Mid Term Project Report



Auto Vehicle

(Data Scraping and Management)

2020-CS-99

2020-CS-74



A detailed report submitted in part fulfilment of the Project

**Degree: BS in Computer Science**

**Supervisor:** Sir Samyan Wahllah

Department of Computer Science

University of Engineering and Technology, Lahore

**About Developers:**

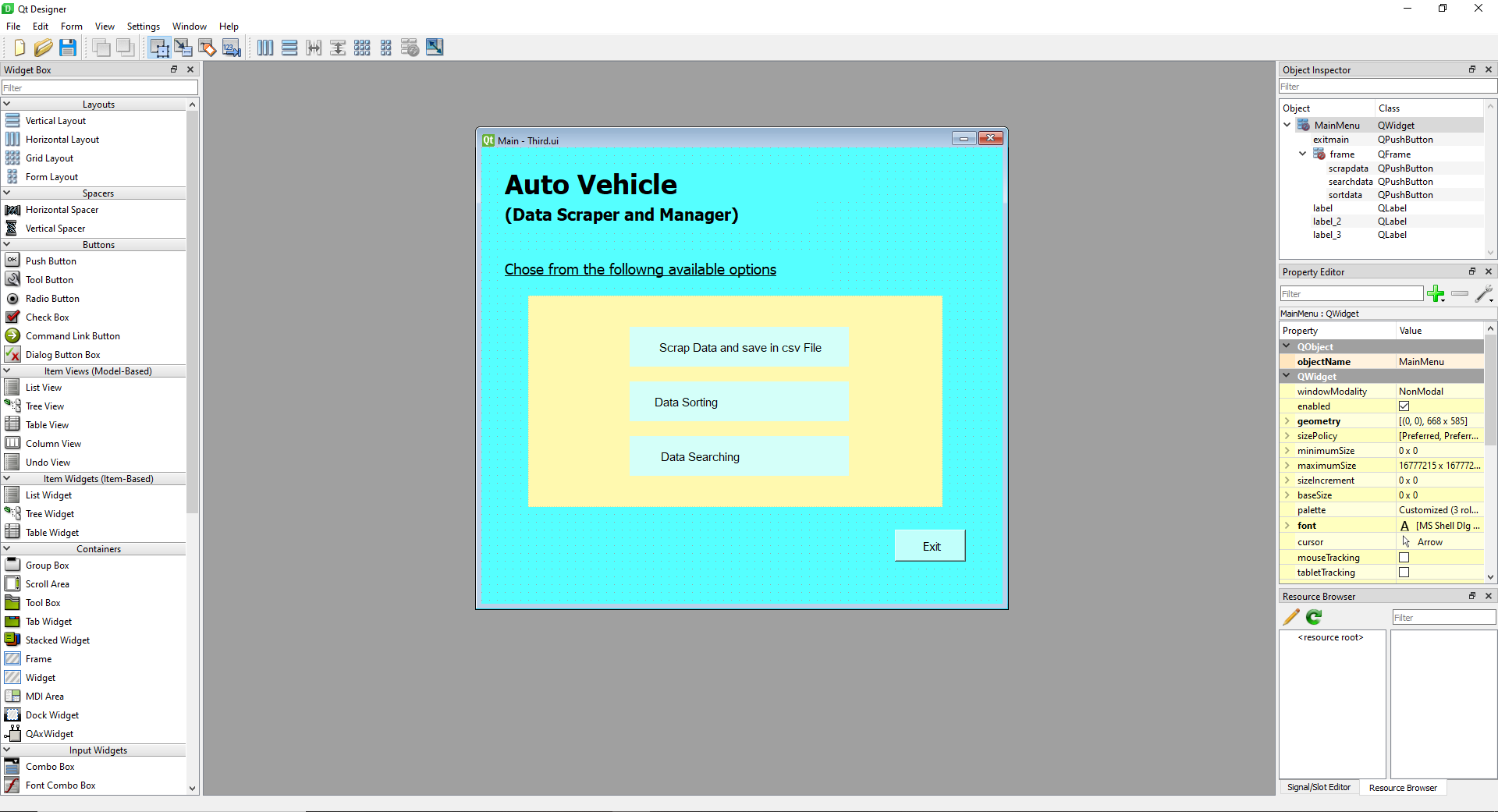
The project has been developed by Students under registration 2020-CS-99 and 2020-CS-74. All rights reserved to them. The project is powered and featured by Duko Community Ltd. Main idea of the project is to make it easy for the user to search about the desired cars available for sale on the internet in one single program rather than going and searching various websites.

