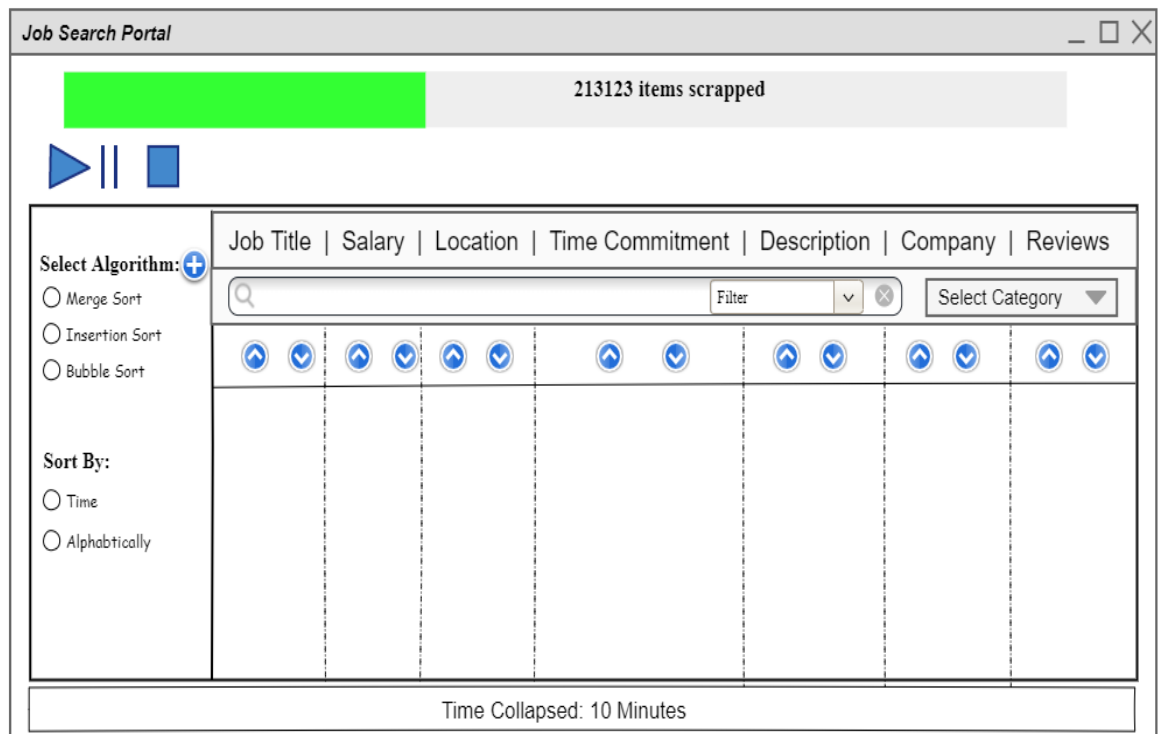


Figure 8. Snapshot of mock UI

So, after being collaborate what we have designed so far has been attached.

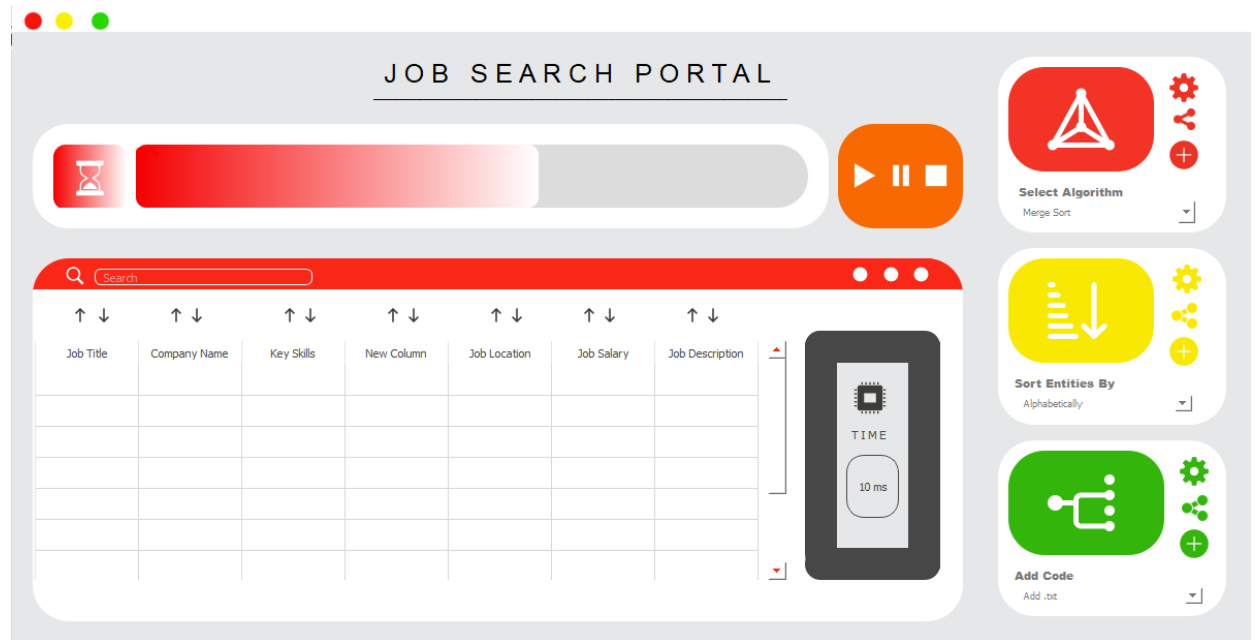


Figure 9. Snapshot of Final UI look

It can be seen the difference between the mock UI made on Pencil tool and the Final Implemented UI. Though styles and designing has totally been changed but structure to the thing lies something in the center.

