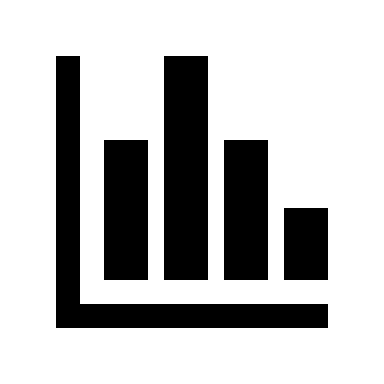
Department of Computer Science

M.Sc. Computer Science

Project report on

Sorting Visualizer

**

By: Vaidehi Admulwar (15617)

Khushal Dalavi (15620)

Dheeraj Bhamare (15635)

A

PROJECT REPORT ON

**SORTING VISUALIZER**

**GUIDED BY**

Prof. Madhuri Khadtare

**SUBMITTED BY**

Vaidehi Admulwar

Khushal Dalvi

Dheeraj Bhamare

IN PARTIAL FULFILMENT OF

M.Sc. (COM SCI) PART II

**UNIVERSITY OF PUNE**

(2022-2023)

MAHARASHTRA EDUCATION SOCIETY’S

**ABASAHAEB GARWARE COLLEGE**

PUNE 411004

**ACKNOWLEDGEMENT**

On this great occasion of accomplishment of our project on” Sorting visualizer”,

We would like to sincerely express our gratitude Prof. Madhuri khadtare ma’am, who has been supported through the completion of this project.

We would also be thankful to our Head of Department

Prof. Chitra Nagarkar ma’am for providing all the require facilities in completion of this project.

Finally, as one of the team members, I would like to appreciate all my group members for their support and coordinate; I hope we will achieve more in our future endeavors.

**INDEX**

1. Abstract

2. Introduction

3. Features

4. Requirement Analysis

5. Requirements

6. UML diagrams

6.1 Class diagram

6.2 Use case diagram

6.3 Sequence diagram

6.4 Activity diagram

7. User guide

8. Unit Testing

9. Conclusion n Future Work

10. Bibliography

1. **Abstract**

This project outlines a study that tested the benefits of animated sorting algorithms for teaching. To visualize four sorting algorithms, a web-based animation application was constructed. A visualization of data is implemented as a bar graph, after which a data sorting and algorithm may be applied. The resulting animation is then performed either automatically or by the user, who then sets their own pace. This is research on the computer science curriculum's approach to learning algorithms. The experiment featured a presentation and a survey, both of which asked students questions which may illustrate improvements in algorithm comprehension. These findings and reactions are catalogued in this document and compared to earlier investigations**.**

1. **INTRODUCTION**

The sorting visualizer project is to create a web application as a visualization tool. A single-page web application built using modern JavaScript technology that will visualize the flow and logic of various sorting algorithms. The UI will contain options to select one of the sorting algorithms which were implemented and several items or elements in the data array, control buttons and navigate to previous or next steps along with an option for sorting speed and color mode. The data array of the selected size will be filled in with randomly generated unique values. The data set is represented as a vertical bar with the height of their respective values. After the sorting is started, the stepwise arrangement of data in ascending order based on their value/height will be visualized in the UI. In this application students or users are enter its own array and illustrate the any sorting algorithm and visualize.

1. **Features**

1) Colored representation of step being executed.

1.1) blue: default

1.2) yellow: Being compared

1.3) Red: Identified as in incorrect position and to be moved

1.4) Green: In correct position

2) 3 Controls for visualizations

2.1) Speed of visualization (5 speed levels)

2.2) Data size ()

2.3) Generation of new data (Randomly generate new data).

3) Time and Space complexity of algorithm being visualized.

1. **Feasibility Study**

Economic Feasibility-

We are developing a project for practice purpose that will give us a lot of confidence to build a bigger project in future. Hence the economic feasibility is very good. Languages used for building this system are free to use.

Operational Feasibility-

User can set any random array also set time speed to visualize and also Set its own array, and choose any sorting algorithm which it apply to array and press on start button to start sorting. This system will have a simple and easy, user- friendly interface.

Technical Feasibility-

In this aspect of feasibility, we consider the technical equipment’s require for system development, for this we require various hardware equipment’s such as computer, HTML and JavaScript language developers which are easily available.

Time Feasibility-

Time feasibility is a measure of how reasonable the project timetable. It is necessary to determine whether the deadlines are mandatory or desirable. The project was completed on time with all the module deadlines met before time.

1. **Requirements**

**Software**

Languages: JavaScript, HTML, CSS.

Debugger Tool: Visual Studio 2021 Community version.