**About Project Development:**

The project was developed by keeping the idea in mind that users should be befitted when using this application. The whole project is being developed in Python which is one of the most trending and powerful coding languages. To develop the application 2 types of Development environments were used:

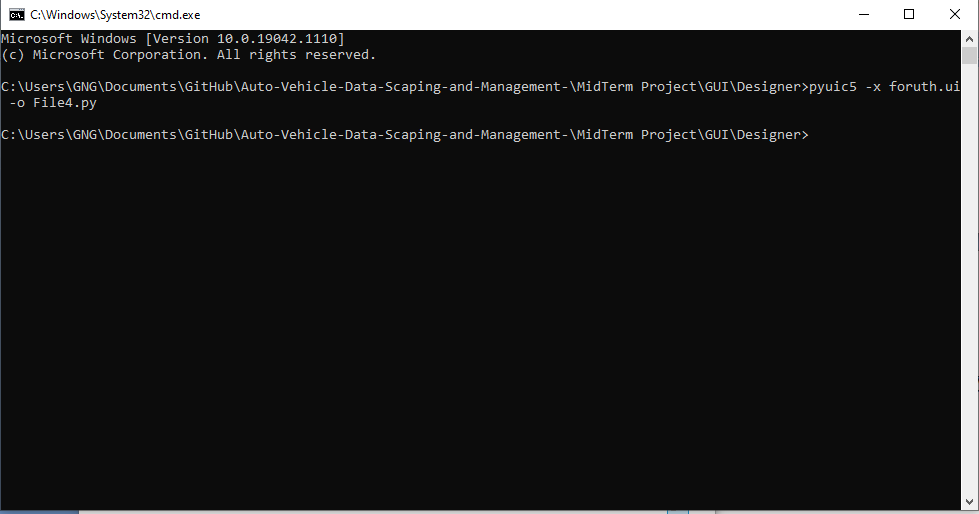
* Visual Studio Code
* Pyqt5(Designer)

The GUI (Graphic user interface) was developed in pyqt5 (Designer) as shown:



After designing the UI the UI was then converted to python file using command prompt and using a simple command:

pyuic5 –x file.ui –o File.py



The file was then opened in Visual studio code and then edited to make the GUI run.

**Steps in Developing Application:**

* **Step1:**

The first step in developing the project was downloading and integrating the tools required developing the application

* **Step2:**

Designing the UI on Pencil tool and then Designing the same GUI on QT Designer and converting the (.ui) files into the python code, while designing the UI some images were also needed which were taken from the internet.

* **Step3:**

Project managing, It means the files that were used to create the project were managed and placed in separate folders to avoid difficulties in finding a specific file.

* **Step4:**

The python files that were being created were then customized and then linked together.

* **Step5:**

One million data was required to run the project so data was scraped from various websites using python code.

* **Step6:**

Data that was scraped was saved into the folder where the python files were being saved as a CSV file.

* **Step7:**

Sorting algorithms were then designed and placed in a separate file after testing them on both the string and integers.

* **Step8:**

Login/Signup coding was integrated in the UI

* **Step10:**

Data from the File of One million data was displayed in the table

* **Step11:**

Integrated the sorting algorithm and displaying the results.

* **Step12:**

Searching methods developed by using python.

* **Step13:**

Integration of searching methods that were being created in the UI.

**Project Description:**

The project works on the data that has been scraped and saved in the CSV file and three major methods are implemented on the data thease methods are:

* Load Data and Display
* Sorting of Data
* Searching from Data

**Data Scraping**

Data scraping is done manually using python scripts and web driver the path of the websites were given manually and then the script was executed it took a lot of time to scrap the data of the website as the data was present on the different pages of the website and the script has to go on the next page scrap data and so on so it took a lot of time.

