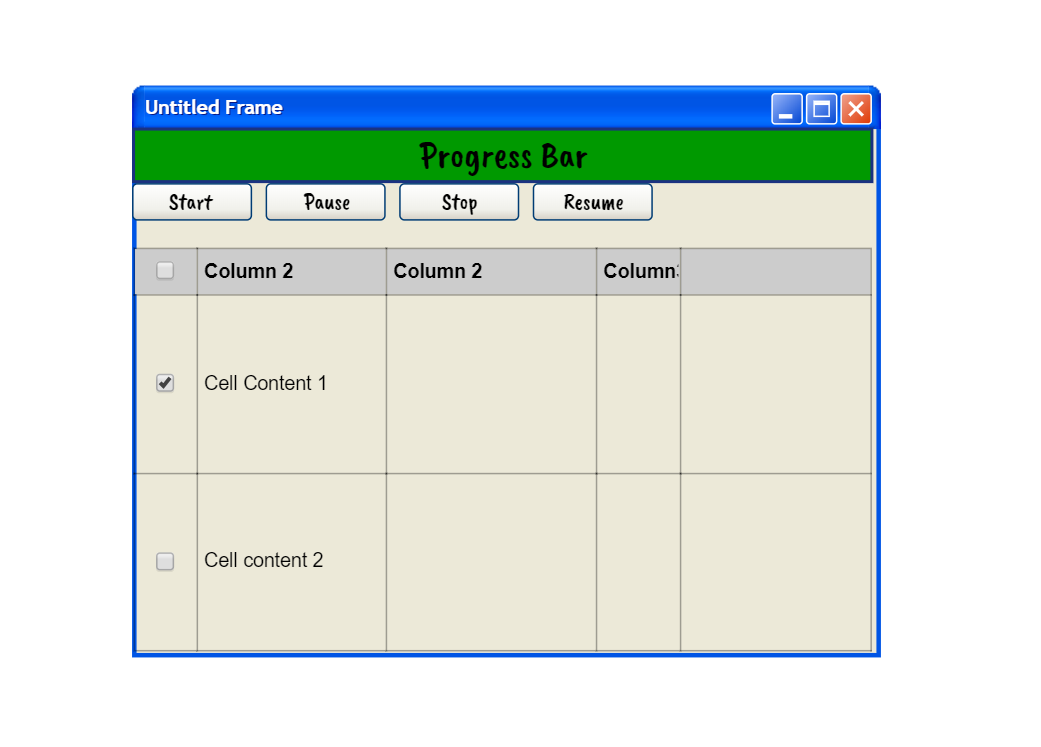
**Proposer Details**

| Group Number | *17* |
| --- | --- |
| Registration Number of Group Members | 2020-CS-134  2020-Cs-157 |

