**Scrapit**

CS162- Final Project Requirement Document



**Project Supervisor**

Mr. Samyan Qayyum Wahla

**Project Members (G25)**

|  |  |
| --- | --- |
| Syed Azeem Ali Hashmi | 2020-CS-156 |

Department of Computer Science

University of Engineering and Technology, Lahore

Pakistan

**Table of Contents**

[1. Project Overview: 3](#_Toc87048402)

[1.1. Project Detail: 3](#_Toc87048403)

[2. Business Case: 3](#_Toc87048404)

[2.1. Business Need: 3](#_Toc87048405)

[2.2. End User: 3](#_Toc87048406)

[2.2.1. Tourists: 4](#_Toc87048407)

[2.2.2. Tourism Service Providers: 4](#_Toc87048408)

[2.2.3. Hotel’s Owner: 4](#_Toc87048409)

[3. Motivation: 4](#_Toc87048410)

[4. Impacts and Implications: 4](#_Toc87048411)

[4.1. Impacts if project proceed: 5](#_Toc87048412)

[4.2. Impacts if project not proceed: 5](#_Toc87048413)

[5. Technical Detail: 5](#_Toc87048414)

[5.1. Entity Names: 5](#_Toc87048415)

[5.2. Attributes of Entity: 5](#_Toc87048416)

[5.3. Sample of Scrapping Source: 6](#_Toc87048417)

[5.4. Repository Link: 6](#_Toc87048418)

[5.5. Sorting Algorithms: 6](#_Toc87048419)

[5.5.1. Quick Sort: 7](#_Toc87048420)

[5.5.2. Bubble Sort: 7](#_Toc87048421)

[5.5.3. Merge Sort: 7](#_Toc87048422)

[5.5.4. Insertion Sort: 7](#_Toc87048423)

[5.5.5. Selection Sort: 7](#_Toc87048424)

[5.5.6. Radix Sort: 7](#_Toc87048425)

[5.5.7. Heap Sort: 7](#_Toc87048426)

[5.5.8. Counting Sort: 8](#_Toc87048427)

[5.5.9. Bucket Sort: 8](#_Toc87048428)

[5.5.10. Hybrid Sort: 8](#_Toc87048429)

[5.5.11. Tree Sort: 8](#_Toc87048430)

[6. Searching Algorithms: 8](#_Toc87048431)

[6.1. Linear Search: 8](#_Toc87048432)

[6.2. Binary Search: 8](#_Toc87048433)

[7. Searching Filters: 8](#_Toc87048434)

[8. Multi-Level Sorting: 9](#_Toc87048435)

[9. Other Feature: 9](#_Toc87048436)

[10. Interface: 9](#_Toc87048437)

[11. Sorting Algorithms: 10](#_Toc87048438)

# Project Overview:

## Project Detail:

World wide web is collection of millions of webpages consist on uncountable data and information. This data and information are useful in many ways depend on how we collect it or how we utilize it. But the main concern is we need information about what. So here I decided to get the data about hotels which is helpful in tourism. The travel and tourism industry has always been an important part of the service sector in many countries. Tourists could easily interact with the service provider, but that meant putting in extra effort into engaging with each service provider to try and find a good plan that covers all criteria. The rise of the internet made things easy to happen. In tourism we mainly need a data to choose which place will be better for us. One of the most popular ways to gather the data is through web scraping. Here I am making a bot called web scraper which automates the process and collect the information on its own. Hotel’s data is very useful for the hotels booking sites or for the tourists or for the tour managers.

Now the main concern is which kind of data we need of hotels. So mainly we need hotels name, price, location, contact number, ratings, reviews, amenities and room type.

All of the procedure will carry on step by step. First it will take URL and will store this information, after scraping, in lists then we show it on csv files. Moreover, I will make a user-friendly interface of desktop application using PyQt5 where all data I have scrapped will be shown in table. This interface includes other things too like one can sort all the data using different sorting algorithms and calculate its time taken too. Filters and multiple level sorting will also be implemented. Status bar will show how much data is scrapped. This interface will include start stop and pause buttons too for scrapping.

# Business Case:

## Business Need:

This project will benefit general tourists as well as tour services providers. For general tourists it will help them to watch everything they need to know about hotels and for service providers it will help them to compare different hotels and finding the best one according to their price. Other than this hotel owners can also use this information to make their hotels range respectively for the better competition.