Python script used:

from typing import Container

import pandas as pd

from bs4 import BeautifulSoup

from selenium import webdriver

import requests

driver = webdriver.Chrome(executable\_path='D:\Projects\Python Project\Manual 2\chromedriver.exe')

products = []

prices = []

locations = []

headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.81 Safari/537.36'}

def data\_getter(page):

    url = f'https://www.pakwheels.com/used-cars/search/-/?page={ page }'

    r = requests.get(url, headers=headers)

    soup = BeautifulSoup(r.text, 'html.parser')

    Price = soup.find\_all('div', {'class': 'price-details generic-dark-grey'})

    name\_data=soup.find\_all('a', {'class': 'car-name ad-detail-path'})

    loc=soup.find\_all('ul', {'class': 'list-unstyled search-vehicle-info fs13'})

    for item in name\_data:

        name=item.text

        name.strip()

        products.append(name)

    for item in Price:

        price=item.text

        price.strip()

        prices.append(price)

    for item in loc:

        location=item.text

        location.strip()

        locations.append(location)

for i in range (2,4554):

    data\_getter(i)

    print(i)

df = pd.DataFrame({'Car Name': products,'Car price': prices,'Car Locations': locations})

df.to\_csv('products.csv', index=False, encoding='utf-8')

print('data scraped successfull')

Data of the cars around the world is scrapped from multiple websites such as

* Pakwheel
* Olx
* Car first
* Etc.

Data that will be scapped is then saved to a csv file then that file is saved in your desktop/Laptop .

**Load Data and Display**

The first option will be Load and Display Data. When you click on that Data will start to load in the Table that will be integrated in the UI.

**Sort Data**

Second option the user is provided is sort data . the data in the csv file will be automatically loaded but the user will have to tell the name of the column he wants to sort and the number of rows he wants to sort from that column.The user is also given a dropdown box in which he will have the option to choose the which of the sorting method he wants to use.

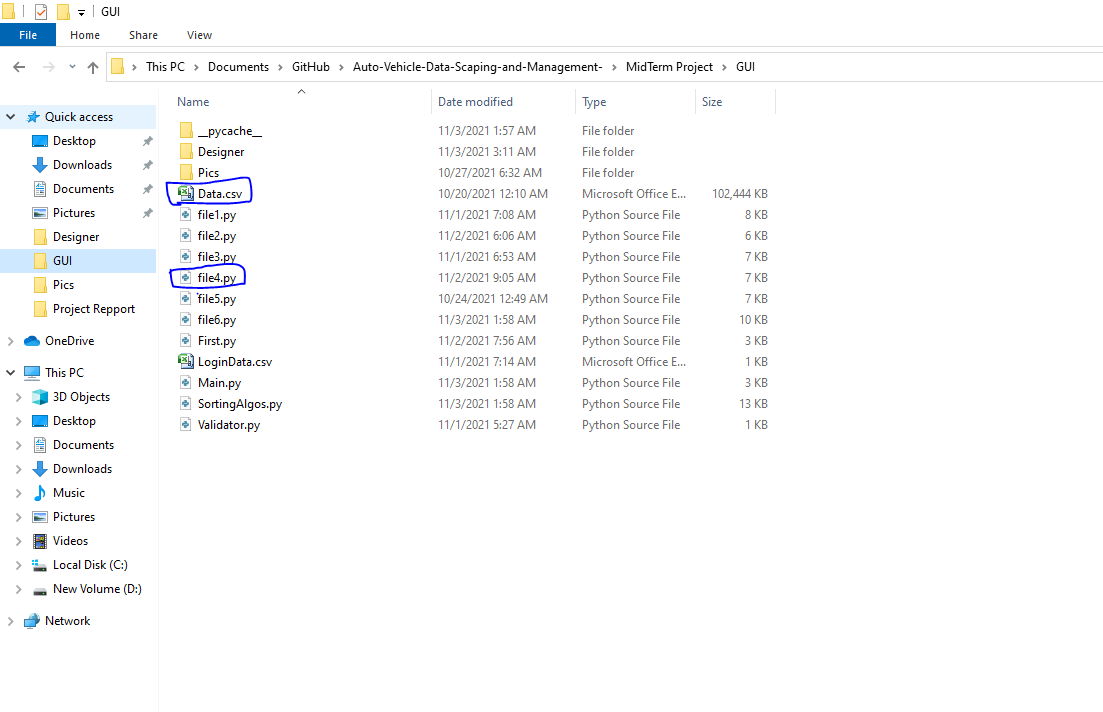
**Search Data**

The final option that is being provided is the search from the data. The screen will open and load the data in the table. A simple and basic search box will be given above the table that will be a linear search linked. There will be advanced filters that the user can also apply by clicking the button Advanced filters.

