**Car Details Platform**

**Project Supervisor**

Mr. Samyan Qayyum Wahla

**Group 01**

Shahzaib Irfan 2021-CS-07

Afraz Butt 2021-CS-12

Muhammad Hamza 2021-CS-41

Department of Computer Science

University of Engineering and Technology, Lahore, Pakistan

Table of Contents

[2 Project Details 4](#_Toc117946390)

[2.1 Project Title 4](#_Toc117946391)

[2.2 Project Statement 4](#_Toc117946392)

[2.3 Description 4](#_Toc117946393)

[2.3.1 Problem Overview 5](#_Toc117946394)

[2.3.2 Audience Motivation 5](#_Toc117946395)

[2.4 Requirements 5](#_Toc117946396)

[3 Technical Details 6](#_Toc117946397)

[3.1 Attributes 6](#_Toc117946398)

[3.2 Scrapping 7](#_Toc117946399)

[3.2.1 Why we use web scrapping? 7](#_Toc117946400)

[3.2.2 Websites 7](#_Toc117946401)

[4 Business Details 9](#_Toc117946402)

[4.1 Overview 9](#_Toc117946403)

[4.2 For Audience 9](#_Toc117946404)

[4.3 For Developers 9](#_Toc117946405)

[4.4 Motivation 9](#_Toc117946406)

[4.5 Conclusion 10](#_Toc117946407)

[4.6 Project Planning 10](#_Toc117946408)

[4.7 Developer 10](#_Toc117946409)

[4.8 Proposed UI Sample 10](#_Toc117946410)

[4.9 Task Division 12](#_Toc117946411)

[5 Algorithms 13](#_Toc117946412)

[6 Final Application 30](#_Toc117946413)

[6.1 UI Model 31](#_Toc117946414)

[6.1.1 Final UI Model 31](#_Toc117946415)

[6.1.2 Components of UI 31](#_Toc117946416)

[6.1.3 Changes in UI: 36](#_Toc117946417)

[6.2 Final Testing 37](#_Toc117946418)

[6.2.1 Insertion sort Using different attributes: 38](#_Toc117946419)

[6.2.2 Selection sort Using different attributes: 39](#_Toc117946420)

[6.2.3 Tim Sort: 42](#_Toc117946421)

[Using Mileage in Ascending Order: 42](#_Toc117946422)

[Using Year in Descending Order: 43](#_Toc117946423)

[6.2.4 Count Sort: 43](#_Toc117946424)

[6.2.5 Heap Sort: 45](#_Toc117946425)

[6.2.6 Searching Bar: 47](#_Toc117946426)

[6.3 Searching: 49](#_Toc117946427)

[6.3.1 Searching start as: 49](#_Toc117946428)

[6.3.2 End as: 50](#_Toc117946429)

[6.3.3 AND: 51](#_Toc117946430)

[6.3.4 NOT: 52](#_Toc117946431)

# Project Details

## Project Title

After a detailed discussion, we titled this project called **Car Details Platform.**

## Project Statement

Today, we have a huge variety of items in our daily lives. For example, in the case of mobile phones and cars, different manufacturers are offering different series of products based on their features. It is difficult for a buyer to select and optimise a product for his needs. Everyone wants to make his or her job easier. Everyone is attempting to save time. We are attempting to tackle their problem by collecting data from various sites and developing user-friendly software.

This software is intended to assist a customer in selecting the best automotive car for him or her. The user can easily search for a car that meets his or her specifications.

## Description

After an observation of this problem, our team designed a software, named as **Cars Details Platform,** to solve this problem. The primary goal of developing this software is to save the time of anyone looking to get a know-how of a vehicle. Many people spend a significant amount of time travelling to various websites or locations to inspect a vehicle. Our main goal now is to scrape data from many well-known websites and then apply some sorting and searching algorithms, as well as to test its time complexity,so the user can directly navigate to the desired ad.

Using this method, we will attempt to collect millions of data points based on worldwide standards. We employ several algorithms to alter every data after obtaining it from the internet. The user can arrange data in several ways, such as ascending or descending order. You can also check other types of data in various ways. The user can also test the speed of sorting by running different algorithms on various sorts of data..

The data for the automobiles is far too huge. As a result, entering data by a single person or a couple of people is challenging. It takes a few weeks to enter data. We opted to obtain this information from various websites based on cost and time constraints. We are employing a technology known as web-scraping for this purpose.

Searching is also a feature of this software. User can also search a car by its name.The algorithms are of insertions sort, merge sort, selection sort and bubble sort. The user will select any one algorithm and data will be sorted according to that algorithm. The user will then input the execute option after selecting the algorithm, and the data will be displayed in a table with the properties.

It will:

* Scrape data of multiple vehicles of different brands and series from many websites.
* Provide an option to sort lists depending upon different attributes.
* Provide the functionality of searching a particular vehicle.
* Allows selection of algorithms for sorting and accordingly shows time taken.

### Problem Overview

It is an internet and communication technology generation. Everyone is purchasing things from excellent online systems. Our goal is to approach men and women who are interested in purchasing a vehicle. If a person is looking to buy a car, he or she will look at a few websites before contacting the seller. The second strategy is to go visit the area where humans sell their autos. Time is very important, and these replies are time-consuming. The first is time spent viewing various and trustworthy websites, while the second is a physical visit that takes more time than the first. Now, our team is attempting to identify as many solutions as possible before implementing the best solution feasible.

### Audience Motivation

Life has become so rapid that we travel from one location to another in minutes and hours rather than days and months. Previously, we would cross a distance in days; now, we cover the same amount in hours. According to our everyday routine, we all want vehicles. Some people use multiple automobiles for various purposes (i.e. for personal use, for business etc).

## Requirements

|  |  |
| --- | --- |
| Programming language | Python 3 |
| Version | 3.10.1(2021-CS-07), 3.10.6(2021-CS-12 and 2021-CS-41 in conda venvs) |
| IDE for Python 3 | There are many IDE’s of Python. Some of them are given below:   * Spyder * Jupyter(Anaconda) * VS Code   From all these IDEs, 2021-CS-12 chose VS Code for development, while remaining members went with Spyder IDE. |
| Library | * pandas * Beautifulsoup * pyttsx3 * requests * datetime * webbrowser * sys * logging |

# Technical Details

## Attributes

|  |  |  |
| --- | --- | --- |
| **Name** | **Data Type** | **Description** |
| Name | String | It contains the name and some other important information of the vehicle.  For example,    In this picture, ‘Toyota Fortuner 2022 legender for sale’ is the name. |
| Year | Int | It tells us the model of vehicle of a specified entity. In the picture the year is “2022.” |
| Grade | Int | This occurs in imported vehicles. The grade indicates whether the car is 4 by 4, single powered, or in case of double imports, R grade. |
| CC | Int | It is the maximum engine capacity of vehicle that has been fixed by company. It contain that perticular information.  For example,    In this picture, “1500cc” is the cc. |
| Mileage | Int | It contains the information about the vehicle that how much kilometers the vehicle has been drived. It helps the uer to buy a suitable and better vehicle.    For example, in previous picture, this car drived 134,997 km. |
| Transmission | String | It contains the information about the vehicle is automatic or manual. |
| Link | href | It is the link of the ad of the entity, which will allow the user to navigate to it directly in the web. In this project, we have constructed this attribute in backend, as there is no representation of links in the ad. |
| Fuel | String | There are three types, one is petrol and second diesel, third is electric. This attribute contains that information in which user can know which type of fuel he will use after buying this vehicle.    In this picture, hybrid means the vehicle works on both diesel and gas. |

## Scrapping

### Why we use web scrapping?

Basically, web scrapping is used to get a large number of data in a single module which further can be used for different purposes and many operations can be performed on the scraped data. As many mobile applications are developed there main task is to first scraped the data form their website which further display to the audience on the user interface of the mobile applications.

### Websites

|  |  |
| --- | --- |
| PakWheels | **PakWheels** is the Pakistan first best online cars buy and selling forum.  **Front Display:**  **How entity looks:**  Picture above contains all the attributes which is scraped for the project.  Name, Year, Mileage, Engine Capacity, Transmission, Fuel Type  **No. Of vehicles:** approximately 200k to 300k+ vehicles. Around 280k scraped from this website.  **Link:** <https://www.pakwheels.com/> |
| Carvago | **Carvago** is an international website which completely inspect the used car for their customers and provided facility to buy and sell car worldwide.  **Front display:**    **How entity looks:**  Picture above contains all the attributes which is scraped for the project.  Name, Year, Mileage, Engine Capacity, Transmission, Fuel Type  **No. Of vehicles:** approximately 600k to 700k+ vehicles.  **Link:** <https://carvago.com/> |

# Business Details

## Overview

This module is designed on the base of the business point of view. As there are many applications that have their record at a specific location or on a specific hub. So, correspondingly this project is based on this idea that this project will be used as a record keeping application and as to performs different functions on the data with the passage of time. Data can be changed or can be updated.