## End User:

Web scrapping especially hotel scrapping will benefit multiple groups of people and industries described below:

### Tourists:

Mainly tourists want to know about the residences where they are traveling so, they can have rough idea about their expenditures so by using the data scrapped by Scrapit firstly they don’t need any kind of travel agency which can provide them a list of hotels or a better option. Moreover, they can contact and book hotels on their own which can save their time and money.

### Tourism Service Providers:

Tour agencies are always in dire need of data about locations and hotels so they can give variety of hotels to their clients which impacts their client strongly as well as variety of hotels guarantees them that client will surely select one of it. So by using the scrapped data they will have bundle of hotels with their needed information.

### Hotel’s Owner:

Hotel’s owners have tough competition when there are a lot of hotels in nearby locality so by using scrapped data, they will know which services are being provided by their rivals so they can improve themselves which will help them get popular soon.

# Motivation:

Mainly I got this idea from my cousin’s business. In pandemic, lots of business got effected and after lockdowns people are facing financial problems so they are raising their charges for the services they are providing. In this atmosphere I decided to work on this project so I can help those who really need it and can save their money on the things which are already free or are of low price. On the other hand time has evolved itself. Now in digital world we need to automate everything so it will beneficial in this way too.

# Impacts and Implications:

Here we will discuss the impacts or effects of the project if it is completed or reprimanded for any reason, how it would affect the business, how it would affect the needs of the business and many of these factors.

## Impacts if project proceed:

This data will be very useful for multiple groups of people and hotel industries as data provided by scrapper is very efficient.

## Impacts if project not proceed:

In case this project not get completed many tourists has to follow that slow and sturdy procedure which is very expensive and time taking.

# Technical Detail:

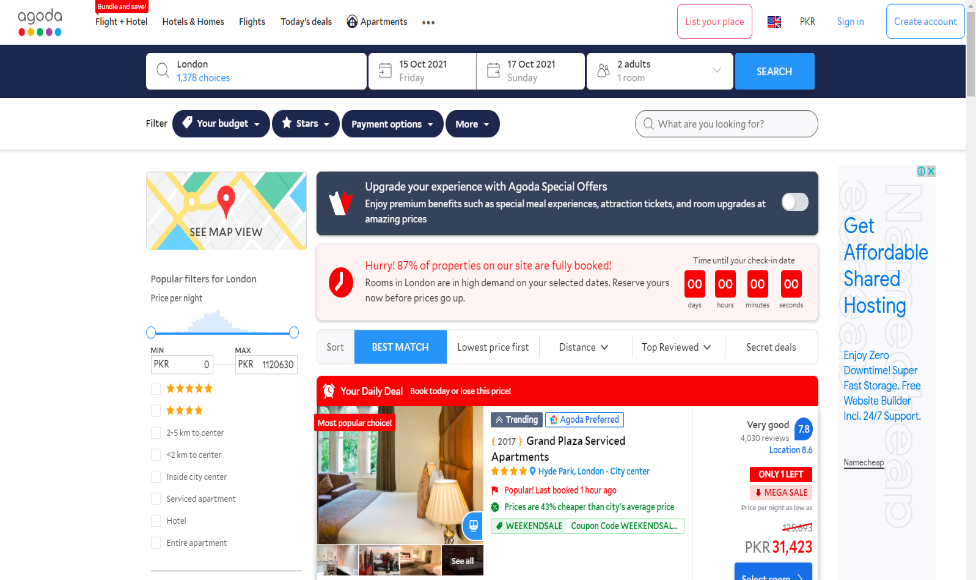
## Entity Names:

1. Name
2. Price
3. Location
4. Reviews
5. Ratings
6. Amenities
7. Contact
8. Room Type

## Attributes of Entity:

|  |  |  |
| --- | --- | --- |
| **Name** | **Data Type** | **Description** |
| Name | String | It tells the name of hotel |
| Price | Int | It will tell the price of room |
| Location | String | It will talk about the location where hotel is situated |
| Reviews | String | It will talk about the reviews given by the visitors |
| Ratings | String | It will show the ratings given by visitors |
| Amenities | String | It will tell extra facilities given by the hotel. |
| Contact | String | It will give us contact number |
| Room Type | String | It tells how many people accommodate the room |

## Sample of Scrapping Source:



## Repository Link:

Repository link is given below:

[*https://github.com/AzeemHashmi/CS261F21PID66*](https://github.com/AzeemHashmi/CS261F21PID66)

## Sorting Algorithms:

Sorting Algorithms overview is given below whereas description of sorting algorithms will be described in next main heading.