That was the basic description and an introduction to the working and practicality of the project.

**Displaying data in Table**

The CSV file and the python files both are saved in the same folder and the folder is opened in the visual studio code:



The file was firstly loaded in the code:

def load\_File(self):

        try:

            self.all\_data = pd.read\_csv('Data.csv')

        except:

            print("An Error Occured!")

Then the data is taken row by row from the CSV file and displayed in the Table. The table and the code to display the data will be more explained in the description about GUI.

def datahead(self):

NumRows = 50000

self.tableWidget.setColumnCount(len(self.all\_data.columns))

self.tableWidget.setRowCount(NumRows)

self.tableWidget.setHorizontalHeaderLabels(self.all\_data.columns)

self.progressBar.show()==True

count=1

k=500

for i in range(NumRows):

if i==k:

self.progressBar.setValue(count)

count+=1

k=k+500

for j in range(len(self.all\_data.columns)):

self.tableWidget.setItem(i, j, QTableWidgetItem(str(self.all\_data.iat[i, j])))

self.tableWidget.resizeColumnsToContents()

self.tableWidget.resizeRowsToContents()

if count==100:

self.LoadData.hide()==True

self.progressBar.hide()==True

self.label\_4.show()==True

After the data is successfully loaded the above code will run and it will read the data from the file row by row and display it in the table. This is a time consuming procedure so to save time and to prevent the GUI screen from freezing the displayed data is limited to 50K out of 1M.

**Sorting Data:**

User is being provided with 7 types of different sorting methods that are developed in python and saved in the separate file that is being called as a module to use the methods in it but in the recent project update the file with the sorting algorithms isn’t called as a module rather the list of data that is to be sorted along with name of the method to use for the data sorting is passed to the file itself to a method in that file that will communicate with 2 files (**Sorting\_Algorithms.py**) and (**file6.py**)**.** The list and name of the method will be passed from the **file6.py**

to the **Sorting\_Algorithms.py.** The method in this file will receive the list and the name of the method and then will identify the name of the method then according to the name of the method it will call the method to sort the list. The list after sorting will be then returned from the method and then displayed in the tabl that will be shown in the GUI Part of the report.

**Code**:

Code for communicating with the main code that will sort and show the data in the table :

def Sort\_columns(self):

import pandas as pd

try:

df=pd.read\_csv('Data.csv')

except:

print('Error in file Opening')

col\_name=self.colname1.text()

try:

NumRows=int(self.colname1\_2.text())

except:

print('none')

self.colname1.clear()

count=1

List=[]

headers=['Name','Body Color','Miles Driven','location','Price','Car Body type','Transmission']

if col\_name in headers and col\_name!='':

try:

for items in (df[col\_name]):

if count<=NumRows:

List.append(items)

count+=1

# print (List)

except:

print("Wrong Column name")

else:

print("none")

self.colname1.clear()

self.colname1\_2.clear()

Method=self.method1.currentText()

Sort.SortCol(self,List,Method)

Main code for sorting the table:

def SortCol(self,List,Method):

numcols =1

numrows=len(List)

self.tableWidget.setColumnCount(numcols)

self.tableWidget.setRowCount(numrows)

# if List!="" and Method!="" :

if Method=="Bubble Sort":

bubbleSort(List)

# print(DispList)

for row in range(numrows):

for column in range(numcols):

self.tableWidget.setItem(row, column, QTableWidgetItem((List[row])))

self.tableWidget.resizeColumnsToContents()

self.tableWidget.resizeRowsToContents()

elif Method=="Merge Sort":

DispList=merge\_sort(List)

# print(DispList)

for row in range(numrows):

for column in range(numcols):

self.tableWidget.setItem(row, column, QTableWidgetItem((List[row])))

self.tableWidget.resizeColumnsToContents()

self.tableWidget.resizeRowsToContents()

elif Method=="Heap Sort":

DispList=heapSort(List)

# print(DispList)

for row in range(numrows):

for column in range(numcols):

self.tableWidget.setItem(row, column, QTableWidgetItem((List[row])))

self.tableWidget.resizeColumnsToContents()

self.tableWidget.resizeRowsToContents()

elif Method=="Tim Sort":

DispList=tim\_sort(List)