As on of the above website which is discussed “PAKWHEELS” also using the same idea for their mobile based application. The data is fetched from website and shown to the audience.

## For Audience

This project is a great helping hand for those people who want to buy a car for themselves without any kind of disturbance and the fear of being scammed. So, for those people who want to buy a car from home or we can say that buying car online. A forum is provided they can see the cars details in which name, fuel type, transmission, year, grade of imported cars and also the link is provided so they can also view the price of their chosen car and yes if they want to filter something or they want to get an attribute in a sequence. These types of filters are also provided.

## For Developers

For developers there will be a bunch of interesting work to do. As the data is present they will performs different types of operations on the data with the passage of the time. As the cars are for buying or selling so they can insert, updated, delete and retrieved the data for the betterment of the application and can provide the facility to the audience to buy and sell the car easily without any fear of being scammed by the people and the dealers. Developers can also update the data correspondingly to the need of the audience.

## Motivation

As the world is shifted towards the technology and all the systems converted to online systems. So, this idea will be helpful for the business men who are used to do the business of the cars on the file system. They can change their business psychology to the online systems. As now a days, people used to search and buying the goods and facilities online. As, this is not helpful in making their business stronger but also it makes a big community of people for their business growth.

## Conclusion

This project has it own’s pros and cons but if the project is used on the base for which it is developed it will be a great helping hand for all the types of the audience to facilitate themselves and will be helpful for them in the automobile industry.

## Project Planning

## Developer

* Shahzaib Irfan(2021-CS-07)
* Afraz Butt(2021-CS-12)
* Muhammad Hamza(2021-CS-41)

## Proposed UI Sample

After a long discussion, A UI model was proposed in “DSA Mid Proposal Template”. Sample of UI model proposed is given below:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Dashboard Layout is provided.     |  |  |  | | --- | --- | --- | | UI Component Name | Type of UI component | Purpose of UI Component/Other details | | scrap\_progress\_bar | Progress Bar | Shows the details of scrap operation from the website. | | Pause | Button | Pause the scrap operation | | Resume | Button | Resumes the scrap operation | | Stop | Button | Stops the scrap operation | | URL input | Text Edit | Input the link for scrap | | Scrap | Button | Start the scrap operation | | X entities Scrapped | Label | Shows the details of entities scrapped | | Entity Count in the table X | Label | Entities count in the table | | X milliseconds as of now | Label | Time occurred in the sorting operation | | Widget Controls | Button | Close, Maximize and minimize the window | | Algorithms | Combo Box | Gives the list of sorting algorithms to choose from | | Ascend and descend | Radio Buttons | Order of sorting | | Sort | Button | Starts the sorting operation | | Column | Combo Box | Choose the column for search | | Contains | Text Edit | Contains the term for search | | Starts with | Text Edit | Term for search starts with | | Ends with | Text Edit | Term for search ends with | | AND | Radio Button + Text Edit | Composite Filter for search | | OR | Radio Button + Text Edit | Composite Filter for search | | NOT | Radio Button + Text Edit | Composite Filter for search | | Algorithm | Combo Box | List of Algorithms for searching | | Search | Button | Starts the search operation | | X milliseconds as of now | Label | Gives the time of search operation | | Table for entities | Table | Holds and displays Entities | | Table Header (x8) | Header (contains radio button) | Shows Name of attribute and radio button to check in multiple sorting | | Horizontal scroll bar | Scroll bar | Horizontal scrolling | | Vertical scroll bar | Scroll bar | Vertical scrolling | |

Some changes were made in UI model that will be discussed in the snippets of UI model below.

#### 

## Task Division

At the approval of our proposal, the tasks were divided. Following were the tasks that each group member was assigned. The important thing is that each group member did not exclusively do the tasks assigned to them, rather they helped other members too. This helped them grasp the entire project subdomains effectively.

|  |  |
| --- | --- |
| Task | Assigned to |
| Searching websites for scrapping | All group members |
| Scrapping of data | 2021-CS-41 |
| Making design | 2021-CS-12 |
| Project description | All group members |
| Business case | 2021-CS-41 |
| Executive summary | 2021-CS-12 |
| Technical details | All group members |
| Algorithm details | 2021-CS-7 |
| Implementation of design in Pencil Tool | 2021-CS-12 |
| UI Design and Backend Linking | 2021-CS-12(principally), CS-07(supplement) |
| Integration | All group members |
| Algorithm Implementation | 2021-CS-07(principally), 2021-CS-12(supplement) |

# Algorithms

|  |  |
| --- | --- |
| **Insertion Sort** |  |
| Description | The insertion sort is the doubly nested loop. This algorithms works well for smaller input. The algorithm makes an item of the list as pivot and the iterate towards the lower indexes of the list to check if the pivot is smaller than items backward. If the pivot is smaller than we swap the two items and so the case runs for the every item of list except the first one.  Insertion sort for smaller inputs is every efficient for smaller inputs say length of list be less than 43. This algorithm is far better than Bubble Sort.  In Cars Detail Platform, our target is to sort data of vehicles in different manners. We are using 7 attributes in our project so it not that easy to sort using Insertion sort. If the attribute is integer than sorting works well. But if the attribute is string type than it is not 100% sure that sorting will work as it is expected. |
| Pseudo Code | InsertionSort(array, type):      for j = 1 to array.length          key = array[j]          i = j-1  if(type ==’Ascend’):              while i>= 0 and array[i] > key:                  array[i+1] = array[i]                  i = i-1         if type == ‘Descend’:              while i>=0 and array[i] < key:                  array[i+1] = array[i]                  i = i-1            array[i+1] = key |
| Code in Python | def InsertionSort(array,start,end):  for i in range(start,end+1):  key = array[i]  j = i - 1    while key < array[j] and j >= start: #backwards linear scan  array[j + 1] = array[j]  j = j - 1    array[j + 1] = key  return array |
| Code for Different attributes | def InsertionSort(array,type,index):  print(index)  start = 0  end = len(array)  # print(end)  for i in range(start, end + 2 ):  key = array[index][i]  j = i - 1    if type == 'ascend':  while key < array[index][j] and j >= start: #backwards linear scan  array[index][j + 1] = array[index][j]  for k in range(0,8):  array[k][j + 1] ,array[k][j] = array[k][j] , array[k][j + 1]  j = j - 1    array[index][j + 1] = key    elif type == 'descend':  while key > array[index][j] and j >= start: # j > = start beacuse we r sorting from start to end as desired by user  array[index][j + 1] = array[index][j]  for k in range(0,8):  array[k][j + 1] ,array[k][j] = array[k][j] , array[k][j + 1]  j = j - 1    array[index][j + 1] = key  return array |
| Time Complexity Analysis | |  |  |  | | --- | --- | --- | | **Code:**  def insertionSort(array,type):      for j in range(1,len(array)):          key = array[j]          i = j-1          if(type == "Ascending"):              while i>-1 and array[i] > key:                  array[i+1] = array[i]                  i = i-1          else:              while i>-1 and array[i] < key:    array[i+1] = array[i]                  i = i-1          array[i+1] = key | **Cost:**  C1  C2  C3  C4  C5  C6  C7  C8  C9  C10  C11  C12 | **Time:**  n  n -1  n - 1  n- 1  ∑nj=2 tj  ∑nj=2 (tj -1)  ∑nj=2 (tj -1)  n-1  ∑nj=2 tj  ∑nj=2 (tj -1)  ∑nj=2 (tj -1)  n -1 | |  |  |  |   **For Ascending Order:**  T(n) = c1 + c2(n-1) + c3(n-1) + c4(n-1) + c5(∑nj=2 tj) + c6(∑nj=2 (tj -1)) + c7(∑nj=2 (tj -1)) + c12(n-1)  T(n) = c1 + c2(n-1) + c3(n-1) + c4(n-1) + c5( -1) + c6() + c7() + c12(n-1)  After further solving this equation, following equation will become  T (n) = (c5/2 + c6/2 + c7/2)n2 + (c1 + c2  + c4 + c5/2 - c6/2 - c7/2 + c12)n – (c2 + c4 + c5 + c12)  **T(n) = O(n2 )**  **For Descending Order:**  T(n) = c1 + c2(n-1) + c3(n-1) + c4(n-1) + c9(∑nj=2 tj) + c10(∑nj=2 (tj -1)) + c11(∑nj=2 (tj -1)) + c12(n-1)  T(n) = c1 + c2(n-1) + c3(n-1) + c4(n-1) + c9( -1) + c10() + c11() + c12(n-1)  After further solving this equation, following equation will become  T (n) = (c9/2 + c10/2 + c11/2)n2 + (c1 + c2  + c4 + c9/2 - c10/2 - c11/2 + c12)n – (c2 + c4 + c9 + c11)  **T(n) = O(n2 )**  **Best case:** Ω(n)  **Worst case:** O(n2)  **Average case:** Θ(n2) |
| Proof of Correctness | **Initialization:**  Before the loop j = 2 ⇒ A [1 .. j1] = A [1] which only contains the elements A [1 .. j1] (of which there is only one) and since there is only one element, they are trivial orderly .  **Maintenance :**  The outer for loop selects the A [j] element and places it correctly in A [1..j1] via the while loop. Since array A [1..j1] has been sorted, place element A [j] at the correct position produces A [1..j] in order (and contains the first j elements).  **Termination:**  The loop ends when j = n + 1 ⇒ A [1 .. j1] = A [1 .. (n + 1) 1] = A [1 .. n] which since the array remains sorted after each iteration gives A [1 .. n] is sorted when the loop ends and contains all of the original elements, the entire original array is sorted. |
| Three Strengths | 1. It is stable sort. 2. Efficient for small data set. 3. It become fast if data is already sorted because its best case is O(n) |
| Three Weakness | 1. For large data sets, it contains large time because its time complexity is O(n2). 2. It iterates array again and again, so time increases 3. It is not efficient as other algorithms 4. If we add new element it checks almost all emenents if element is larger. |
| Dry Run | |  |  |  |  |  | | --- | --- | --- | --- | --- | | **j** | **key** | **i** | **A[j]** | **A** | |  |  |  |  | **{**1,2,4,9,7,6**}** | 2 | 4 | 9 | 7 | 6 | | 2 | 2 | 1 | 2 | **{**1,2,4,9,7,6**}** | | 3 | 4 | 2 | 4 | **{**1,2,4,9,7,6**}** | | 4 | 9 | 3 | 9 | {1,2,4,9,7,6} | | 5 | 7 | 4 | 9 | {1,2,4,9,7,6} | | 5 | 7 | 3 | A[4] = 7 | {1,2,4,7,9,6} | | 6 | 6 | 5 | 9 | {1,2,4,7,6,9} | | 6 | 6 | 4 | A[5] = 7 | {1,2,4,7,7,6} | | 6 | 6 | 3 | A[4] = 6 | {1,2,4,6,7,9} | |