Discussing it in the bit detail which UI Component was used, elaboration of it has been listed below:

- **Resource File**

In order to make our UI more attractive and user friendly, icons was decided to be part of our UI. In order to achieve that thing, QRC or

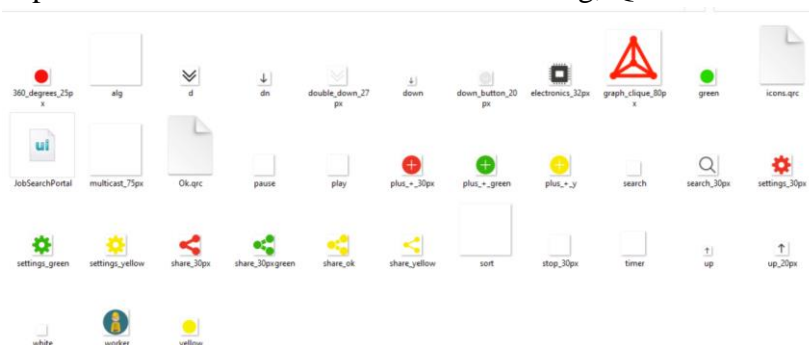


Figure 10. Snapshot of QRC File

Resource file needs to be added in our .ui file. So, all the icons you see on the screen are stored in separate file of Resource.

- **Layout**

Qt designer offers the user a very unique feature that's helps one to design the structures of their UI in more sorted way. So, in the screen at the very right end vertical layout has been selected to form a structure.

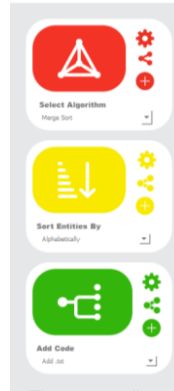


Figure 11. Snapshot of vertical layout in the Final UI

- **Customized Widgets**

Widgets are the essential components for making UIs in Qt. Widgets can show information and status data, get client input, and

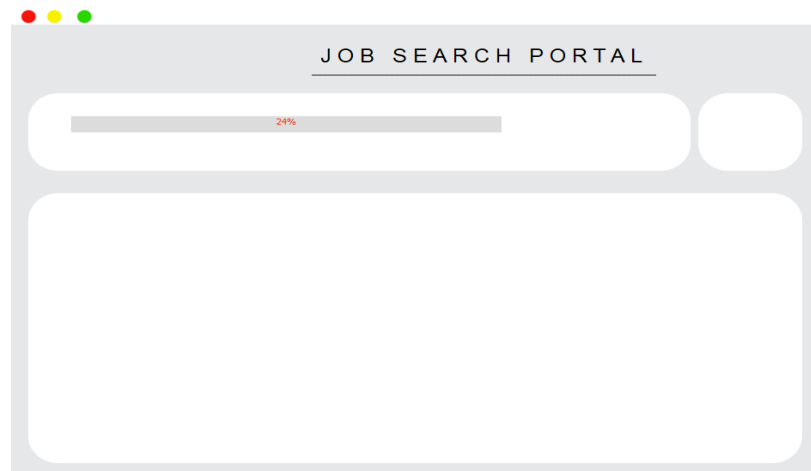


Figure 12. Snapshot of QT Widgets

Give a holder to different gadgets that ought to be assembled together. In order to stand the structure of our UI we add widgets and then customize them according to our need.

- **QTable Widget**

The QTable Widget class furnishes a thing based table view with a default model. Table gadgets give standard table showcase offices to applications. After scrapping data from a certain website, there will need to show data somewhere so user could easily be able to discern what he wishes for. So, in order to fulfill this purpose we use QTable Widget.

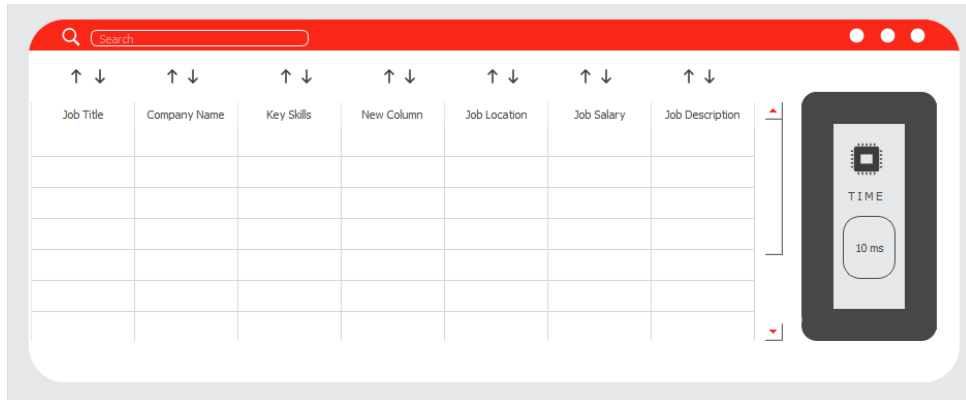


Figure 13. Snapshot of QT Table Widget

- **Progress Bar**

An progress bar is utilized to provide the client with a sign of the advancement of an operation in our case, how much data has been scrapped and to promise them that the application is as yet running.