# print(DispList)

for row in range(numrows):

for column in range(numcols):

self.tableWidget.setItem(row, column, QTableWidgetItem((List[row])))

self.tableWidget.resizeColumnsToContents()

self.tableWidget.resizeRowsToContents()

elif Method=="Quick Sort":

DispList=quick\_sort(List)

# print(DispList)

for row in range(numrows):

for column in range(numcols):

self.tableWidget.setItem(row, column, QTableWidgetItem((List[row])))

self.tableWidget.resizeColumnsToContents()

self.tableWidget.resizeRowsToContents()

elif Method=="Insertion Sort":

DispList=Insertion\_Algo(List)

# print(DispList)

for row in range(numrows):

for column in range(numcols):

self.tableWidget.setItem(row, column, QTableWidgetItem((List[row])))

self.tableWidget.resizeColumnsToContents()

self.tableWidget.resizeRowsToContents()

elif Method=="Slection Sort":

DispList=selectionSort(List)

# print(DispList)

for row in range(numrows):

for column in range(numcols):

self.tableWidget.setItem(row, column, QTableWidgetItem((List[row])))

self.tableWidget.resizeColumnsToContents()

self.tableWidget.resizeRowsToContents()

The code works in such a way that the name of the column and the method of sorting is taken from the GUI and then passed down to the code that will send that name of column and method to the method in the File(**SortingAlgos.py**)the function will then detect the method that is called and then call the method of sorting that are already defined in the same folder and it will apply the sorting on it and display the sorted list in the table.

**Sorting Algorithms:**

The algorithms used to create sorting methods are explained with examples.

Algorithms used are:

* Bubble Sort
* Insertion sort
* Merge Sort
* Quick Sort
* Selection Sort
* Tim Sort(Hybrid)
* Heap Sort

**Bubble Sort:**

This sorting method is really slow as it takes the 2 entries at a time, compares them and swaps their positions according to the value of entries.

For example:

Suppose you have an unsorted array of numbers as below

1,3,4,5,3,2,76,7,4,32,2,1,4

This sorting method will pick the first 2 entries i.e. 1 and 3 and compare them and swap the places of them if required. In this case no need to do that so it will leave it like that and move on.This will help in bubbling the highest entry present in the list at the end and then the process will repeat till all the numbers in the list are sorted.

**Python Code:**

**def bubbleSort(arr):**

**n = len(arr)**

**for i in range(n):**

**for j in range(0, n-i-1):**

**if arr[j] > arr[j+1] :**

**arr[j], arr[j+1] = arr[j+1], arr[j]**

**return arr**

**Insertion Sort:**

It's a sorting method that is similar to how you sort the cards in your hand, you choose a card randomly and then compare it to all others then place it to its right place in the set of cards in your hand.Similarly this algorithm does the same, If you have a set of some numbers then u apply this sorting methods it will move along the numbers and compare it with its preceding key number. If a number that comes in the set that is smaller than its preceder, it will compare that number till it finds a number that is smaller then this number.

For example:

You have a list of some integers as below:

1,5,3,2,7,65,434,54,35,7

Here the sorting method will start from 1, there is no preceder so nothing will happen and the iteration will move to 5, here it will compare 5 with 1, it is clear that 5 is larger then 1 so 5 will remain in its position next up is 3 so it will be compared with its preceder which is 5 after comparing 3 will be less than 5 so the algorithms will shift 3 to the place of 5 and this is how the sorting will be carried out when you will use this sorting method.

**Python Code:**

**def Insertion\_Algo(arr):**

**for i in range(1, len(arr)):**

**key = arr[i]**

**j = i-1**

**while j >= 0 and key < arr[j] :**

**arr[j + 1] = arr[j]**

**j -= 1**

**arr[j + 1] = key**

**return arr**

**Merge Sort:**

The key process of the sorting method in this algorithm is that it distributes the list into smaller parts then sort them and merge them.

For example:

Suppose you have a list of some integers as below:

3,5,3,1,65,3,5,7,3,43,5

The algorithm will break the list into smaller parts as shown:

Distributing :

3,5,3,1,65,3 5,7,3,43,5

Distributing further:

3,5,3 1,65,3 5,7,3 43,5

Sorting:

3,3,5 1,3,65 3,5,7 5,43

Merging:

1,3,3,5,65 3,5,5,7,43

1,3,3,3,5,5,5,43,65 is the sorted array

That's how the algorithm works.