|  |  |
| --- | --- |
| **Selection Sort** |  |
| Description | The selection sort algorithm is a doubly nested loop algorithm. This is a comparison based algorithm. The list is divided in two two parts sorted and unsorted. The sorted part is on the left side and the unsorted part is on the right side of the list. As one can guess, initially the entire list will be unsorted part.    In the picture above, the largest number is selected from the unsorted array and placed in sorted part of the array. The process will be inverse as descending sort is used in picture above. In Cars Details Platform, our target is to sort data of cars in different manners. We are using 7 attributes in our project so it not that easy to sort using Selection sort. If the attribute is integer than sorting works well. But if the attribute is string type than it is not 100% sure that sorting will work as it is expected. |
| Pseudo Code | SelectionSort(array):        for j = 1to array.length - 1:          min = j          for i = j + 1 to array.length:                if array[i] < array[min]:                  min = i         exchange array[j], array[min] |
| Code in Python | def SelectionSort(array,start,end):  for i in range(start,end+1):  min\_ind = i  for j in range(i+1,end):  if array[j] < array[min\_ind]:  min\_ind = j  (array[i], array[min\_ind]) = (array[min\_ind], array[i]) |
| Code for different Attributes | def SelectionSort(array, type, index):  if type == 'ascend':  SelectionSort1(array[index], 0, len(array[index]) - 1, array)  elif type == 'descend':  SelectionSortDescending(array[index], 0, len(array[index]) - 1, array)  return array  def SelectionSort1(array,start,end, array2D):  for i in range(start,end):  for j in range(i + 1 , end):  if array[j] < array[i]: #minimum elements toRun  array[j] , array[i] = array[i] , array[j]  for k in range(0,len(array2D)):  array2D[k][j + 1] ,array2D[k][j] = array2D[k][j] , array2D[k][j + 1]    array[end] = array[end - 1]  return array |
| Time Complexity Analysis | |  |  |  | | --- | --- | --- | | **Code** | **Cost** | **Time** | | def selectionSort(array, size):        for j in range(size):          min = j          for i in range(j + 1, size):                if array[i] < array[min]:                  min = i            array[j], array[min] = array[min], array[j] | C1  C2  C3  C4  C5  C6 | n+1  n  ∑ni=j+1 (ti +1)  ∑ni=j+1 ti  ∑ni=j+1 ti  n |   T(n) = (n+1) +n + (∑ni=j+1 (ti +1) ) + (∑ni=j+1 ti ) + (∑ni=j+1 ti ) + n  T(n) = 3n + + (∑ni=j+1 (ti +1) ) + (∑ni=j+1 ti ) + (∑ni=j+1 ti ) + 1  After solving this equation we will reach at the following solution:  T(n) = O(n2)  **Best case:** Ω(n2)  **Worst case:** O(n2)  **Average case:** Θ(n2) |
| Proof of Correctness | **Initialization**:  The loop invariant is true at the beginning of the loop. The beginning of the loop is when j = i+1 and min = i, so the loop invariant states that array[i] is the smallest which is true.  **Maintenance:**  The invariant is preserved in the loop body (including the increment of the loop index j). We know that array[min] is the smallest of array[i..j] between lines 4 and 5. The body of the loop checks if array[j] is smaller than array[min] and sets min to j if it is, thus we know between lines 7 and 8 that array[min] is the smallest of array[i...(j+1)]. The loop invariant will still be true between lines 4 and 5 when j is incremented during the loop.  **Termination:**  As a result, we can deduce that the invariant is true once the loop is completed, which occurs when j=len (array). We know that on line 9 array[min] is the smallest from among L[i..len(array)] by replacing j=len(array) into the loop invariant. |
| Three Strengths | 1. It performs very well if data set is small. 2. It become fast if data is already sorted. 3. Too large space is not required because it is in-place algorithm and only one space required for an extra/temporal variable. |
| Three Weakness | 1. For large data set it is inefficient because time complexity will increase a lot. 2. It is unstable sort. 3. Other algorithms, such as quicksort, have better performance compared to the selection sort. |
| Dry Run | Let the input array A is   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 2 | 3 | 4 | 23 | 10 | 16 | 11 | 20 |   When=1:   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 2 | 3 | 4 | 10 | 23 | 16 | 11 | 20 |   After i=1:   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 2 | 3 | 4 | 10 | 11 | 16 | 23 | 20 |   After i=2:   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 2 | 3 | 4 | 10 | 11 | 16 | 23 | 20 |   After i=3:   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 2 | 3 | 4 | 10 | 11 | 16 | 20 | 23 |   After i=4:   |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 2 | 3 | 4 | 10 | 11 | 16 | 20 | 23 | |