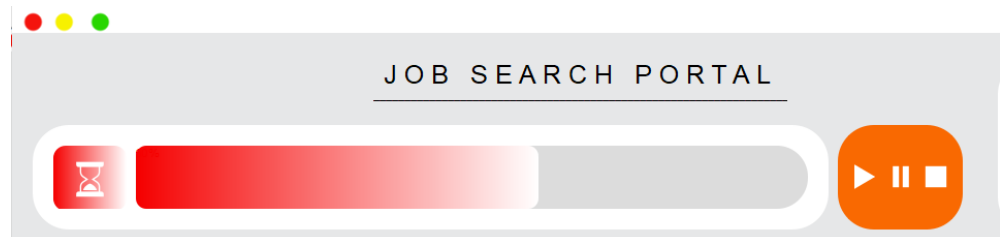


Figure 14. Snapshot of QT Progress Bar

4.3.4 Collaboration

2020-CS-130 and 2020-CS-128, we had collaborated all the way long in order to design the UI. We at the very first step discuss what should be the tools or widgets that are required by our project. We had first discuss it on the pencil paper, after merging both our ideas a simple UI look was decided.

Then, we both partially design it on the Pencil Tool. After that, it was the time to implement it in the very professional way as it was the screen we were going to represent to the user of our project.

4.4 Pace towards Code Execution

Now, when UI was totally designed it was time to write code for what we was intended to do. So, step by step detail of how we proceed is mentioned below:

4.4.1 UI to Py

As, already described in the Chapter above QT designer allows one to generate code automatically for your UI form which you have designed using drag and drop. As, we were using anaconda environment, so we had to use anaconda CMD to convert UI file .py extension. In order to fulfill this task, we need to run command by setting directory first where you UI file has been saved. The snapshot of command in Anaconda Command Prompt has been attached hereby for reference.

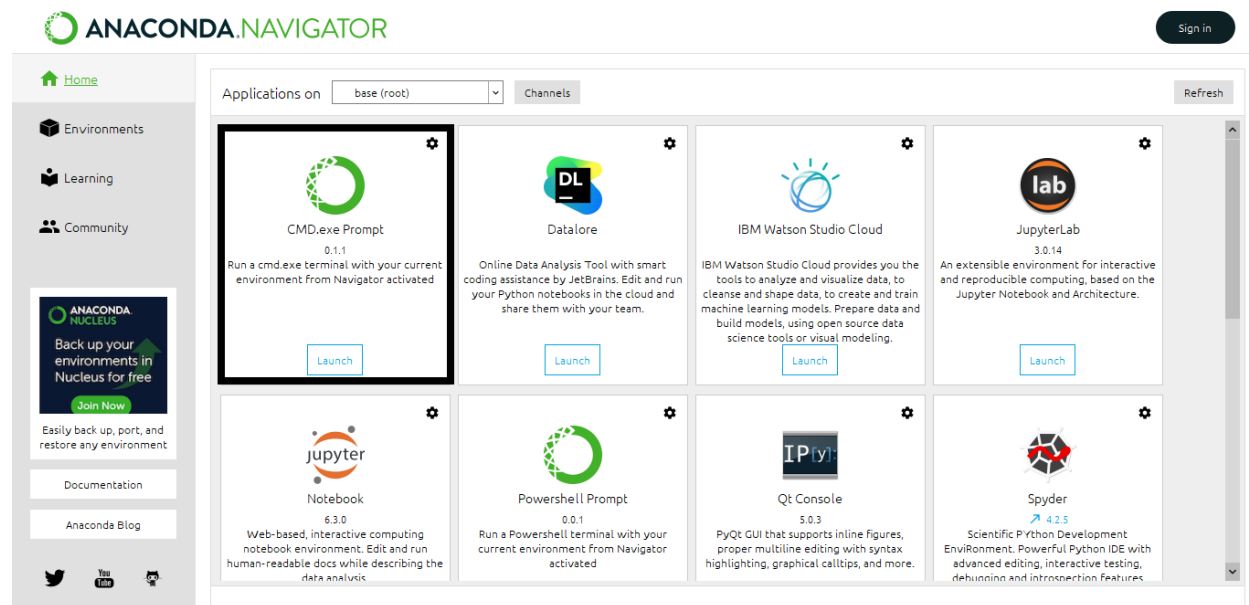


Figure 15. Command Prompt of Anaconda

- Go to the directory where your ui record is.
- Press shift right-click your mouse.
- Snap open order window here .
- This will open the cmd , actually look at what is the index of your (pyuic4. bat) document
- Write in the cmd : `C:\Python34\Lib\site-packages\PyQt4\pyuic4.bat - x filename.ui - o filename.py` (hit Enter)

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.