**def merge\_sort(arr):**

**if len(arr)>1:**

**mid\_point=(len(arr))//2**

**L=arr[:mid\_point]**

**R=arr[mid\_point:]**

**merge\_sort(L)**

**merge\_sort(R)**

**i=0**

**j=0**

**k=0**

**while i <len(L)and j<len(R):**

**if L[i]<R[j]:**

**arr[k]=L[i]**

**i+=1**

**else:**

**arr[k]=R[j]**

**j+=1**

**k+=1**

**while i<len(L):**

**arr[k]=L[i]**

**i+=1**

**k+=1**

**while j<len(R):**

**arr[k]=R[j]**

**j+=1**

**k+=1**

**return arr**

**Quick Sort:**

In this sorting method, a pivot is picked and the numbers which are greater than it are placed after the pivot and the numbers which are less then the pivot are placed before the pivot. There are basically 4 methods of picking a pivot:

1. Pick first element as pivot
2. Pick last element as pivot
3. Pick random element as pivot
4. Pick median as pivot

Let x be an element chosen as a pivot, the sorting method will place the x at its right position then the larger elements after the x and smaller element before x.

This sorting method mainly depends on a method partition().The partition method works in the following way, we start from the leftmost element and keep track of the index of smaller or equal elements as i. While traversing, if we find a smaller element, we swap the current element with arr[i]. Otherwise we ignore the current element.

For example:

Let suppose u have a list of integers as follow:

5,8,7,9,12,54,32,18,-1,2,-,1,-6,-2

So we have 13 elements. Let's choose the mid element as pivot which is 32.

5,8,7,9,12,54,**32**,18,-1,42,61,-6,-2

5,8,7,9,12,18,-1,-6,-2 54,42,61

Again choosing Mideans as pivot

5,8,7,9,**12,**18,-1,-6,-2 54,**42**,61

5,8,7,9,-1,-6,-2 18 null 54,61

Repeating

5,8,7,**9**,-1,-6,-2 54,**61**

5,7,8,-1,-6,-2 null 54 null

Repeating

5,7,8,**-1**,-6,-2

5,7,8 -6,-2

Repeating

5,**7**,8 -6,**-2**

5 8 -6 null

**Python Code:**

**def partition(nums, low, high):**

**pivot = nums[(low + high) // 2]**

**i = low - 1**

**j = high + 1**

**while True:**

**i += 1**

**while nums[i] < pivot:**

**i += 1**

**j -= 1**

**while nums[j] > pivot:**

**j -= 1**

**if i >= j:**

**return j**

**nums[i], nums[j] = nums[j], nums[i]**

**def quick\_sort(nums):**

**def \_quick\_sort(items, low, high):**

**if low < high:**

**split\_index = partition(items, low, high)**

**\_quick\_sort(items, low, split\_index)**

**\_quick\_sort(items, split\_index + 1, high)**

**\_quick\_sort(nums, 0, len(nums) - 1)**

**return nums**

**Selection Sort:**

This sorting method finds the minimum value from the list and then places it in the beginning of the list.this algorithm maintains 2 sub arrays, one that is sorted and other that has yet to be sorted.This will be more clear with an example.

For example:

Suppose you have an unsorted array of integers as below:

1,5,3,6,87,9,4,5

The sorting method will find the minimum value from the list, in this case it’s 1

Sorted array:

1

Unsorted array:

5,3,6,87,9,4,5

Sorted array:

1,3

Unsorted array:

5,6,87,9,4,5

Sorted array:

1,3,4

Unsorted array:

5,3,6,87,9,5

And so on until the array is sorted

**def selectionSort(arr):**

**for i in range(len(arr)):**

**min\_idx = i**

**for j in range(i+1, len(arr)):**

**if arr[min\_idx] > arr[j]:**

**min\_idx = j**

**arr[i], arr[min\_idx] = arr[min\_idx], arr[i]**

**return arr**

**Tim Sort:**

It is the hybrid algorithm of insertion sort and merge sort. This algorithm uses 2 algorithms together in a hybrid from reading a hybrid sorting method.

Examples of insertion sort and merge sort are discussed above so no need to revise them just need to know that this algorithm uses both of them together to sort the array.