|  |  |
| --- | --- |
| **Merge Sort** |  |
| Description | The Merge Sort is a divide and conquer algorithm which works by recursively dividing list in two equal sub-parts. Then the sub-parts are sorted into the function main list by the help of comparison.    In Cars Detail Platform, we used this algorithm because it is a very efficient algorithm. This algorithm sorts the integer type data types efficiently. But the cost for sorting list of alphabets can be much greater than normal.  It is an efficient algorithm. It sort data set with average time complexity O(nlog n ). |
| Pseudo Code | Merge(arr, p, q, r):      n1 = q - l + 1      n2 = r - q      L = array of size n1      R = array of size n2      for i = 0 to n1          L[i] = arr[l + i]      for j = to n2          R[j] = arr[q + 1 + j]      i = 0      j = 0      k = l      while i < n1 and j < n2:          if L[i] < 0              arr[k] = L[i]              i =i+ 1          else              arr[k] = R[j]              j =j+ 1          k = k + 1  MergeSort(arr, p, r):      if p < r          q = p+(r-l)/2          mergeSort(arr, p, q)          mergeSort(arr, q+1, r)          merge(arr, p, q, r) |
| Code in Python | def MergeSort (array, start, end):  if start < end: # check for length of array not equal to 1  q = (start + end) // 2 # // is used to convert answer to floor directly  MergeSortDescending(array, start, q) # first half of array sliced recursively  MergeSortDescending(array, q + 1, end) # second half of array sliced recursively  MergeDescending(array, start, q, end)      def Merge (array, p, q, r):  n1 = q - p + 1 # counter variable to copy array  n2 = r - q # counter variable to copy array  # L, R = [], [] # python initiallization  # for i in range(0,n1):  # L.append(array[p+i]) # copying L side of array  # for j in range(0,n2):  # R.append(array[q+j+1]) # copying right side of array  L = array[p:q + 1] # tried to lesser the total time by slicing but not worked properly  R = array[q+1:r+1]  L.append(-(math.inf)) # sys.maxize gives maximum integer we r using as a sentinel  R.append(-(math.inf))    if type(L[0]) == str:  i, j = 0, 0  for k in range(p, r + 1):  if str(L[i]) >= str(R[j]):  array[k] = L[i]  i += 1  else:  array[k] = R[j]  j += 1  else:  i, j = 0, 0  for k in range(p, r + 1):  if (L[i]) >= (R[j]):  array[k] = L[i]  i += 1  else:  array[k] = R[j]  j += 1 |
| Code for different attributes | def mergeSort(arr,type,index):  copy = arr[index].copy()  start = 0  end = len(arr) - 1    if type == 'ascend':  MergeSortAscnding(arr[index] , start, end)  elif type == 'descend':  # end -= 1  MergeSortDescending(arr[index] , start, end)  # print(arr[index])  # print(copy)    for i in range(len(arr[index])):  inx = copy.index(arr[index][i])      for k in range(8):  if k != index:  arr[k][inx] ,arr[k][i] = arr[k][i] , arr[k][inx]    return arr |
| Time Complexity Analysis | **Best case:** Ω(n log n)  **Worst case:** O(n log n)  **Average case:** Θ(n log n) |
| Proof of Correctness | **Initialization**:  We have k = p prior to the first iteration of the loop, hence the subarray A[p... k - 1] is empty. Since I = j = 1, both L[i] and R[j] are the smallest items of respective arrays that have not been cloned back into A, this empty subarray includes the k - p = 0 smallest elements of L and R.  **Maintenance:**  Let us first assume that l[i] = R[j] so that each iteration maintains the loop invariant. The smallest element not yet copied back into A is L[i]. Because A[p... k - 1] contains the k - p smallest items, the subarray A[p... k] will include the k - p + 1 smallest elements after line 14 duplicates L[i] into A[k]. The loop invariant is re-established for the following iteration by incrementing k(in the for loop update) and i(in line 15). If, instead, L[i] > R[j], lines 16-17 take the necessary steps to keep the loop invariant.  **Termination:**  k = r + 1 when the programme ends. The subarray A[p... k - 1], which is A[p... r], comprises the k - p = r - p + 1 smallest items of L[1... n1 + 1] and R[1... n2 + 1], in sorted order, thanks to the loop invariant. Together, the arrays L and R have n1 + n2 + 2 = r - p + 3 items. All but two of the largest items have been cloned back into A, and the sentinels are the two largest elements. |
| Three Strengths | 1. It is stable algorithm. 2. It contains similar time because its time complexity if same i.e. n logn. 3. It is faster than insertion and selection sort for huge data sets. |
| Three Weakness | 1. It is slower than other sorting algorithms if there is a small data set. 2. No of steps and time is same even the array is sorted or unsorted. 3. More memory is required because this method splits array recursively and divides it in two arrays. So more memory is required if data is increasing or no of recursive call will increase. |
| Dry Run | |  |  |  | | --- | --- | --- | | **Unsorted list** | **Divide** | **Sorted list** | | {12,23,2,43,51,35,19,4} | {12,23,2,43}  {51,35,19,4} | {} | | {12,23,2,43}  {51,35,19,4} | {12,23} {2,43}  {51,35} {19,4} | {} | | {12,23} {2,43}  {51,35} {19,4} | {12,23} {2,43}  {35,51} {4,19} | {12,23} {2,43}  {35,51} {4,19} | | {12,23} {2,43} {35,51} {4,19} | {2,12,23,43} {4,19,35,51} | {2,12,23,43} {4,19,35,51} | | {2,12,23,43} {4,19,35,51} | {2,4,12,19,23,35,43,51} | {2,4,12,19,23,35,43,51} | | {} | {} | {2,4,12,19,23,35,43,51} |   **Unsorted array:** {12,23,2,43,51,35,19,4}  After applying merge sort:  **Sorted array:** {2,4,12,19,23,35,43,51} |

|  |  |
| --- | --- |
| **Bubble Sort** |  |
| Description | The Bubble Sort algorithm is a comparison based algorithm. The comparison is made on adjacent cells and the unsorted items are placed on their appropriate places. This algorithm is a doubly nested loop algorithm. This algorithm works in n square time.  This sorting alogorithm is not used in professional workplaces as it is not efficient and takes too much time to even an array of 10000 items. However, this algorithm is used for the study purposes as it assists the students to learn algorithms because it very simple. |
| Pseudo Code | BubbleSort(array):    for i = 1 to array.length      for j = 1 to array.length - i - 1        if array[j] > array[j + 1]          exchange array[j], array[j+1]  return array |
| Code in Python | def BubbleSort(array,start,end):  for i in range(len(array) - 1):  for j in range(0, len(array) - i - 1):  if array[j] > array[j + 1]:  array[j] , array[j+1] = array[j+1],array[j]      return array |

|  |  |
| --- | --- |
| **Quick Sort** |  |
| Pseudo Code | Partition(array , p, r):      key = array[r]      i = p-1      for j = p to r          if array[j] < key              i = i+1              exchange array[i], array[j]     exchange array[i+1] , array[high]      return i+1  QuickSort(array,low , high)      if low < high          q = partition(array,low,high)          quickSort(array,low,q-1)          quickSort(array,q+1,high) |
| Code in Python | def QuickSort (A, p, r):  if p < r:  q = Partition (A, p, r)  QuickSort (A, p, q - 1)  QuickSort (A, q + 1, r)  return A  def Partition (A, p, r):  x = A[r]  i = p - 1  for j in range(p, r):  if A[j] <= x:  i += 1  A[i], A[j] = A[j], A[i]  A[i + 1], A[r] = A[r], A[i + 1]  return i + 1 |

|  |  |
| --- | --- |
| **Heap Sort** |  |
| Description | Heap sorting is the type of algorithm in which comparision based sorting is used. In this type of sorting one value of array is taken pivot and sorting is done around that pivot. Then Recursive call is applied on both sides of the pivot.  POINTS TO BE NOTED:  Heap sort is a type of sorting which is used in-place.  Heap sorting is not stable algorithm but can be made stable.  **BEST CASE:** The best case for heap sorting is O(n *lg* n).  ADVANTAGES: Heap sorting is easy to understand and is simple. It is more effective then any other sorting and is time saving.  **EXAMPLE:**  23,6,43,1,78,9,87,3 In the given example by using heap sorting the biggest value of array is taken as the reference point and is kept at the top and then others value are checked according to the reference point. The given example after heap sorting is as follow  1,3,6,9,23,43,78,87  **PSEUDOCODE:**  HeapSort(A):  Build\_MaxHeap(A)  For i : 1 to n do:  Exchange first and last elements  MaxHeapify(A,I,n-1) |