Web Server: Google Chrome

**Hardware**

We strongly recommend a computer fewer than 5 years old.

Processor: Minimum 1 GHz; Recommended 2GHz or more

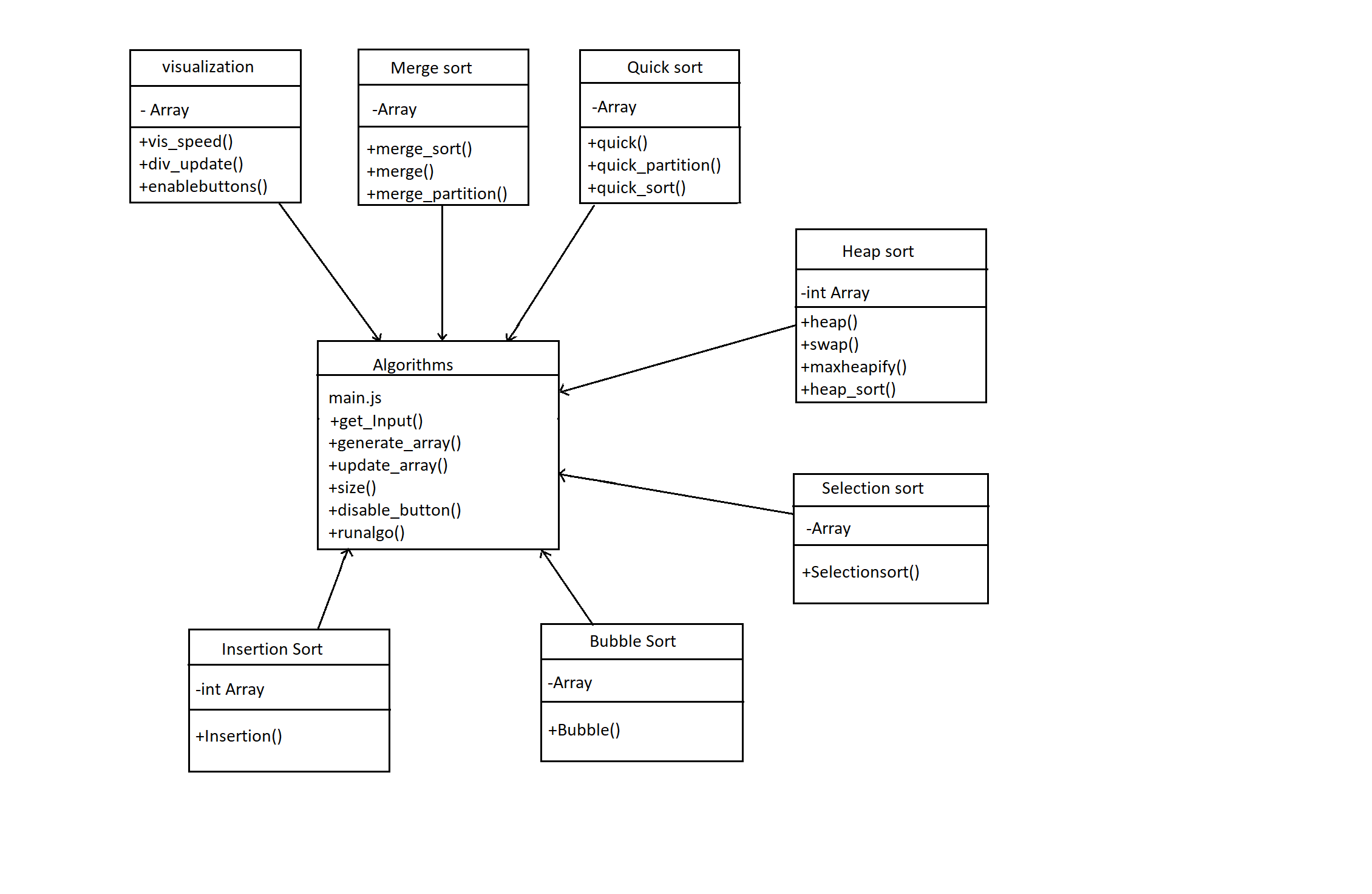
Ethernet connection (LAN) OR a wireless adapter (Wi-Fi)