**MINIMUM=32**

**def find\_minrun(n):**

**r = 0**

**while n >= MINIMUM:**

**r |= n & 1**

**n >>= 1**

**return n + r**

**def insertion\_sort(array, left, right):**

**for i in range(left+1,right+1):**

**element = array[i]**

**j = i-1**

**while element<array[j] and j>=left :**

**array[j+1] = array[j]**

**j -= 1**

**array[j+1] = element**

**return array**

**def merge(array, l, m, r):**

**array\_length1= m - l + 1**

**array\_length2 = r - m**

**left = []**

**right = []**

**for i in range(0, array\_length1):**

**left.append(array[l + i])**

**for i in range(0, array\_length2):**

**right.append(array[m + 1 + i])**

**i=0**

**j=0**

**k=l**

**while j < array\_length2 and i < array\_length1:**

**if left[i] <= right[j]:**

**array[k] = left[i]**

**i += 1**

**else:**

**array[k] = right[j]**

**j += 1**

**k += 1**

**while i < array\_length1:**

**array[k] = left[i]**

**k += 1**

**i += 1**

**while j < array\_length2:**

**array[k] = right[j]**

**k += 1**

**j += 1**

**def tim\_sort(array):**

**n = len(array)**

**minrun = find\_minrun(n)**

**for start in range(0, n, minrun):**

**end = min(start + minrun - 1, n - 1)**

**insertion\_sort(array, start, end)**

**size = minrun**

**while size < n:**

**for left in range(0, n, 2 \* size):**

**mid = min(n - 1, left + size - 1)**

**right = min((left + 2 \* size - 1), (n - 1))**

**merge(array, left, mid, right)**

**size = 2 \* size**

**return array**

**Heap sort:**

This sorting method 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 in the list.

**def heapify(arr, n, i):**

**largest = i**

**l = 2 \* i + 1**

**r = 2 \* i + 2**

**if l < n and arr[i] < arr[l]:**

**largest = l**

**if r < n and arr[largest] < arr[r]:**

**largest = r**

**if largest != i:**

**arr[i],arr[largest] = arr[largest],arr[i]**

**heapify(arr, n, largest)**

**def heapSort(arr):**

**n = len(arr)**

**for i in range(n // 2 - 1, -1, -1):**

**heapify(arr, n, i)**

**for i in range(n-1, 0, -1):**

**arr[i], arr[0] = arr[0], arr[i]**

**heapify(arr, i, 0)**

You can find the python codes of these sorting algorithms in file (**Sort\_Algorithms.py**).

**Searching From Data**

The user is given a simple and basic searching tool in the window of search data but there is also an advanced filter button provided there so that the user can also apply the advanced filters on the search.

The advanced filter option have the various options such as:

* Starts with
* Ends with
* Contains

With the help of the internet the codes of these searching methods are being developed and saved in a separate file named as (**Advanced\_Filters.py**). This file will also have a function like the (**Sort\_Algorithm.py**) that will communicate with the file that will bring the data from GUI. The data will contain the name of the method only. The CSV file will be loaded and converted to the list in (**Advanced\_Filters.py**), then the method will be identified and the method that is identified will be called to search the data from that CSV file data list. After the data is searched, that data will be loaded in the table given in the GUI.

The code that will take the data from the GUI and send it to the other file for further working is:

def Search(self):

self.tableWidget.clear

Method=self.comboBox.currentText()

text1=self.plainTextEdit.toPlainText()

SM.SearchData(self,text1,Method)

It will firstly clear the table if any data is present there then it will take the data from the GUI and pass it to the method in another file that is mainly responsible for data searching and displaying.

def SearchData(self,text1,method):

import pandas as pd

from PyQt5.QtWidgets import QApplication, QWidget, QFileDialog, QTableWidget, QTableWidgetItem

try:

all\_data = pd.read\_csv('Data.csv')

except:

print("An Error Occured!")

list=[]

for items in all\_data:

list.append(items)

index\_list=[]

if method=='Starts with"':

returnedlist=startswith(list, text1)

for i in returnedlist:

index\_list.append(i)

elif method=='Linear Search':

returnedlist=Linear\_search(list, text1)

for i in Linear\_search(list, text1):

index\_list.append(i)

elif method=='Ends With':

returnedlist=endswith(list, text1)

for i in endswith(list, text1):

index\_list.append(i)

numrows=len(index\_list)

numcols=1

for row in range(numrows):

for column in range(numcols):

for i in index\_list:

self.tableWidget.setItem(row, column, QTableWidgetItem((list[i])))

self.tableWidget.resizeColumnsToContents()

self.tableWidget.resizeRowsToContents()

This is the main code that will get the data from the Method(Search in GUI code file)and then it will further apply operation on it.