|  |  |
| --- | --- |
| **Counting Sort** |  |
| Description | The Counting sort algorithm is basically a count of the maximum number in the list. An additional list is created whose length is the maximum number in the list. The running time of this algorithm is big oh of n.  **WORST CASE PERFORMANCE**: The worst case performance of counting sorting is O(n)  **WORST CASE SPACE COMPLEXITY:**  The worst case space complexity of counting sorting is O(n).  **TIME COMPLEXITY:**  The time complexity of counting sort is O(n).  **Pseudo Code:**  CountingSort(A, B, k):  let C[0…k] be a new array  for i = 0 to k:  C[i] = 0  for j = 1 to A.length:  C[A[j]] = C[A[j]] + 1  for i = 1 to k:  C[i] = C[i] + C[I – 1]  for j = A.length downto 1:  B[C[A[j]]] = A[j]  C[A[j]] = C[A[j]] – 1  **Python Code:**  def CountingSortDescending(A):  maxA, minA = MaxElement(A), MinElement(A)  C = [0] \* ((maxA - minA) + 1)  B = [0] \* len(A)  for i in range(len(A)-1, -1, -1):  C[A[i] - minA] += 1  for i in range(len(C) - 1, 0, -1):  C[i - 1] += C[i]  for i in range(0, len(A)):  B[C[A[i] - minA] - 1] = A[i]  C[A[i] - minA] -= 1  return B  def MaxElement(A):  return max(A)  def MinElement(A):  return min(A)  **STABILITY:** Counting sort is an stable type of sorting algorithm in which the space complexity is O(n).  **CONCLUSION:** Counting sorting is used in daily life as it is stable as well as it is an accurate type of sorting algorithm. |
|  |  |

|  |  |
| --- | --- |
| **Tim Sort** |  |
| Description | Tim Sort is a stable , in place sorting algorithm Named after Tim Peters, It works by dividing the input array into small blocks(called MinRuns). On these small blocks, insert Sort is used. Afterwards, these small blocks are merged back into the original array using Merge Sort Function. It is best known as the sorting algorithm Python uses for its sort() and sorted() functions.  **AVERAGE TIME COMPLEXITY:** The average time complexity of tim sort is near to O(n *lg* n).  **BEST TIME COMPLEXITY:** The best time complexity of tim sort is O(n).  **WORST TIME COMPLEXITY:** The worst time complexity of shell sorting is O(n *lg* n).  **EXAMPLE:**  For an array of 64 integers, Tim Sort will first divide the array into two blocks of 32 integers, sort them using Insertion Sort, and then merge back into 64.  **PSEUDOCODE:**  Function TimSort(Array,start,end):  minRuns = 32  for i:1 to Array.length with increment minRun, do:  end = min(i-1 + n-1, n-1)  insertionSort(Array,start,end)  s = minRun  while S < Array.length do:  for j:1 to n with increment of 2s do:  mid = min(j-1 + s -1, n-1)  Merge(Array,start,mid,end)  s = s \* 2  end function  Function InsertionSort(Array,start,end):  for i in range(start + 1,end+1):  key = array[i]  j = i - 1    while key < array[j] and j >= start: #backwards linear scan  array[j + 1] = array[j]  j = j - 1    array[j + 1] = key  return array  **Python Code:**  def TimSort(array,start,end):  min\_run = 70 #Blocks of smallest arrat dat is effeciently sorted using Insertion Sort  n = len(array)  for i in range(0,n,min\_run):  end = min((i + n - 1) , (n - 1))  insertionSortTim(array,i,end)  s = min\_run  while s < n:  for j in range(0,n,s \*2):  mid = min((n - 1), (j + s - 1)) #merge function midpoint getting, n-1 is for when formula value exceeds array bounds  if s < mid:  Merge(array,start,mid,end)  s = s \* 2  return array  **ADVANTAGES:** Tim Sort does not use any additional auxiliary space.  **CONCLUSION:** Tim Sort is useful in many real world data. |

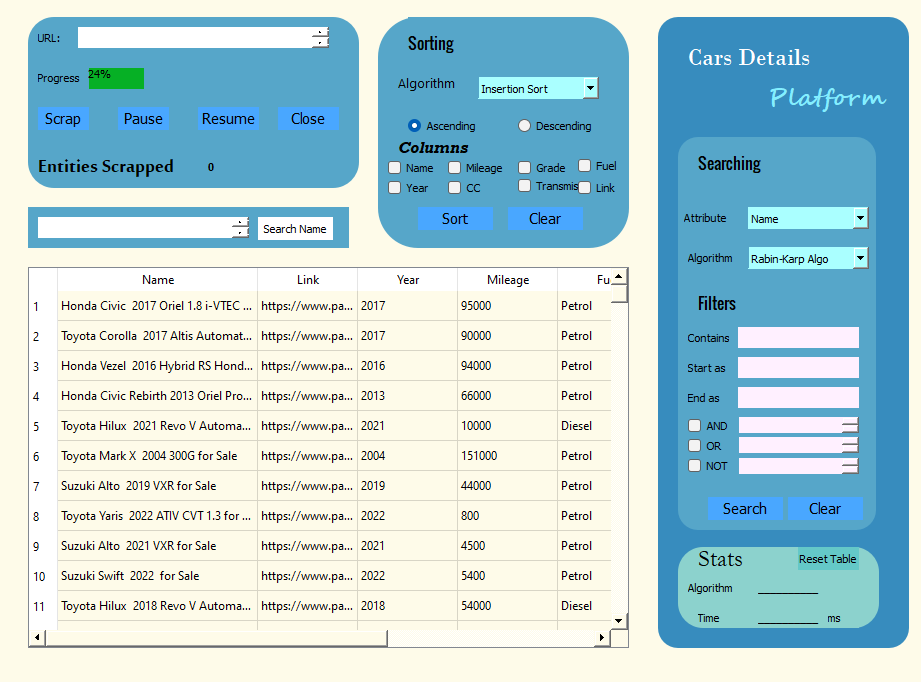
# Final Application

## UI Model

As discussed earlier, a UI model was made by us with some changes. This is our final UI model, linked with backend.

### Final UI Model

A rendering of Application is given below.



### Components of UI

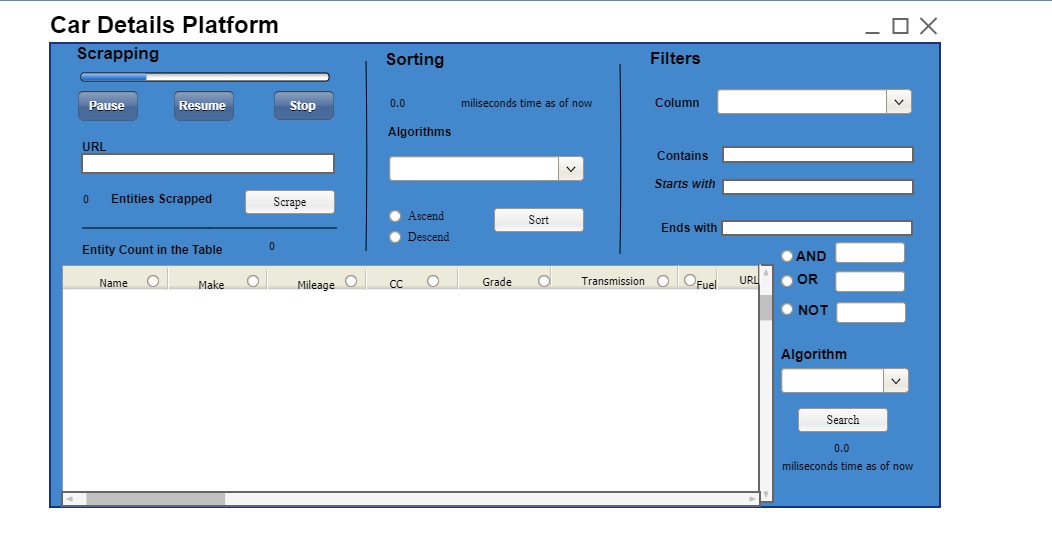
The components of final model are given below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Component Name** | **Component Type** | | **Object Name** | **Purpose with snapshot** |
| **Scrapper** | | | | |
| Cars Details Platform | Label | | label and label\_2 | It acts a logo of the application. |
| Scrap | PushButton | | btn\_Scrap | Starts scrapping in multi threaded environment. |
| Close | PushButton | | Btn\_close | It closes the scrapping thread, and updates the value of label of entities scrapped. |
| Pause | PushButton | | Btn\_pause | It pauses scrapping. |
| Resume | PushButton | | Btn\_resume | It resumes scrapping operation, from where it was left. |
| Name | Plain text Edit | | txt\_car\_contains | A traditional, javascript inspired searchbar which shows by name searching. |
| Progress Bar | progressBar | | progressBar | It will show percentage of data scrapped from web out of 100k. |
| **Statistics Widget** | | | | |
| Algorithm Name | label | | Lbl\_algo\_stat | It will show the name of selected algorithm for operation, searching and sorting. |
| Time Taken | label | | Lbl\_time\_stat | It will show time in milliseconds taken for algorithm execution, for either searching or sorting. |
| Reset Table | PushButton | | Btn\_reset | It resets the table back to its initial stages, in the unsorted and unsearched manner. |
| **Sorting** | | | | |
| Columns | CheckBoxes (x7) | Col\_name,col\_mileage,  col\_grade,col\_year  ,col\_fuel,col\_cc,  col\_transmission,  col\_link | | It is used to select attribute as a base or key for sorting. |
| Algorithm | ComboBox | Combo\_algo\_sort | | It selects the algorithm for sorting. |
| Modes | Radio Buttons | Radio\_ascend, radio\_descend | | It selects the mode i.e. Ascending or Descending for sorting. |
| Sort | Push Button | Btn\_sort | | It starts the sorting process. |
| **Search Statistics** | | | | |
| Attribute | ComboBox | Combo\_attribute\_search | | It selects the attribute to be searched, which acts as a key for searching. Basically, It is the column. |
| Algorithm | ComboBox | Combo\_algo\_Search | | It selects the algorithm for searching. |
| Contains | textedit | Txt\_col\_contains | | We can write here the text that we want to search from selected attribute. It even acts without filters. |
| Start as | Textedit | Txt\_start\_search | | It inputs the text which we want to start as. |
| End as | textedit | Txt\_end\_search | | It inputs the text which we want the target string to end as. |
| AND | CheckBox, text edit | Radio\_and, txt\_And | | The checkbox acts as state manager. The text input takes a pattern, which we want to check AND condition as a Boolean logic. |
| OR | checkbox,textedit | Radio\_or,txt\_or | | As above, but Boolean OR logic. |
| NOT | checkbox,textedit | Radio\_not, txt\_not | | As above, but Boolean NOT logic. |
| Clear | pushButton | Btn\_clr\_search | | It clears the searching filters. |
| Search | pushButton | Btn\_search | | It will search the text that we have entered in the “editLine” , just before the Button, from selected attribute using selected algorithm. |
| **Table** | | | | |
| Snapshot | tableWidget | table | | It is used to display all data. |
| Snapshots |  | | | |