(base) C:\Users\92309>pyuic5 -x JobSearchPortal.ui -o JobSearchPortal.py_

```

Command to convert .ui to .py

Figure 16. Command to convert .ui to .py

4.4.2 QTable Widget

Table Widget give standard table showcase offices to applications. The things in a QTableWidget are given by QTableWidgetItem. Assuming you need a table that utilizes your own information model you should utilize QTableView as opposed to this class. Table gadgets can be developed with the necessary quantities of lines and section.

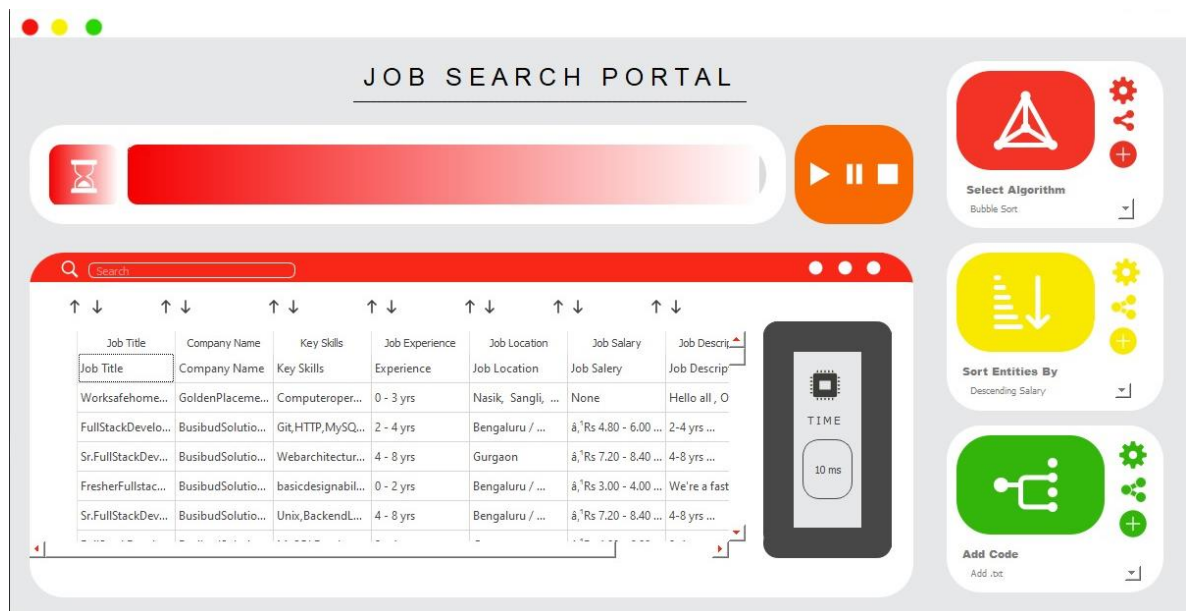


Figure 16. Scrapped Data into QTable Widget

In this step, we had to use our QTable Widget to get the data from csv or directly from website, storing them on lists of python and then displaying the data to the user. As, QTable Widget uses items to get the data into itself, so for this purpose we dynamically set the rows to get the data inserted into itself.

4.4.3 Mouse Events

The MouseEvent interface addresses occasions that happen because of the client collaborating with a pointing gadget (like a mouse). Normal occasions utilizing this interface incorporate click, dblclick, mouse up, mouse down. The below picture shows where we incorporate mouse events.

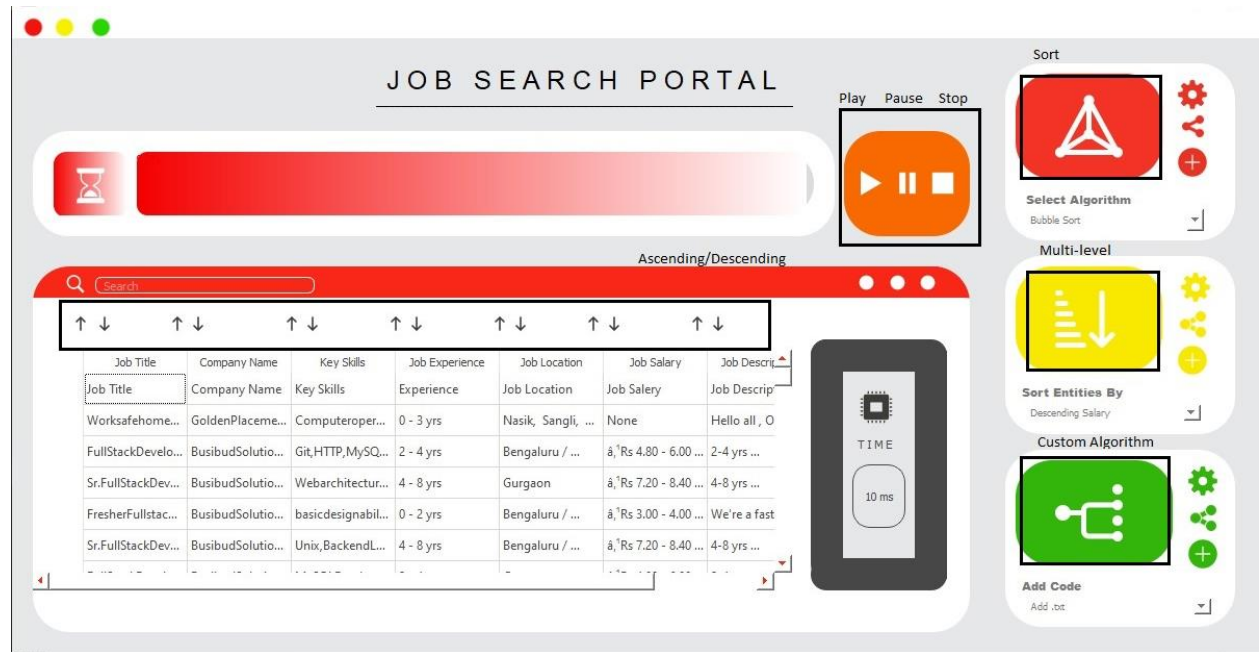


Figure 17. Scrapped Data into QTable Widget

From Figure 17, the blocks in the black are the areas in the UI where mouse events would be incurred. Following are the name of the mouse events:

- Play
- Pause
- Stop
- Ascending/Descending
- Sort
- Multi-level Sorting
- Custom Algorithm

4.4.4 Sorting Ascending/Descending

Here, it was the need to sort our data according to the Algorithm that was selected by the user. The user need to select algorithm name from the combo box where all the algorithms has already been added. So after selecting algorithm, all data would be sorted according to that specific algorithm and time would be shown very next to the table in TIME widget.



