Software Requirements Specification

For

# Sorting Visualizer

9th NOVEMBER, 2022

Prepared by

|  |  |  |
| --- | --- | --- |
| **Specialization** | **SAP ID** | **Name** |
| B. TECH CSE AIML(NH) | 500082404 | Aditi Prasad |
| B. TECH CSE AIML(NH) | 500083748 | Somyansh Avasthi |
| B. TECH CSE AIML(NH) | 500085073 | Ananya |
| B. TECH CSE AIML(NH) | 500082822 | Agnibha Dutta Ray |

**Table of Contents**

|  |  |
| --- | --- |
| **Topic** | |
| Table of Content | |
| Revision History | |
| 1 | Introduction |
|  | 1.1 Purpose of the Project |
|  | 1.2 Target Beneficiary |
|  | 1.3 Project Scope |
|  | 1.4 References |
| 2 | Project Description |
|  | 2.1 Reference Algorithm |
|  | 2.2 Data/ Data structure |
|  | 2.3 SWOT Analysis |
|  | 2.4 Project Features |
|  | 2.5 Design and Implementation Constraints |
|  | 2.6 Design diagrams |
|  | 2.7 Assumption and Dependencies |
| 3 | System Requirements |
|  | 3.1 User Interface |
|  | 3.2 Software Interface |
| 4 | Non-functional Requirements |
|  | 4.1 Performance requirements |

**INTRODUCTION**

**1.1. Purpose of the Project**

To find out the most optimized approach to sort the data based upon the time and space complexity and then visualizing the result on the screen.

**1.2.** **Target Beneficiary**

Talking about the applications of sorting, it is utilized to solve a variety of complex problems. According to estimates, it has been observed that over 25% of all computing time is spent on sorting, with some organizations spending more than 50% of their computing time sorting lists. On a very basic level we use sorting for searching particular elements in a dataset. Secondly, sorting can also be used for matching entries in lists. Due to these applications, our project can be used in the banking sector, teaching sector, the finance sector, e-commerce sector, etc.

**1.3.** **Project Scope**

Our project will be able to sort the data based upon the filters the user has entered. It will be able to visualize the data according to the user’s needs. We can also use it in the gaming scenarios like we create abacus, sudoku and all.

**1.4.** **References**

 [1] Sorting Visualiser  dev community

[2] SORTING ALGORITHM VISUALIZER

[3] https://digitalcommons.ric.edu/cgi/viewcontent.cgi?article=1129&context=honors\_projects

[4]https://www.theseus.fi/bitstream/handle/10024/507342/Bikram\_Karki.pdf;jsessionid=898A8374114942395546323B3B139DB9?sequence=2

[5] http://faculty.tamuc.edu/dcreider/csci520/Note520/Note%206.htm

[6]https://www.interviewkickstart.com/learn/time-complexities-of-all-sorting-algorithms#:~:text=a%20simplified%20form.-,Space%20Complexity%3A,complexity%20is%20O(1).

[7] https://panthema.net/2013/sound-of-sorting/

[8] https://dl.acm.org/doi/10.1145/169059.169078

**PROJECT DESCRIPTION**

**2.1 Reference algorithm and Data Structures**

In this project we take the array and we sort in different types of sorting algorithms like insertion sort , selection sort , bubble sort , merge sort and quick sort  then we visuallising this by making bars and see which sorting algorithm is best suitable for smaller and larger array.By this we can’t dabbled among the sorting algorithms and we completely see the complete understanding of sortings algorithms.We also see the comparisons among the sortings  and their time and space complexity by the bars ,our output is coming in the form of bars.

Points to ellaborate for  all sortings :

A. Insertion sort -In this type of sorting , the array is searched sequentially and unsorted items are moved and inserted into the sorted sub-list (in the same array). This algorithm is not suitable for large data sets as its average and worst case complexity are of Ο(n2), where **n** is the number of items.

B.Selection sort- In this type of sorting , the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

C.Bubble sort- In this sorting , comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of Ο(n2) where **n** is the number of items.

D.Merge sort- In this type of sorting ,  first  it divides the array into equal halves and then combines them in a sorted manner. The time complexity of this sorting is o (n logn ). It is applicable for short array.

E.Quick sort- In this sorting ,it is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays. A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value.

Quicksort partitions an array and then calls itself recursively twice to sort the two resulting subarrays. This algorithm is quite efficient for large-sized data sets as its average and worst-case complexity are O(n2), respectively.

**2.2.** **SWOT Analysis**

**Strengths:**

It can cater the needs of the organisation on sorting the data with least time and space complexity based on the size of the data or array.

**Weakness:**

The user has to specify the parameters based on which the data needs to be sorted.

**Opportunity:**

We can use various artificial intelligence algorithms to enhance the project and make it decide on its own on which dataset which sorting algorithm can be performed to provide most optimized results.

**Threat:**

Dependency on the user as choosing the wrong parameters might result in accomplishing less optimal results.

**2.3.** **Project Features**

The output will first show a non sorted array in the form of a bar graph , then it will visualise the process in which the array is being sorted by swapping the bar graph position according to the sorting algorithm.

**2.4 Design and Implementation Constraints-**

Technology used: Vs code, or any other supporting python versions .

Language requirement:  Python

Programming standards:

- name conventions for variables and functions

- indentation

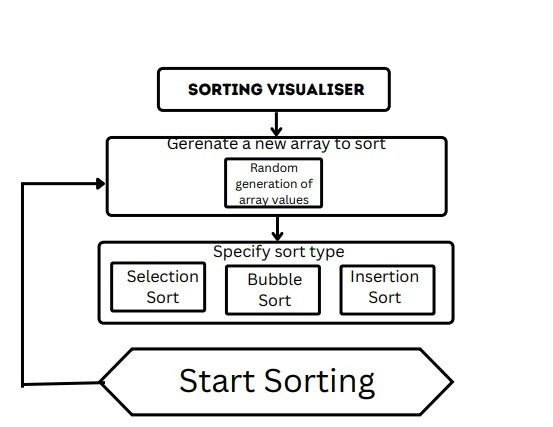
- simple coding methodology

- sortings of array

- Creating and generate window

-visualising

**2.5 Design diagram:**

**\**

**2.6 Assumptions and Dependencies:**

- Minimal hardware specifications PC/computer device good internet connection , best platform for coding like vs code, python 3.8 or 3.9, spyder and making the best visualiser for all sortings.

Dependencies: - Internet connectivity .

**SYSTEM REQUIREMENTS**

**3.1 User Interface –**

-It is a basic graph layout which shows the dataset as a bar graph.

-There are buttons which ask from the users which type of sorting should be done.

-The graph which shows the dataset as bar will move according to the requested sorting

-After completing a sort , the program will halt.

-It won’t accept any other sort algorithm from the user as the dataset is already sorted.

-After pressing the shuffle button, it will shuffle the dataset and start accepting the sorting procedure again.

**3.2 Software Interface –**

-We have used python for this project.

-We have used some libraries like tkinter for basic GUI functions and random to generate datasets for our project.

-We are using 3.9 Python version.

**NON-FUNCTIONAL REQUIREMENTS**

**4.1 Performance requirements** –

-The code should show the dataset smoothly to compare between all the sorting algorithms.

- The code should not be complicated.

- The Graph should show how the datasets are sorted accurately.

- The proposed system should work with requirement being python 3.9.