**GUI Designing and Description**

| Screen Design | 4.PNG | | |
| --- | --- | --- | --- |
|  | Component | Name | Object name |
| pushbutton | Launch | Launch |
|  |  |  |
|  |  |  |
| How To Access the use case | Run the Application | | |
| Description | First page user will see  Clicking on the launch button makes the button disappear and a loading bar will show up.  After the loading bar reached 100% the whole window will be closed and the application will switch windows | | |

| Screen Design | 5.PNG | | |
| --- | --- | --- | --- |
|  | Component | Name | Object name |
| Pushbutton1 | Sign In | sigin |
| Pushbutton2 | Sign Up | signup |
| Pushbutton3 | Exit | mainexit |
|  | PlainTextEdit1 | Email | Email |
|  | PlainTextEdit2 | Password | Password |
| How To Access the use case | Run the Application->Click launch and wait | | |
| Description | Sign In will take to Login page  Sign Up will save the data in the textedits  Exit will terminate the application | | |

| Screen Design | 6.PNG | | |
| --- | --- | --- | --- |
|  | Component | Name | Object name |
| Pushbutton1 | Sign In | sigin |
|  | PlainTextEdit1 | Email | Email |
|  | PlainTextEdit2 | Password | Password |
| How To Access the use case | Run the Application->Click launch and wait->Click Sign In | | |
| Description | If Email, password are correct screens will switch to main menu | | |

| Screen Design | 7.PNG | | |
| --- | --- | --- | --- |
|  | Component | Name | Object name |
| Pushbutton1 | Load and Display data | LoadData |
|  | Pushbutton2 | Data Sorting | Sort |
|  | Pushbutton3 | Data Searching | search |
| How To Access the use case | Run the Application->Click launch and wait->Click Sign In->Type email and password you signed up with and Click Sign In | | |
| Description | Load and display data will switch the screen to display data screen  Data sorting button will switch the screen to sorting data screen  Data searching button will take you to data sorting screen | | |

| Screen Design | 8.PNG  9.PNG10.PNG | | |
| --- | --- | --- | --- |
|  | Component | Name | Object name |
| Pushbutton1 | Load and Display | LoadData |
|  | Pushbutton2 | Go Back | Sort |
| How To Access the use case | Run the Application->Click launch and wait->Click Sign In->Type email and password you signed up with and Click Sign In->Click Load and display Data | | |
| Description | First a plain screen will appear  Clicking load and display will activate the code and a progress bar  After fully loading data progress bar will disappear and data will be displayed  Displayed data limited to 50K only as to save time and preventing GUI freezing | | |

| Screen Design | 11.PNG  12.PNG | | |
| --- | --- | --- | --- |
|  | Component | Name | Object name |
| Pushbutton1 | apply | Apply1 |
|  | Pushbutton2 | Clear table | Push2 |
|  | PlainTextEdit1 | Column name | colname |
|  | PlainTextEdit2 | Rows | row |
|  | Combobox | methods | combobox |
| How To Access the use case | Run the Application->Click launch and wait->Click Sign In->Type email and password you signed up with and Click Sign In->Click Data sorting | | |
| Description | Write the name of the column u want to sort data from  Write the numbers of rows you want to sort data from  Chose the sorting method from combo box  Click apply and wait  Sorted data will be displayed in the table | | |

| Screen Design | 13.PNG | | |
| --- | --- | --- | --- |
|  | Component | Name | Object name |
| Pushbutton1 | search | Search1 |
|  | Pushbutton2 | Apply Advanced filters | filters |
|  | Pushbutton2 | Go back | Gobacksearching |
| How To Access the use case | Run the Application->Click launch and wait->Click Sign In->Type email and password you signed up with and Click Sign In->Click Data searching | | |
| Description | 10k Data will be preloaded in the table.  After u click search the thing u want to search will be searched from the whole data,Table will be cleared and the searched data will be displayed  Clicking on advanced filters will open a new window aplly filters on the search | | |

**Ending:**

The GUI and all the files are interlinked, all the sorting methods are integrated and the data is also being displayed .