Figure 18. Taking Algorithm as input from combo box

From Figure 18, what we do we take input from user from combo box by the use of combo events of PyQt5. We apply conditions on which type of algorithm was desired by the user, and then call that respective function of algorithm in the if else condition of combo box. The respective array of entity get sorted and gets displayed in the QTable Widget.

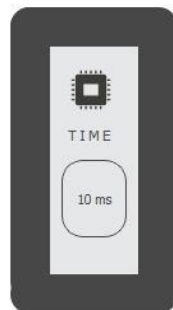


Figure 19. Outputting time taken by a certain algorithm

Moreover, how much the user selected algorithm takes to sort such a large amount of data will also be get displayed in the Time widget. After calling time function, time function will return value and will set the label in the widget to value.



Figure 20. Sorting each algorithm in ascending/ descending order

You can also sort a particular column on ascending or descending basis, 0 to onwards in case of numbers/ onwards to 0 and A-Z or Z-A.

One can also search particular entity by giving the input in the Search Bar.

4.4.5 Multi-Level Sorting

We will check each character of the string and sort them by ascending order. On reviews bases we perform sorting as more reviews more salary is assigned and on reviews basis salary is sorted and on salary bases the job title is sorted so total 3 columns are sorted reviews salary and job title.

Here, we will also entertain our user through a specific kind of multilevel sorting. It will help one's using that scrapping tool with multilevel ascending order of alphabets. It will sort three columns Job Title, Location and Company Names e.g. you want to sort and fetch all the Job Titles starting with letter B it will firstly display all the Job Titles starting with B, parallel will sort Locations with B and vice versa for Company Names. It will help organization for better job analysis.