Hard Drive: Minimum 32 GB; Recommended 64 GB or more

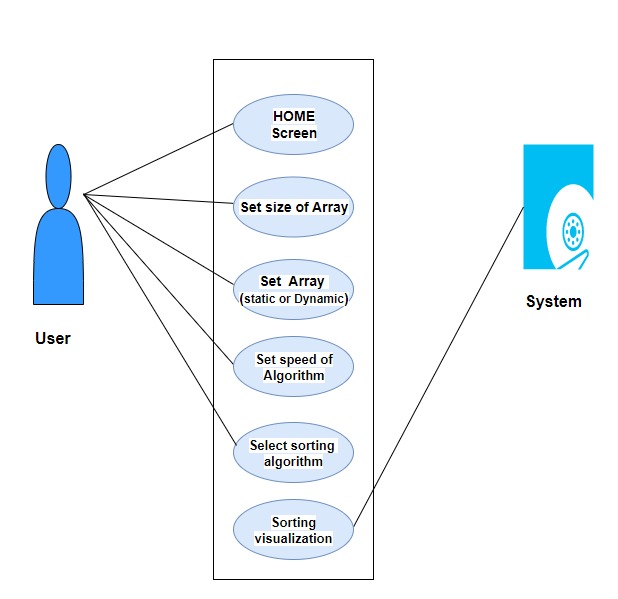
Memory (RAM): Minimum 4 GB; Recommended 8 GB or above

1. **UML DIAGRAMS**

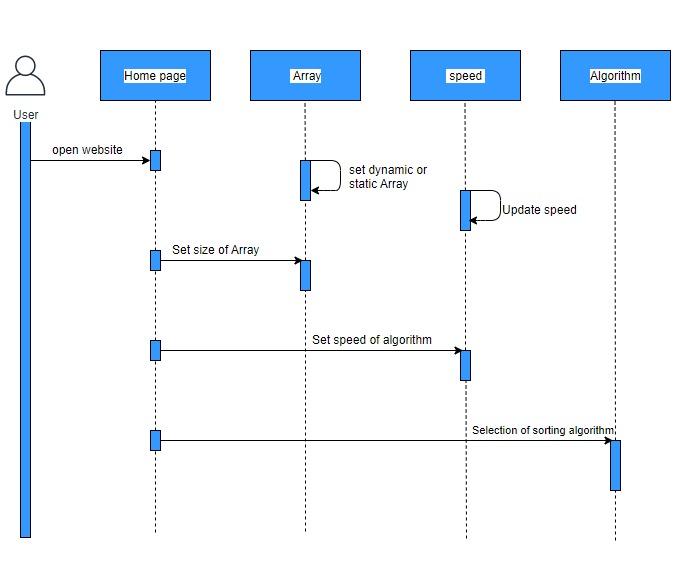
6.1 Class diagram:



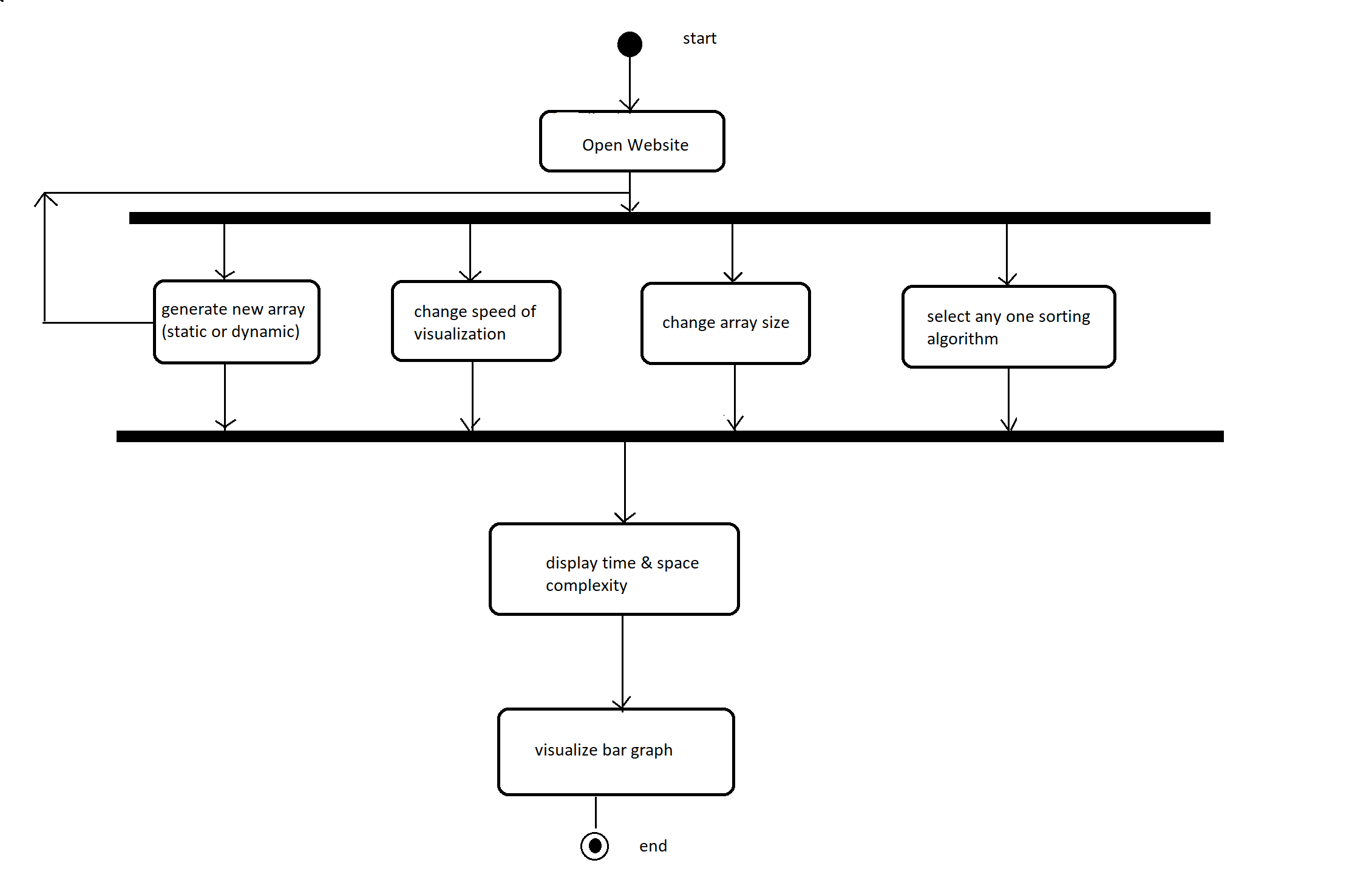
6.2 Use case diagram:



6.3 Sequence Diagram:



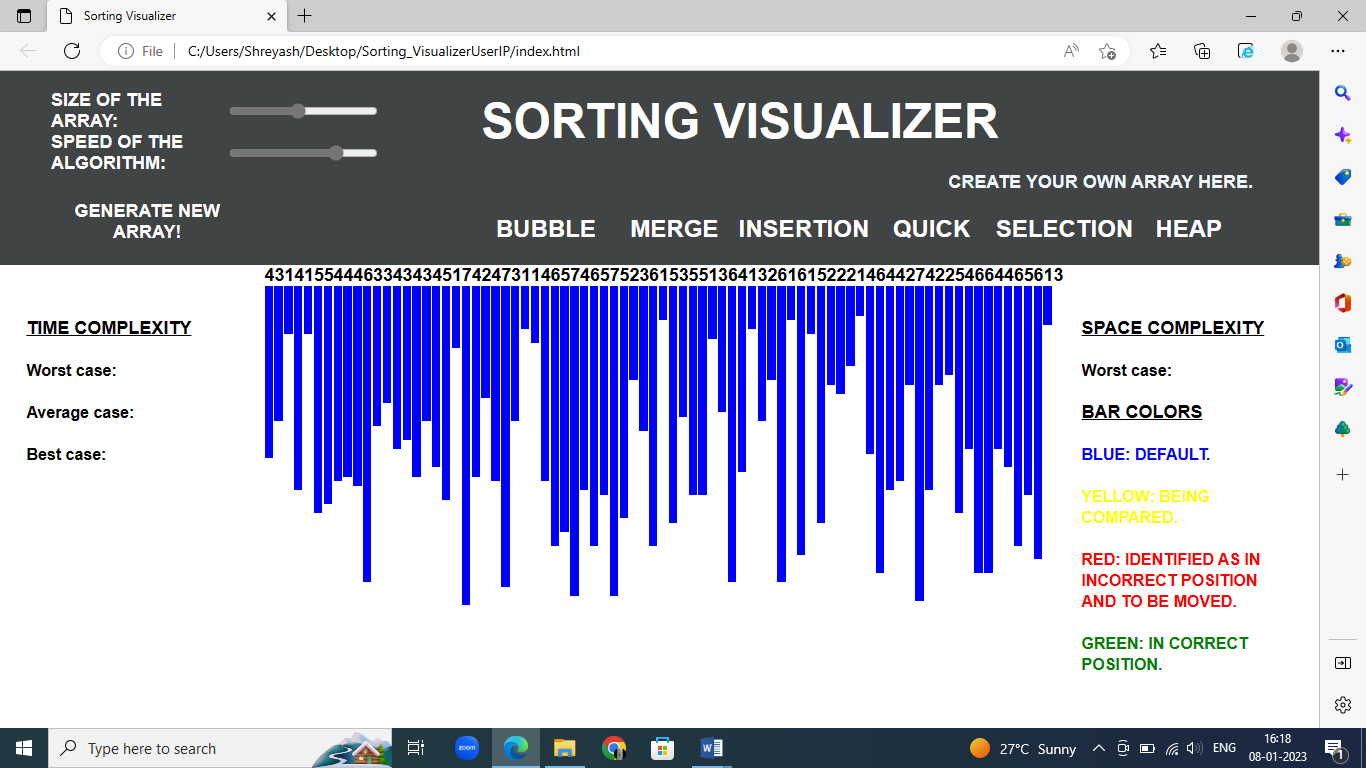
6.4 Activity Diagram:



1. **User Guide**

Step 1:

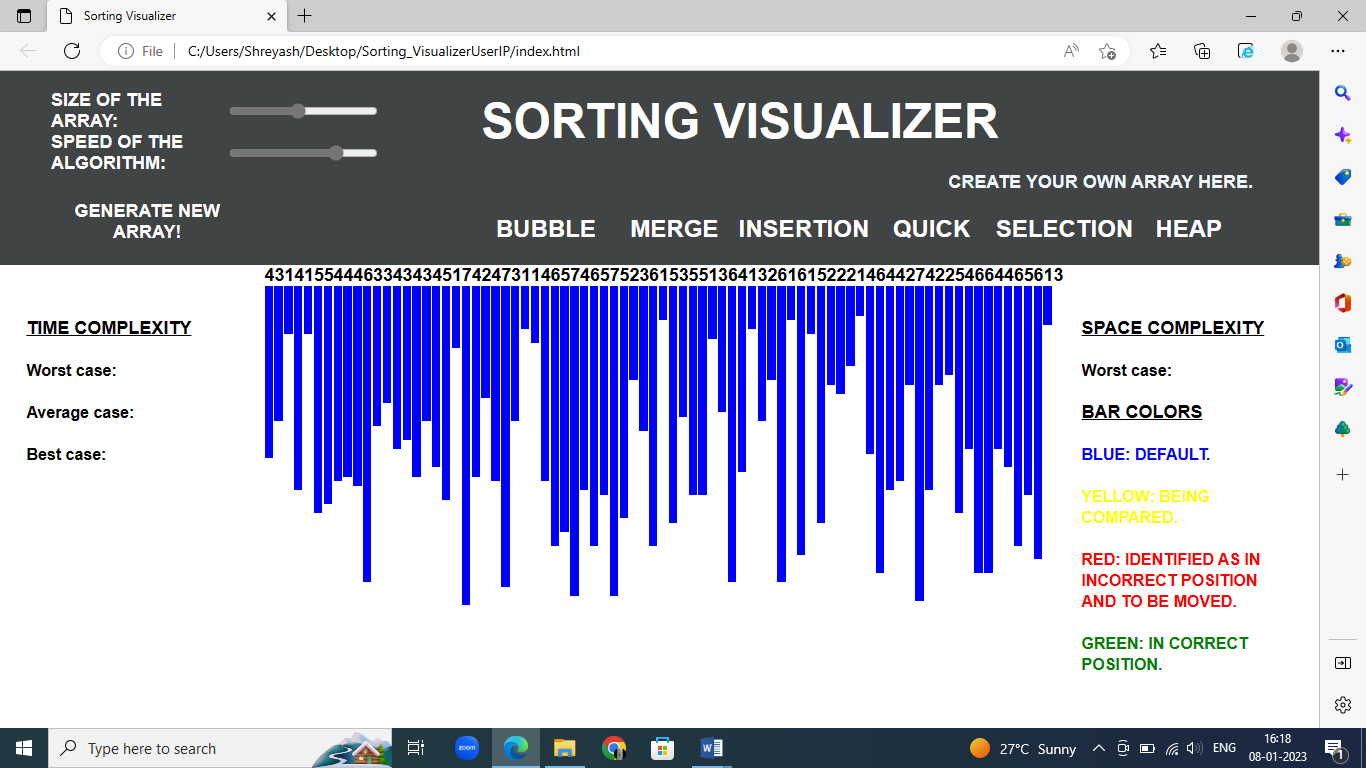
This is way website looks like



Step2:

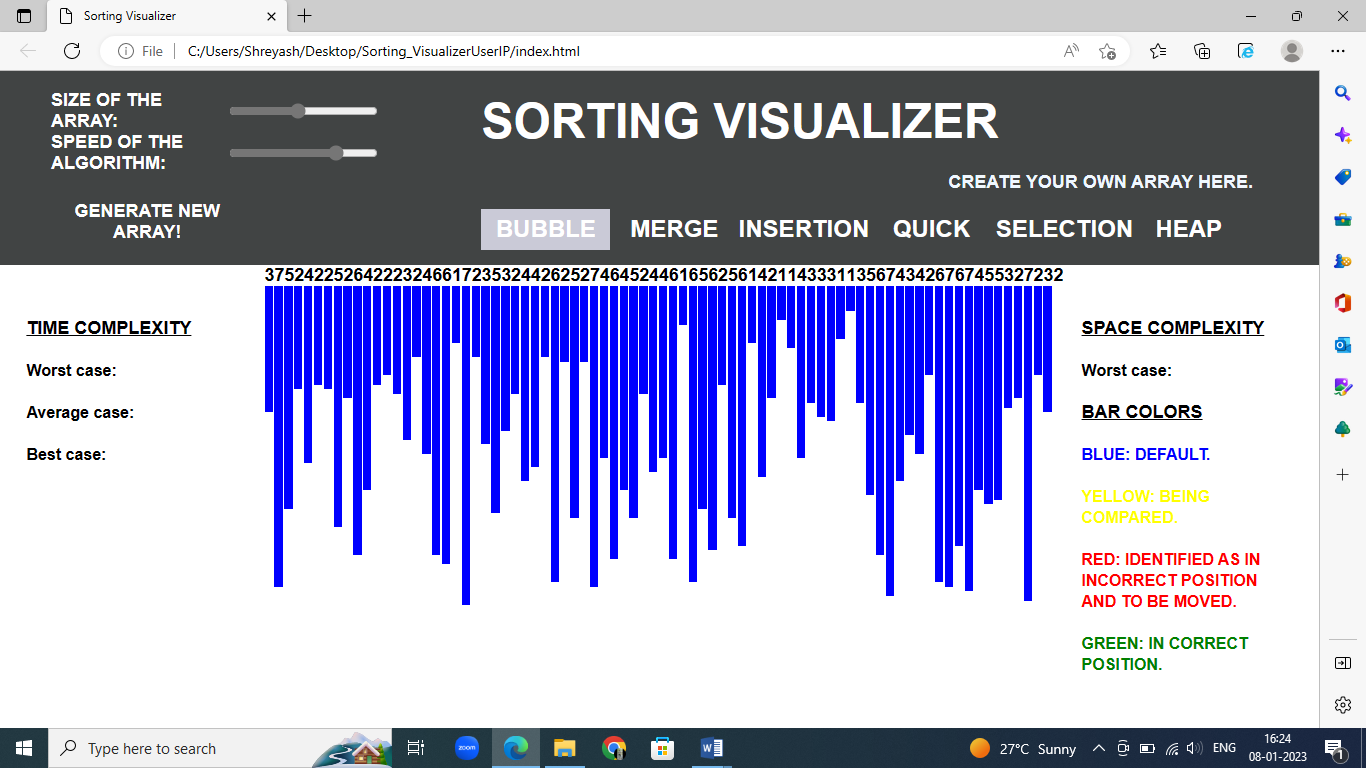
Change size of array and speed of visualizer here.

Also, you can generate new array.