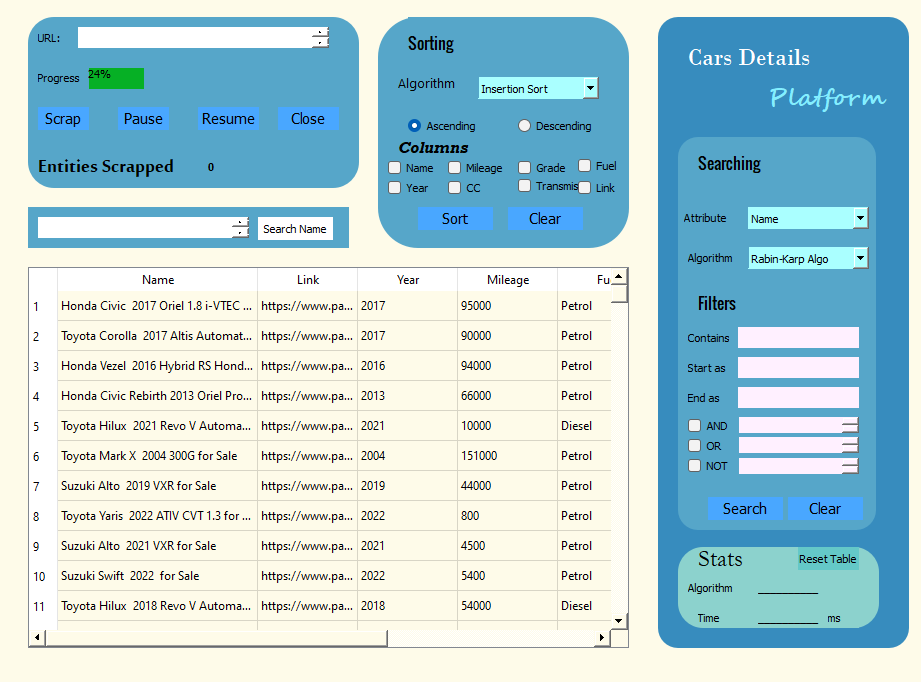
### Changes in UI:

The Base structure of UI remained same. That is, above table widget, scrapper and sorters were there. And to its side was the searcher. The difference in final rendering is that we used widgets to render a viewable side to the aspect. Components used were modified according to the objections raised by the audience in our presentation of proposal.

Base UI rendered in documentation:



Final UI model:



## Final Testing

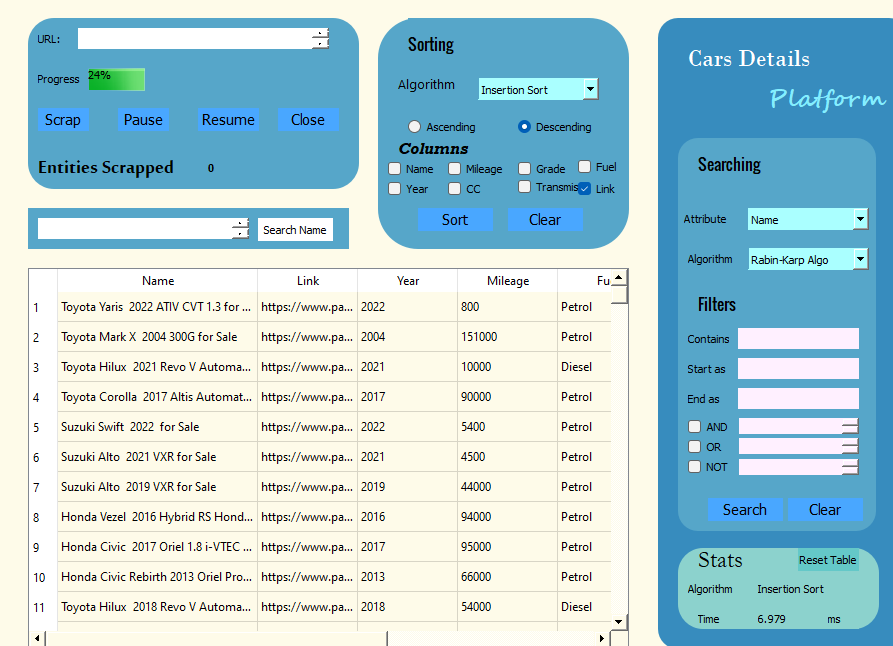
We will use random attributes on each sorting method to best judge its effectiveness. The data used for this demonstration is 270 entities, although in reserve csv one million data is also available.