### Quick Sort:

Quick Sort works in a way that it chooses a pivot as a reference element and divide the list into two halves placing smallest elements on left and largest elements on right. It continuously follows the same procedure until list become sorted.

### Bubble Sort:

Bubble Sort works in sequential manner means it compares adjacent elements and swap them if they are in wrong order.

### Merge Sort:

Merge Sort works on the principle of Divide and Conquer rule. It divides the list until it reaches the element whose length is 1 than place it in ordered way and merge it together.

### Insertion Sort:

Insertion sort works in a sequence. It compares one item at a time with all other elements and then place it according to required order.

### Selection Sort:

Selection Sort is also called comparison sort. In this sorting algorithm we logically divide the list into two halves i.e., sorted and unsorted part. We select the smallest element in list and place it on first order and next after next.

### Radix Sort:

Radix sorting is an integer sorting algorithm that sorts data with integer keys by grouping keys of same position and significant value (in math unit).

### Heap Sort:

This is an advanced version of the binary search tree BST. It makes the heap by adjusting the position of the elements in the array itself.

### Counting Sort:

Counting sort counts the number of quantities of object and store it in another array and then do some calculations on it. In its result list become sorted.

### Bucket Sort:

Bucket sort works on the principle that it divides each element in number of bucket then each bucket sorts itself and then whole buckets arrange themselves in sequence.

### Hybrid Sort:

Hybrid Sort merges two sorting algorithms i.e., insertion sort and merge sort into one. It first implements divide rule and then conquer it with insertion algorithm.

### Tree Sort:

A tree sort is a sorting algorithm that builds a binary search tree from the items to be sorted, then iterates the tree so that the items come out in order.

# Searching Algorithms:

Searching algorithms means the method to find or search something in list. There are different ways to search elements. Some searching techniques are discussed below:

## Linear Search:

This searching technique will work sequentially. It means it will check each element of list until the required element found.

## Binary Search:

This search technique follows divide and conquer rule. It divides the sorted array unless each division will become of length 1 and then found the required element.

# Searching Filters:

1. Contains
2. Starts with
3. Ends with

# Multi-Level Sorting:

I will perform multi- level sorting by checking elements character by character. If name Ali and Azeem has to compare than first it will list Ali than Azeem.

Moreover, it will arrange parallel columns according to the column we are arranging.

# Other Feature:

This project will provide you a URL bar so you can scrap everything you want.

# Interface:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | UI Component Name | Type of UI component | Purpose of UI Component/Other details | |  | | | | Progress Bar | Progress Bar | To represent how much data has been scrapped | | Play | Button | It will use to continue the scrapping | | Pause | Button | It will use to pause the scrapping | | End | Button | It will terminate the scrapping | | Sorting Algorithms | Radio Button | It will perform scrapping using that specific algorithm whose radio button is pressed. | | Arranging | Radio Button | It will arrange data in ascending and descending way depending on radio button pressed | | URL | Text Box | It will take URL as input | | Search | Button | It will search the URL entered | | Data Table | Table | It will show the data scrapped | | Time Elapsed | Text Box | It will display the time lapse taken by scrapping algorithm | | Search Box | Text Box | It will display data on table according to the keywords provided by user | |