**Proposal Details**

|  |  |
| --- | --- |
| ***Project*** |  |
| Proposed Project Title | Pharmacy Scrapping |
| Executive Summary | In this project we will scrap medicine or pharma data from different websites.  Purpose is to automate data, record of sales and record of medicines.  This system is quite user-friendly. All the updates regarding the new drugs introduced in the market and other drugs related information can be found in the system.  We can sort data of a single column. There is a progress bar which will show the amount of data which is scraped. We can start, stop, Pause, Resume the scrapping. The user can choose specific algorithm from the given options. Data can be scraped up to 1 million entries. Attributes of entity will be at least 7. After sorting of column we can display time in milliseconds. Advanced filters for string columns will be implemented such as contains, end with, starts with etc. User can sort data of multiple columns at a time. User can search the scraped data for specific entry. One page of UI will display the list of chosen entity. |
| ***Business Case*** |  |
| Outline the business need for the project | *[This section to contain a clear articulation of the business need in the form of a statement that addresses the problem or opportunity. This statement should be no more than three or four sentences. Mention clearly which type of businesses will benefit from this project.]* |
| End user of the product | *[Clearly identify the real life domain and user that you are going to target]* |
| Motivation for Project | *[This section to contain a clear statement of motivation which drives you to this project]* |
| State the level of impact expected should the project proceed and implications of not proceeding | *[State whether the implementation would have an impact at an operational level and/or strategic level and state the impact(s) in 2-3 lines]* |
| ***Technical Details*** |  |
| Name of Entity | *[List the name of entity that you are going to scrap]* |
| Attributes of Entity  (Minimum seven attributes/rows can be increased) | |  |  |  | | --- | --- | --- | | *Name* | *Data Type* | *Description* | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |  |  |  | |
| Sample of Scrapping Source | ***Dawaii.pk:***    ***Sehat.pk:*** |
| Github Repository Link |  |
| Sorting Algorithms |  |
| |  |  | | --- | --- | | **Algorithm Name** | **Description(Each algorithm in 2-3 lines)** | | Selection Sort | In selection sort we start by finding the minimum value in a given list and move it to a sorted list. Then we repeat the process for each of the remaining elements in the unsorted list. | | Insertion Sort | Insertion sort involves finding the right place for a given element in a sorted list. So, in beginning we compare the first two elements and sort them by comparing them. Then we pick the third element and find its proper position among the previous two sorted elements. This way we gradually go on adding more elements to the already sorted list by putting them in their proper position. | | Merge Sort | This is a divide and conquer algorithm. In this algorithm we split a list in half and keeps splitting the list by 2 until it only has single element. Then we merge the sorted list. We keep doing this until we get a sorted list with all the elements of the unsorted input list. | | Bubble Sort | Simplest sorting algorithm. Iterates over the list, in each iteration it compares elements in pairs and keeps swapping them such that the larger element is moved towards the end of the list. | | Hybrid Sort | Hybrid Sort is a [hybrid](https://en.wikipedia.org/wiki/Hybrid_algorithm) [stable](https://en.wikipedia.org/wiki/Category:Stable_sorts) [sorting algorithm](https://en.wikipedia.org/wiki/Sorting_algorithm), derived from [merge sort](https://en.wikipedia.org/wiki/Merge_sort) and [insertion sort](https://en.wikipedia.org/wiki/Insertion_sort), designed to perform well on many kinds of real-world data. | | Quick Sort | In this algorithm we partition the list around a pivot element, sorting values around the pivot. In my solution I used the last element from the list as pivot value. Best performance is achieved when the pivot value splits the list in two almost equal halves. | | K-Select/Quick Select | Quick-select/K-Select is a selection algorithm to find the Kth smallest element in an unordered list. It is related to the quick sort sorting algorithm. | | Counting Sort | This algorithm does not do comparison between the elements. We use the mathematical properties of the integers to sort. We count how many time a number has come and store the count in the array where index is mapped to key’s value. | | Heap Sort | We create two segments of the list one sorted and one unsorted. In this we use heap data structure to efficiently get the max element from the unsorted segment of the list. Heap method uses recursion to get the max element at the top. | | Bucket Sort | Bucket Sort is a sorting algorithm that divides the unsorted array elements into several groups called buckets. Each bucket is then sorted by using any of the suitable [sorting algorithms](https://www.programiz.com/dsa/sorting-algorithm) or recursively applying the same bucket algorithm. | | Shell Sort | Shell Sort involves sorting elements which are away from each other. We sort a large sub-list of a given list and go on reducing the size of the list until all elements are sorted. | | Radix sort | Radix sort is one of the sorting algorithms used to sort a list of integer numbers in order. In radix sort algorithm, a list of integer numbers will be sorted based on the digits of individual numbers. Sorting is performed from least significant digit to the most significant digit. | | 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. | | |
| Searching Algorithms | *[List down the names of searching algorithms with description of each in 2-3 lines]* |
| Searching Filters for each data type | *[List down the details of searching filters that you will provide for each data type of attribute]* |
| Multi-Level Sorting | [Describe how you will provide multi-level sorting] |
| Any other features | [Describe details of any other feature that you want to implement, or any bonus task] |
| ***Interfaces for your project*** |  |
| |  |  |  | | --- | --- | --- | | UI Component Name | Type of UI component | Purpose of UI Component/Other details | | Label | Label | To ask user to write any word to sort or select the algorithm. | | Combo Box | Combo Box | It is a drop down box from which user will select the algorithm from which he/she want to work. | | Buttons | Button | There are two buttons on this UI one is cancel which is used to cancel the sorting and other is next and this helps to move to next UI page. | | Window Frame | Frame | Used here to enhance the quality of UI. | |  |  |  | | |



|  |  |  |
| --- | --- | --- |
| UI Component Name | Type of UI component | Purpose of UI Component/Other details |
| Progress Bar | Progress Bar | Progress bar showing the progress of tasks/ number of entities scrapped. |
| Table | Table | Table will shoe the entities which are sorted or to be sorted. |
| Buttons | Button | There are two buttons on this UI one is cancel which is used to cancel the sorting and other is next and this helps to move to next UI page. |
| Window Frame | Frame | Used here to enhance the quality of UI. |