4.5 Unfinished Ideas

4.5.1 Custom Sorting Algorithm

We had decided in the very start that if user, company or any organization wants to sort the scrapped on the basis of their own algorithm they could add them. What only they need to do is create a tested text file of code of an algorithm and add it from the Custom Widget as shown in Figure 21. But due to certain difficulties it was unable to get completed. But in the near future, we will get through the difficulties and update it for our end users.

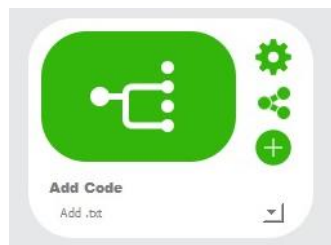


Figure 22. Adding Custom Algorithm

4.5.2 Dynamic Scrapping

While Dynamic sites are of extraordinary advantage to the end client and the designer, they can be hazardous when we need to scratch information from them. For instance, think about that in a unique site page: a large part of the usefulness occurs in light of client activities and the execution of JavaScript code with regards to the program. Information that is consequently created, or shows up 'on request', and is 'naturally produced' because of client communication with the page can be hard to repeat automatically at a low level – a program is a beautiful modern piece of programming all things considered!

Because of this degree of dynamic association and interface mechanization, it is hard to utilize a basic http specialist to work with the powerful idea of these sites and we want an alternate methodology. The easiest answer for scratching information structure dynamic sites is to utilize a robotized internet browser, like selenium, which is constrained by a programming language like Python.

4.6 Task Division