# Sorting Algorithms:

Sorting algorithms are used to sort data sequentially. There are different algorithms to sort data. Some are described below:

|  |  |
| --- | --- |
| ***Insertion Sort*** | |
| Description | In insertion sort, we consider the array is divided into two parts i.e.., sorted and unsorted part. Usually, we consider first element sorted in this sorting algorithm so we set the marker on second element and compare it with the elements placed on left and according to the requirement we insert it in its correct position in left and set the marker to the next right position till the end of array. In this way complete array get sorted.  Visual representation of insertion sort is: |
| Pseudo Code | To sort array:   * Loop will iterate from 1 index to last index * Consider loop variable as key and compare it with rest elements * If key contain large value, then rest swap them * Do same procedure on all elements |
| Python Code |  |
| Strength | * Good for small elements data * It is stable sorting algorithm |
| Weakness | * Worst for large number of elements of data |

|  |  |
| --- | --- |
| ***Merge Sort*** | |
| Description | Merge sort follows divide and conquer rule. It comprises on two functions i.e.., MergeSort and Merge. MergeSort function calls itself recursively and divides the array into two halves till each half becomes the size of 1. Then by using Merge function we compare each element of two halves and sorts it in temporary array. This procedure continues until all elements become sorted. |
| Pseudo Code | * Divide the array into two halves * Apply the mergesort function in first half * Apply the mergesort function in second half * Repeat the same procedure * Call merge function |
| Python Code |  |
| Strength | * Good for large number of inputs * Time taken is consistent |
| Weakness | * Bad for small number of inputs * Memory usage is bad with respect to performance |

|  |  |
| --- | --- |
| ***Bubble Sort*** | |
| Description | Bubble sort is also called sinking sort. Its sorting technique is so simple. It works in way that it compares two adjacent elements of a list and swap them if they are in wrong order. It continues this process until the complete list become sorted.  According to real world usage its performance is not as efficient. |
| Pseudo Code | * Compare two adjacent elements from start of array * Swap them if they are in wrong order * Step forward * Repeat the procedure until list become sorted |
| Python Code |  |
| Strength | * It detects quickly that list is sorted or not * In this case it reduces its time complexity |
| Weakness | * According to real world usage its performance is not as efficient. * It takes too much time |

|  |  |
| --- | --- |
| ***Quick Sort*** | |
| Description | Quick Sort follows divide and conquer rule. It works in a way that it chooses a pivot as a reference element and divide the list into two halves placing smallest elements on left and largest elements on right. Then it calls itself recursively on left and right halves and continues it until the array become sorted.  Choosing a pivot can be of four types   * First element * Last element * Median Element * Randomly Selected Element |
| Pseudo Code | * Choose the pivot * Place small element to the left of pivot and greater elements to the right * Apply quick sort on left and right parts |
| Python Code |  |
| Strength | * It does not need additional storage for sorting * It is faster |
| Weakness | * It is not stable * Speed depends on the size of input list |

|  |  |
| --- | --- |
| ***Selection Sort*** | |
| Description | Selection sort is simple technique of sorting. In this sorting algorithm we consider the list divided into two parts i.e.., sorted and unsorted.  Select the element at start index of list and compare it with the rest. Where you find the smallest number in list swap the first element with that number and increase the loop variable so this time it selects the second number and compare it with the elements on it right.  Repeat the procedure until the list become sorted. |
| Pseudo Code | * Select the first element * Swap it with smallest element in list on right side * Step forward and select the element next to first one and so on * Compare them with right side elements until list become sorted |
| Python Code |  |
| Strength | * It makes minimum number of swaps * Its auxiliary memory is limited * It is simple |
| Weakness | * Time complexity is worst for large input of data |

|  |  |
| --- | --- |
| ***Heap Sort*** | |
| Description | In heap sort we first make a heap of given elements. Heap is a tree like structure. After making a heap we swap parents node with child nodes to make it confirm that parent node holds the largest value. After that we swap the root node with last node of heap and then delete the last node. In this way array become sorted |
| Pseudo Code | * Build heap * Create max heap of given data * Swap root with last node * Delete last node * Repeat procedure until list become sorted |
| Python Code |  |
| Strength | * It is faster * It doesn’t waste time on linear time scan |
| Weakness | * It is not stable * It is slower just for real numbers |

|  |  |
| --- | --- |
| ***Tree Sort*** | |
| Description | Tree sort basically works on two concepts. BST and in order traversal. In this sorting technique we firstly make a binary search tree from given input data by considering the rule that left child has small value than parent and right child has greater value than parent. After that we move in BST by following in order traversal which means elements on left will be written firstly. This is how list become sorted. |
| Pseudo Code | * Make Binary search tree of given input * Do in-order traversal |
| Python Code |  |
| Strength | * It has good time efficiency * It is self-balancing |
| Weakness | * It needs separate memory for tree storage |