### Insertion sort Using different attributes:

**Using Name in Ascending order:**



**Using Link in descending mode:**

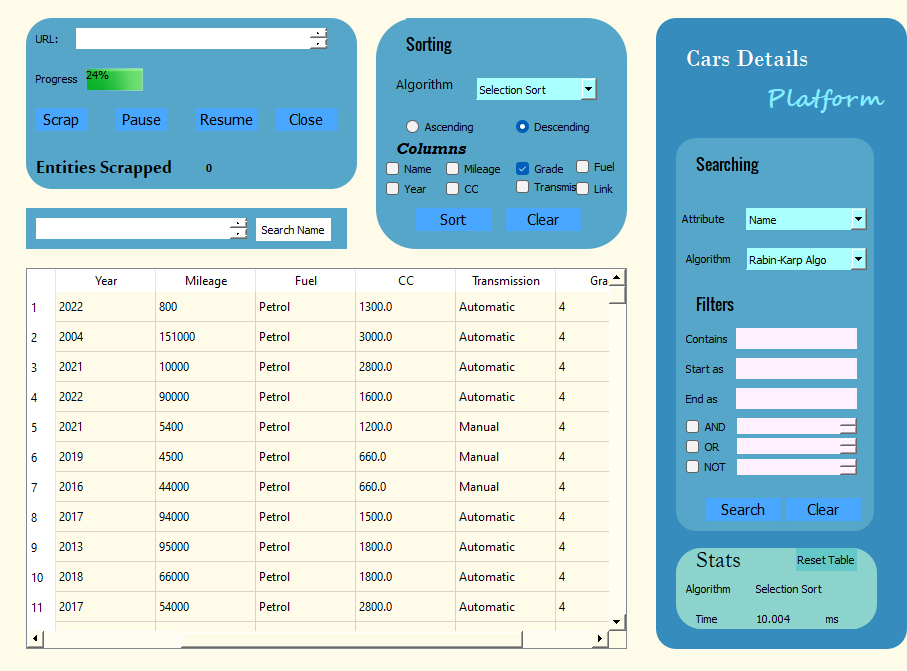


### Selection sort Using different attributes:

**Using Fuel in Ascending order:**

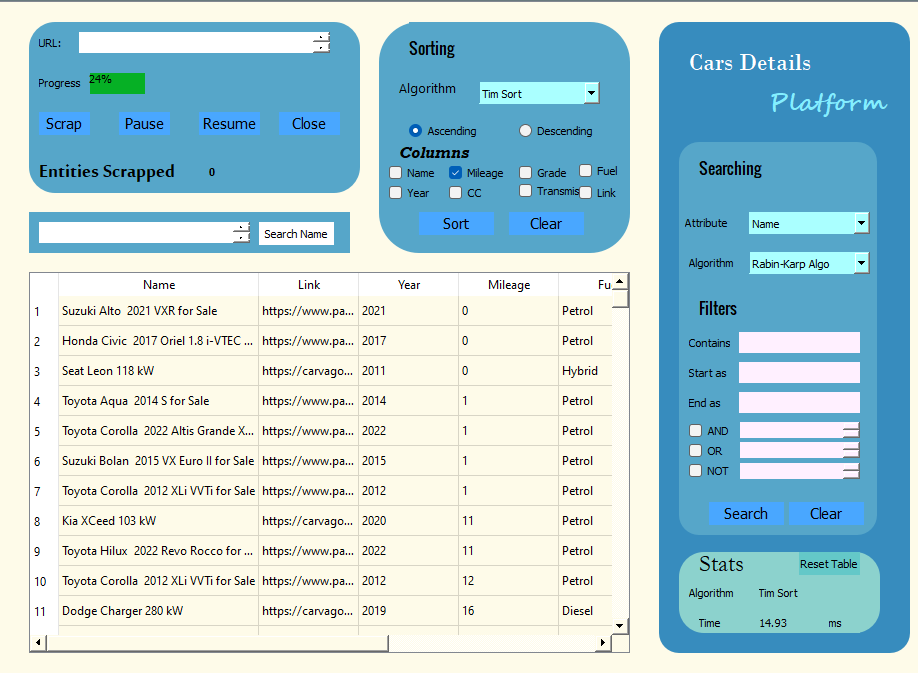


**Using Grade in Descending order:**



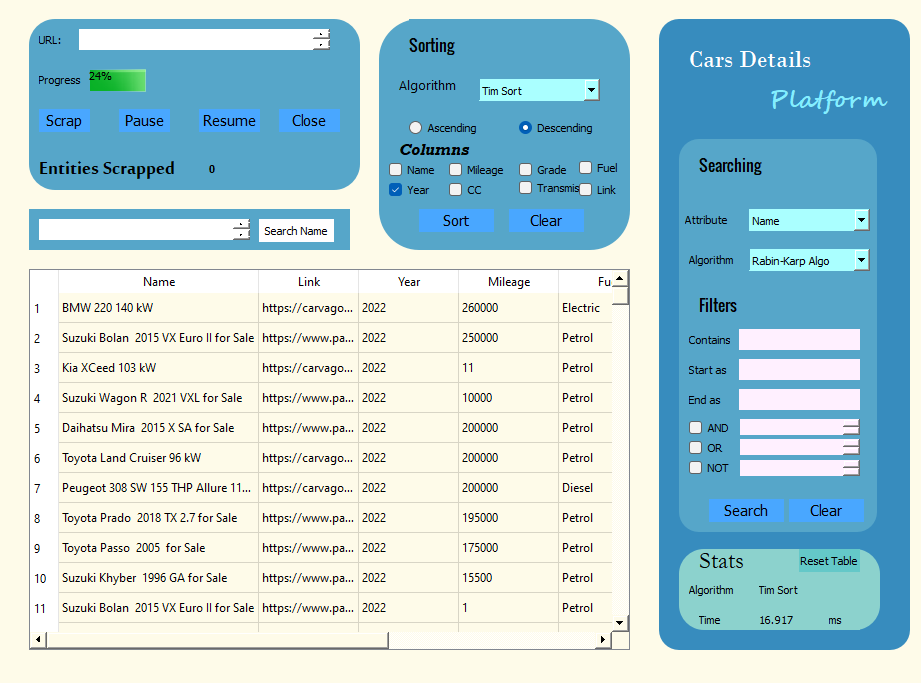
### Tim Sort:

### Using Mileage in Ascending Order:



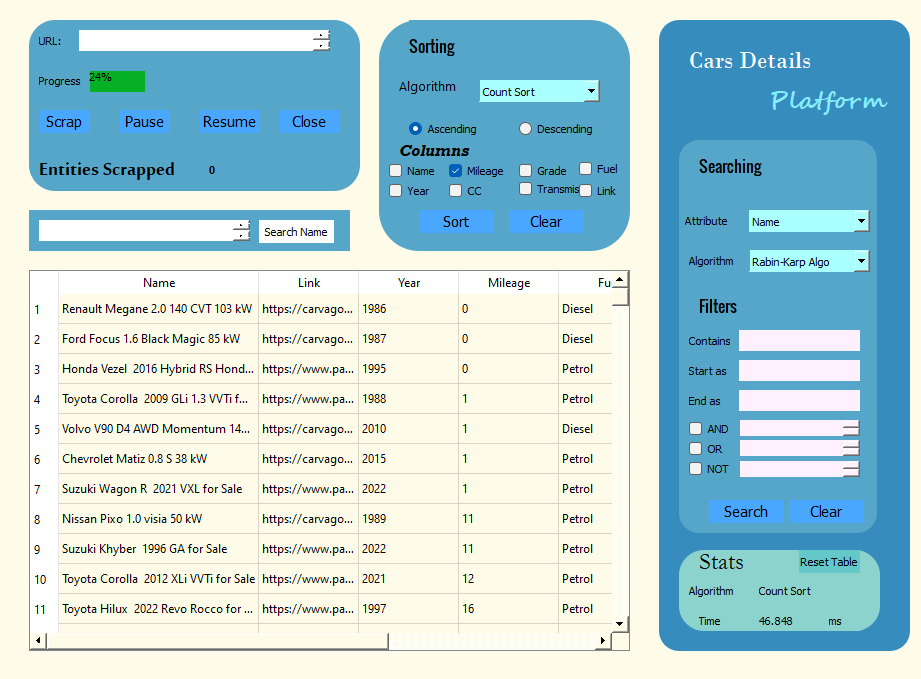
### Using Year in Descending Order:

### 



### Count Sort:

**Using Mileage in Ascending Order:**

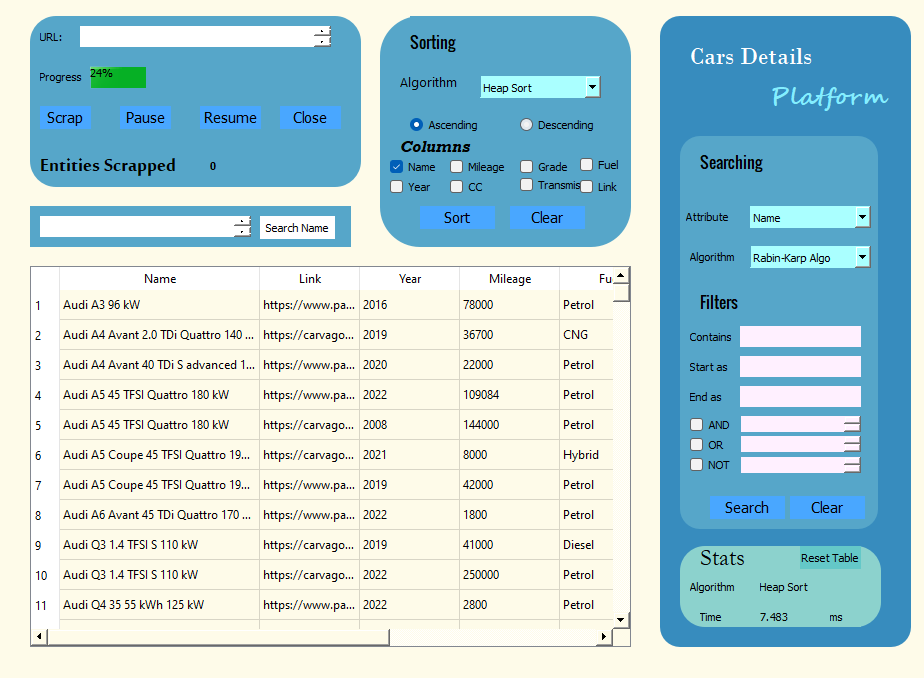


**Using Year as Descending Filter:**

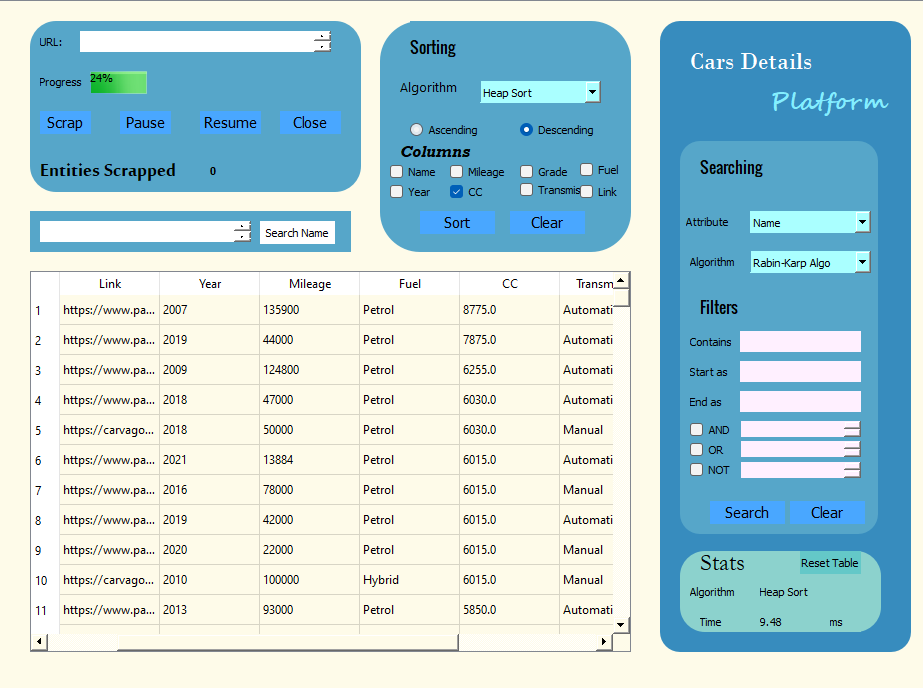


### Heap Sort:

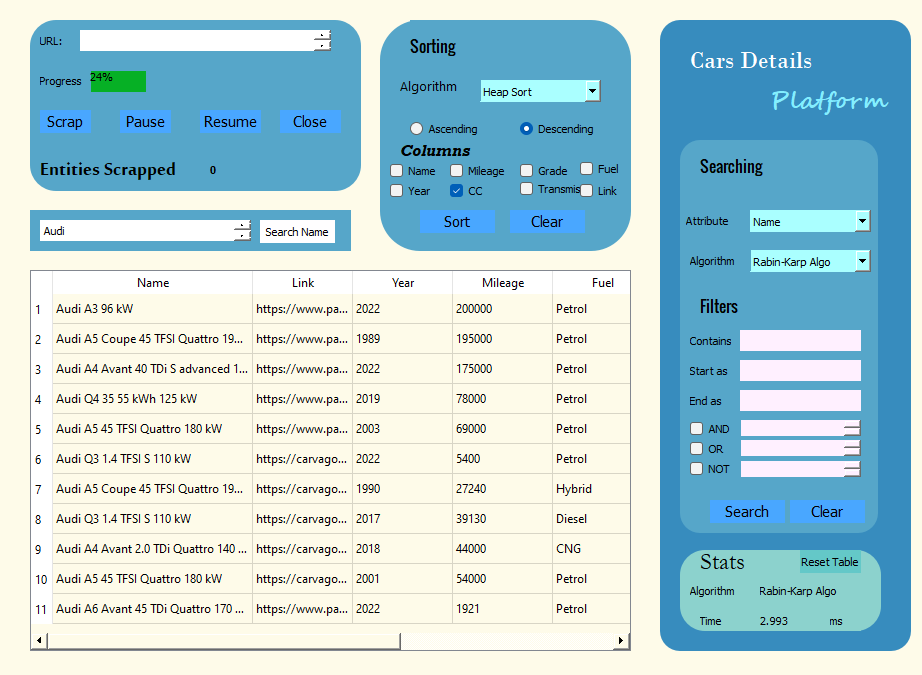
**Using Name as ascending Filter:**



**Using CC as descending Filter:**

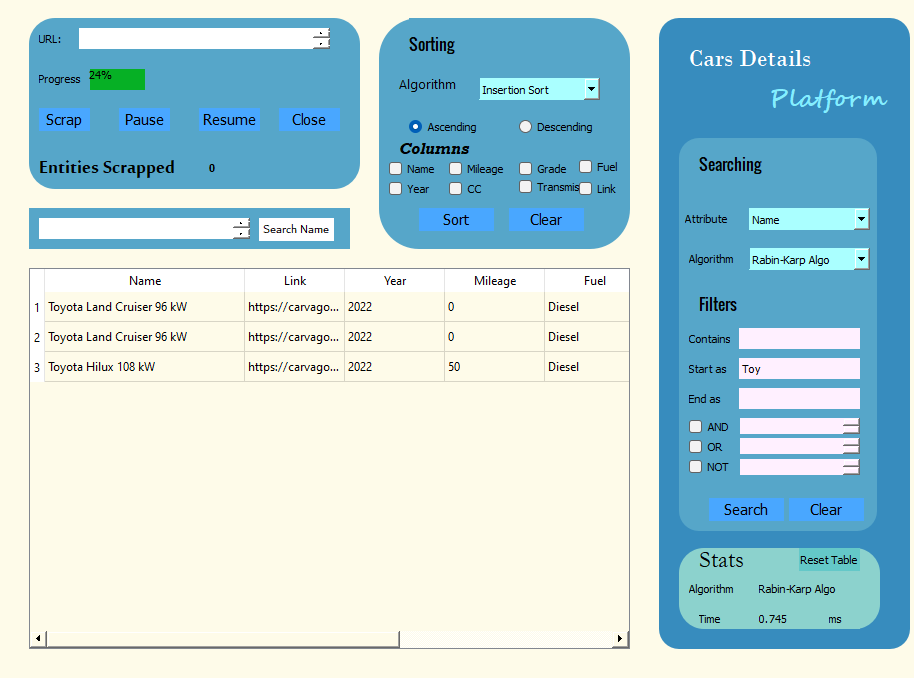


### Searching Bar:

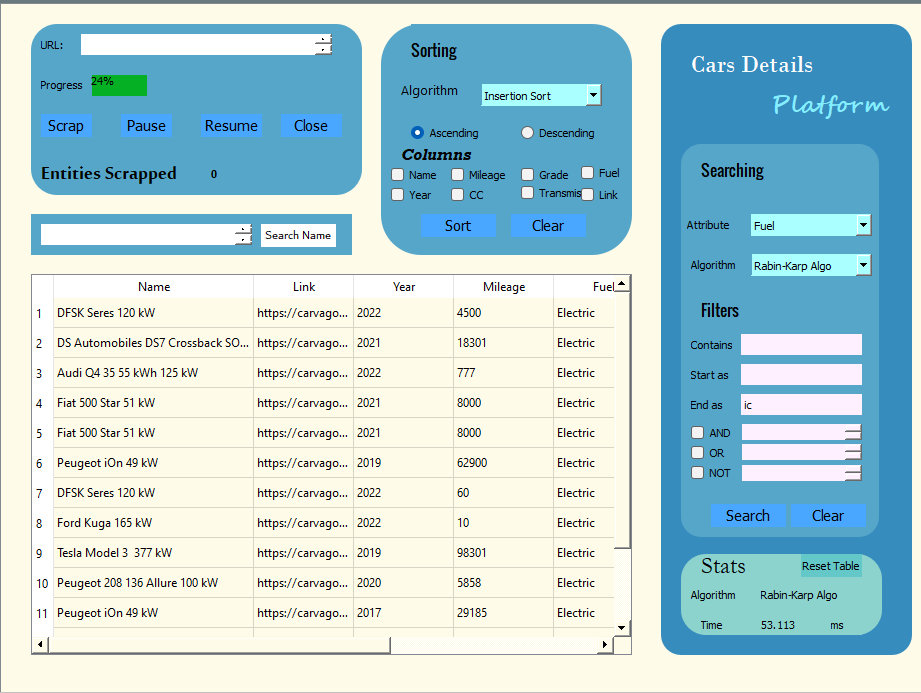


## Searching:

### Searching start as:



### End as:



### AND:

Graphical user interface, application

Description automatically generated

### NOT:

Graphical user interface, application

Description automatically generated