Group Member	Task
2020-CS-130 & 2020-CS-128	Idea of UI on Notebook`
2020-CS-130	Updated Idea
2020-CS-128	Mock UI on Pencil Tool
2020-CS-128	Updated Mock UI on Pencil Tool
2020-CS-130	Exploring Qt designer
2020-CS-130	Forming a structure by widgets
2020-CS-128	Vertical Layout of UI
2020-CS-130 & 2020-CS-128	Icons and Resource File
2020-CS-130	Color Scheme
2020-CS-128	Table Widget and Progress Bar
2020-CS-130 & 2020-CS-128	Customizing Widgets using stylesheet
2020-CS-130	Final UI Look
2020-CS-128	Updated Final UI Look
2020-CS-130	Description of 7 Algorithms

2020-CS-128	Description of 5 Algorithms
2020-CS-130	Proof of correctness of 7 Algorithms
2020-CS-128	Proof of correctness of 5 A
2020-CS-128	Pseudo Code
2020-CS-130	Python Code
2020-CS-130	Strengths
2020-CS-128	Weaknesses
2020-CS-130	Loading data from csv to Table
2020-CS-130 & 2020-CS-128	Mouse Events
2020-CS-128	Sorting through Insertion Sort
2020-CS-128	Sorting through Merge Sort
2020-CS-130	Sorting through Quick Sort
2020-CS-130 & 2020-CS-128	Sorting on each individual column
2020-CS-128	Commencing Progress Bar
2020-CS-130	Time Function

User Manual

5.1 How to commence Scrapping

When you will launch the application the very first screen you'll see has been attached hereby for reference in Figure

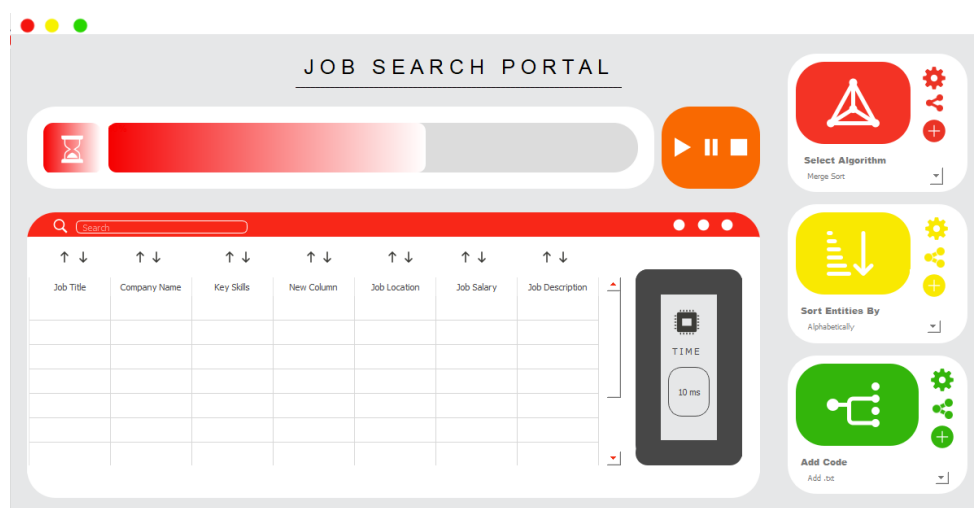


Figure 22. First Screen when application gets launched

In order to start scrapping, you have to click on the Play button. As soon as play button gets pressed, scrapped data will start getting plot in the Table in their respective columns.

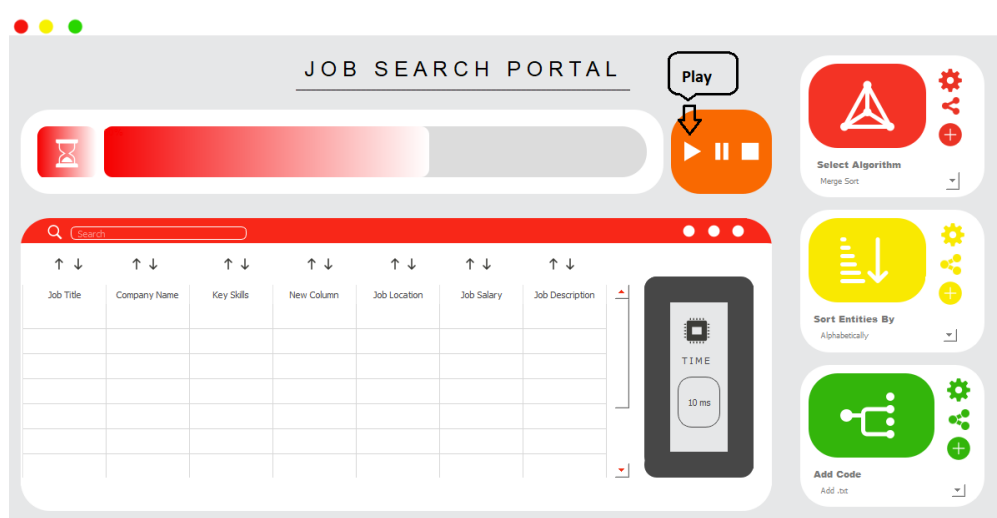


Figure 23. Commence Scrapping

Just when you had pressed the play button data will be plotted in fields as in Figure 23

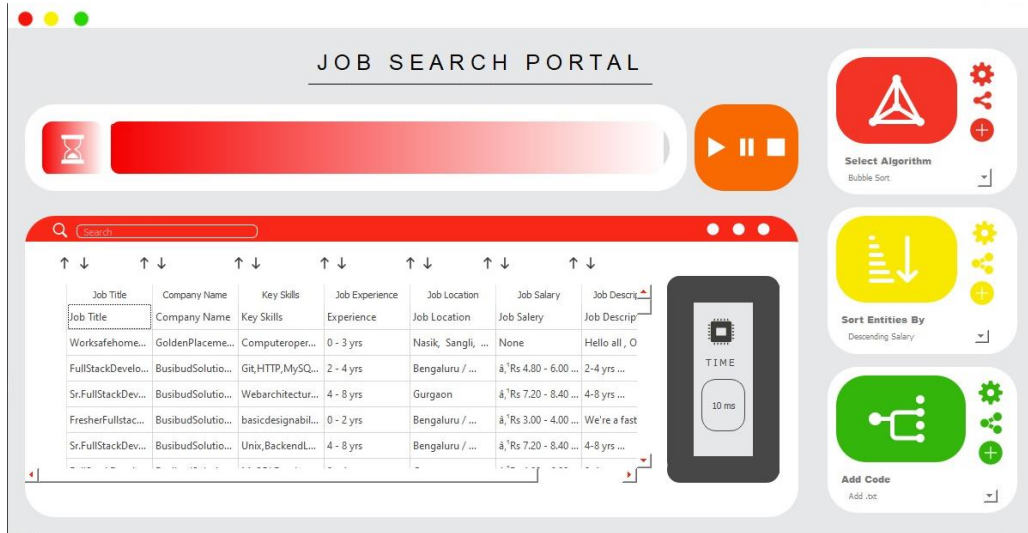


Figure 23. Data plotted in fields

5.2 How to Pause/Stop Scrapping

In order to stop scrapping, you have to press Stop button. It will stop the scrapping and data will be stored only up to there. And in order to put a hold on scrapping and pausing it for a while, so you can continue it again, you have to click on Pause button. For reference, snapshot has been attached in Figure 24.

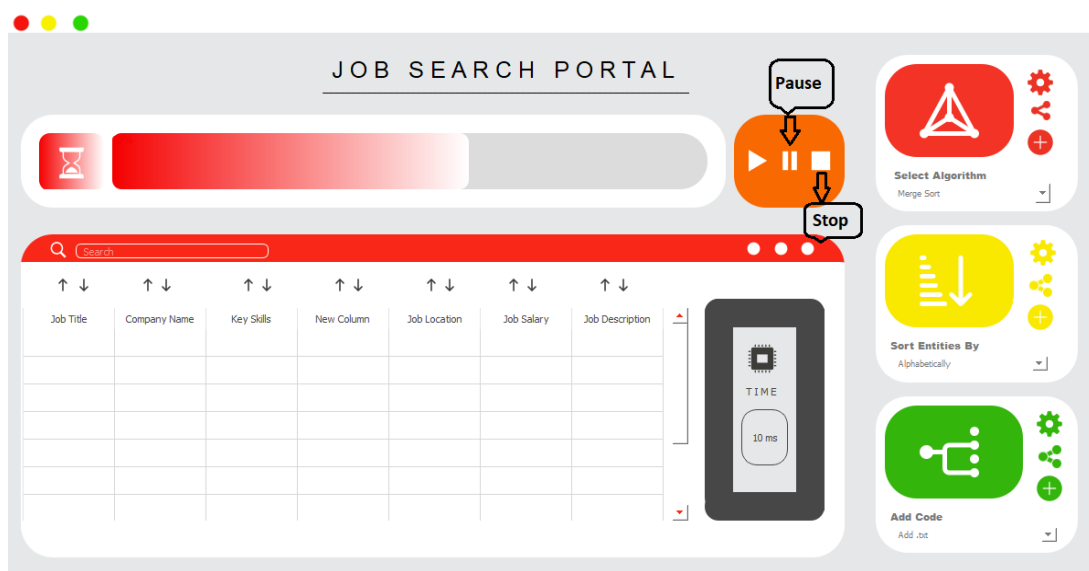


Figure 24. How to Pause/Stop Scrapping

5.3 How to Sort Data

In order to sort data, you have to click on combo box in the very up-right of the screen as shown in Figure 25.

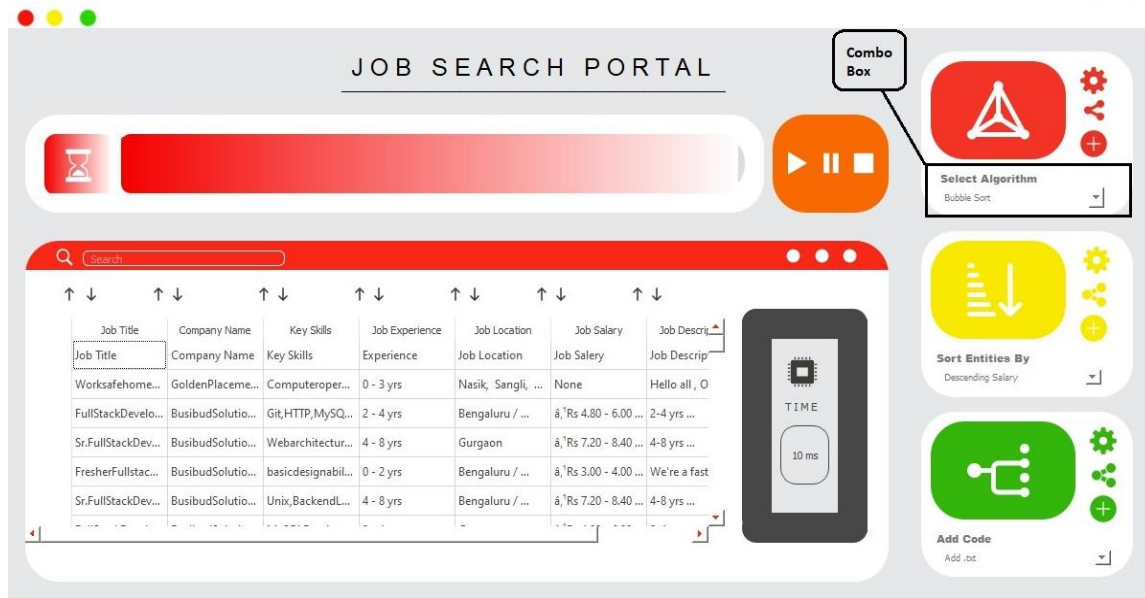


Figure 25. Selection Algorithm from combo box

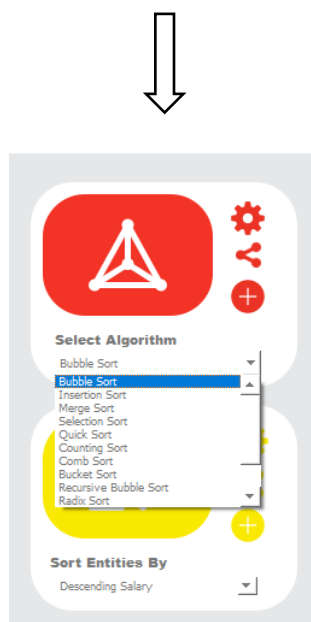


Figure 26. List of Algorithms in combo box

So, after selecting an algorithm from the list of combo box, and then clicking on the red prism button, your data will be get sorted according to that specific algorithm.

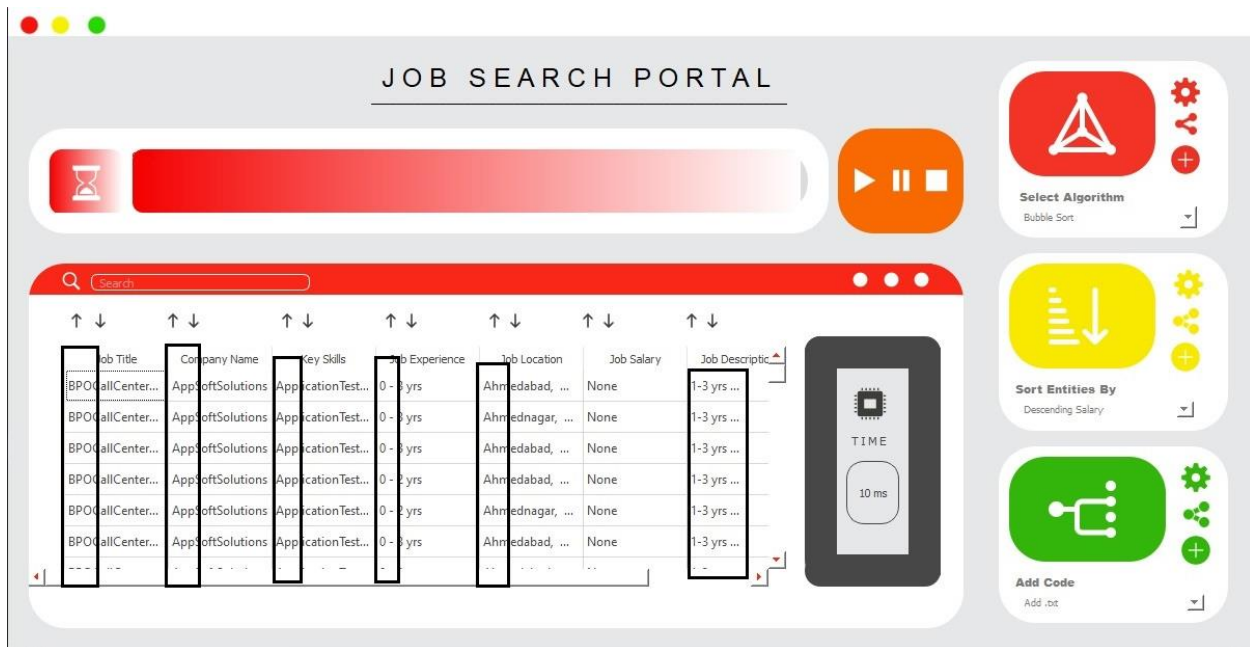


Figure 27. Snapshot of sorted data

You can also sort your desired column by pressing ascending/descending button above each row.



Figure 28. Sorting your desired column

Moreover, after sorting how much it takes to sort will be displayed in Time widget.

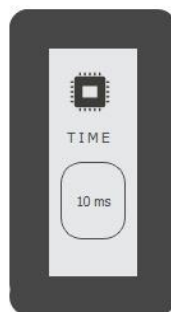


Figure 29. Outputting time taken by a certain algorithm

You can also sort your data on the multilevel basis, you have to click on combo box in the very up-right of the screen as shown in Figure 30.

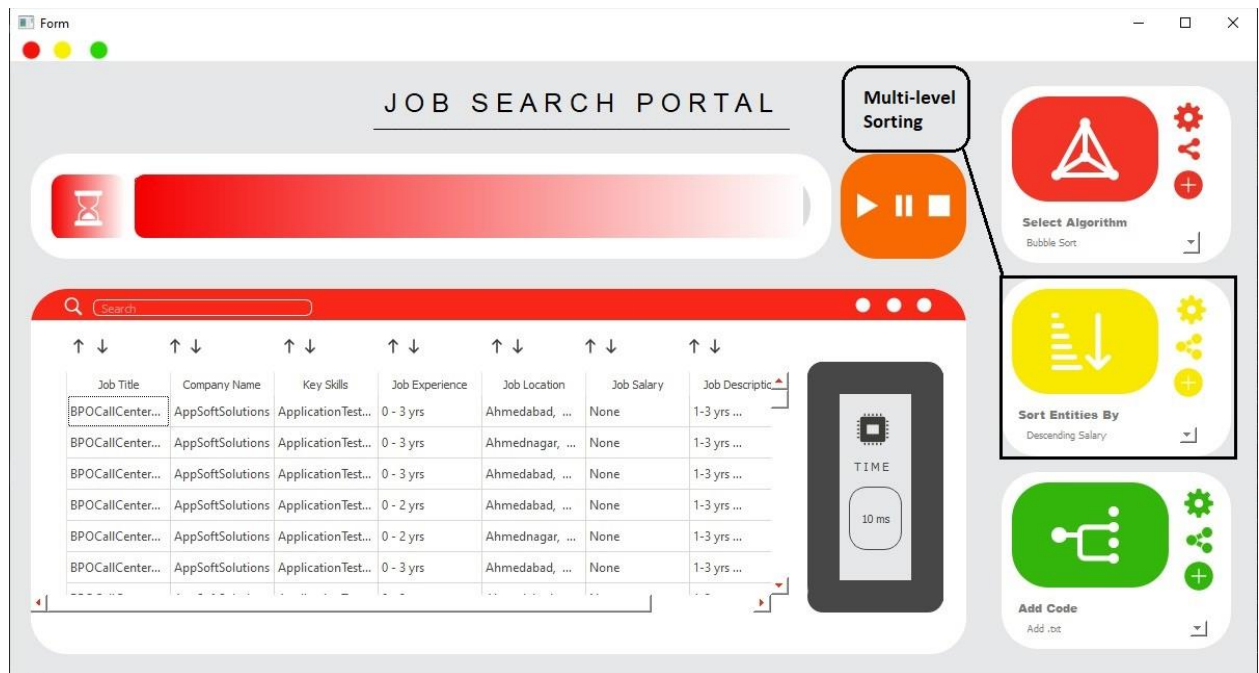


Figure 30. Selection of Multi-level Sorting

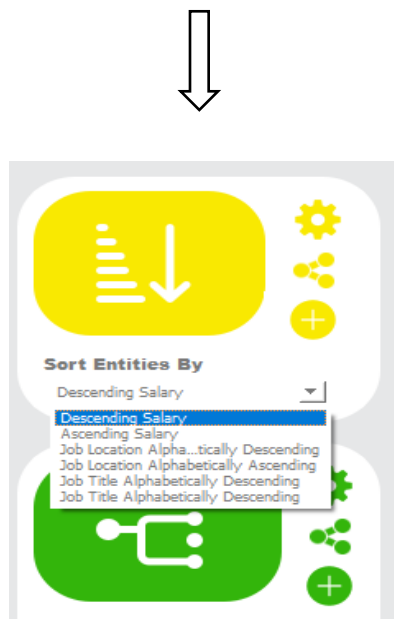


Figure 31. List of Multi-level Sorting Algorithms in combo box

Thus, after selecting any multi-level sorting algorithm from the combo box, your data will get sorted and will be displayed in the widget along with the time.