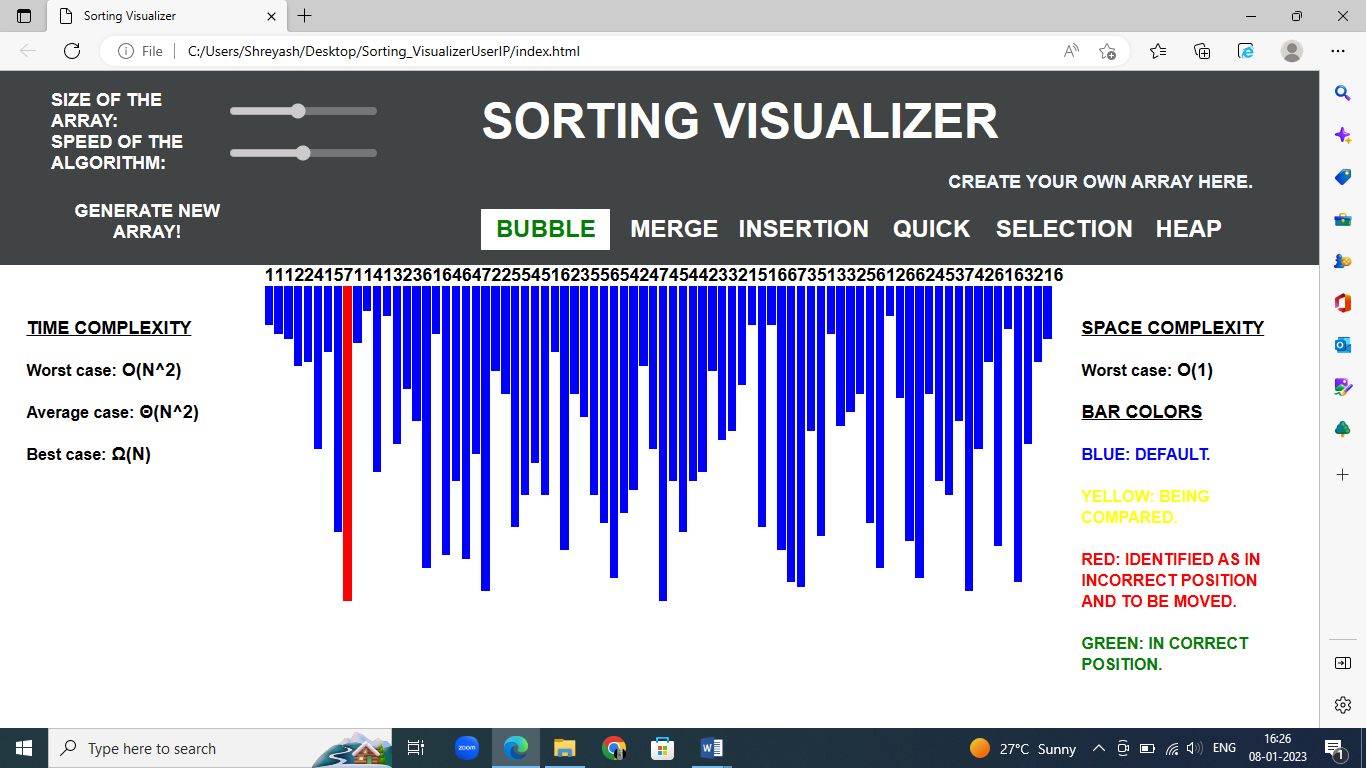
Step 3:

Select any sorting algorithm you want to visualize. Here for demo, we have chosen bubble sort.

****

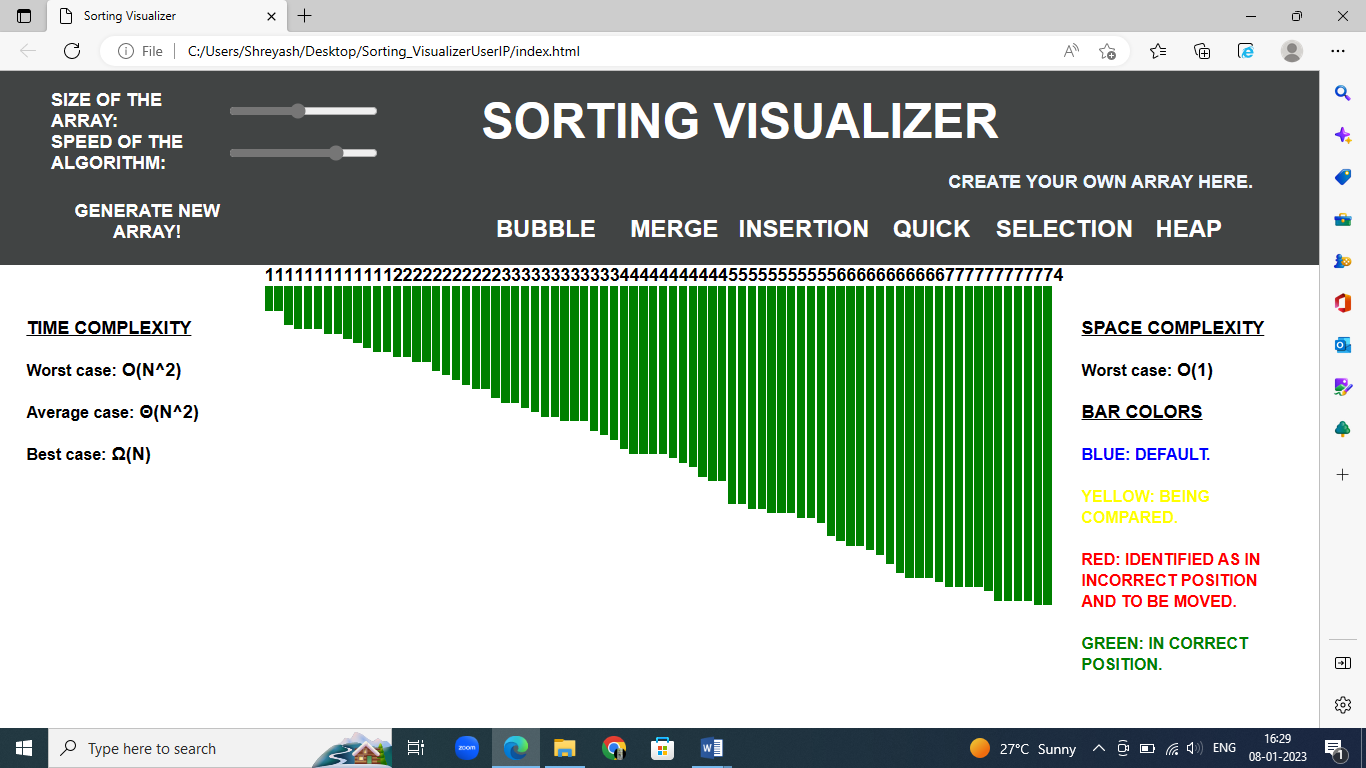
Step 4:

As soon you click on your sorting algorithm it starts running.

****

Step 5:

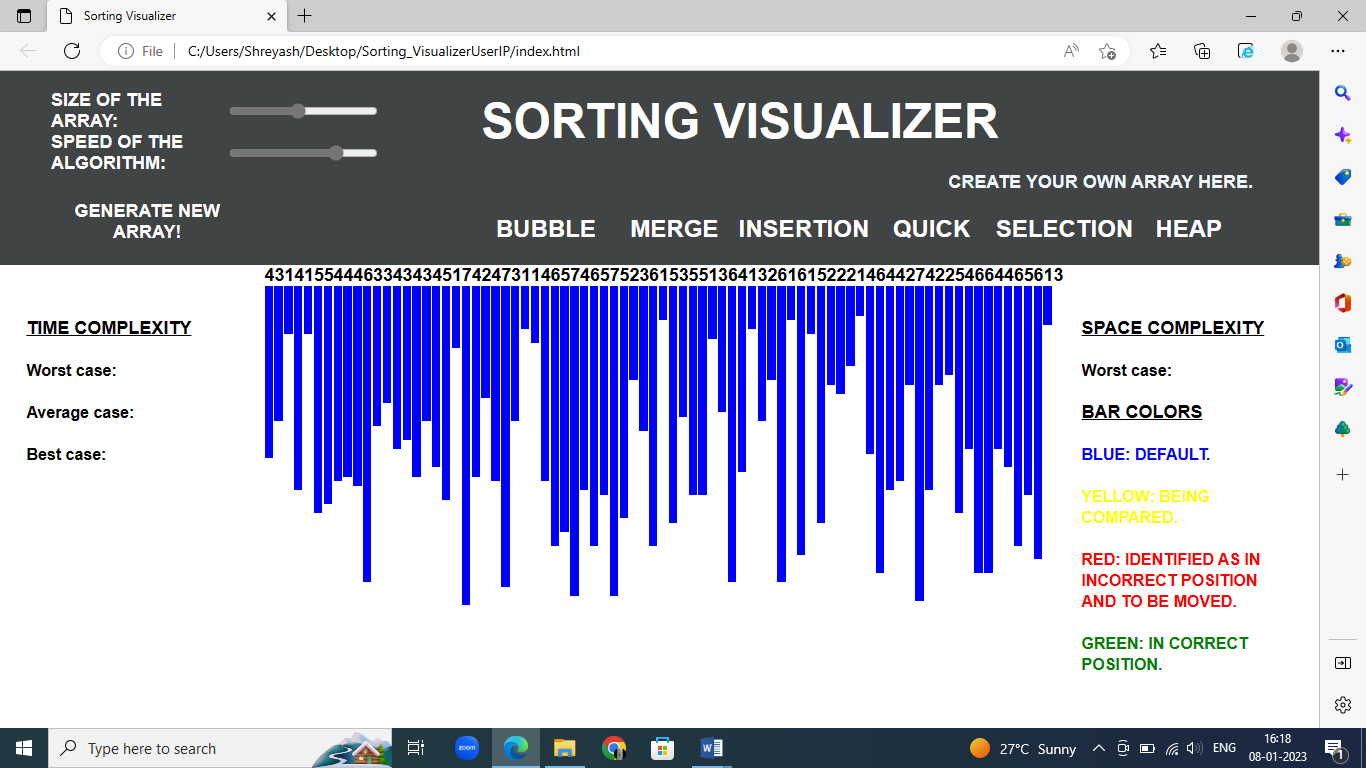
This is how your sorted array looks like after sorting.

****

Step for visualizations if user wants to create his own array

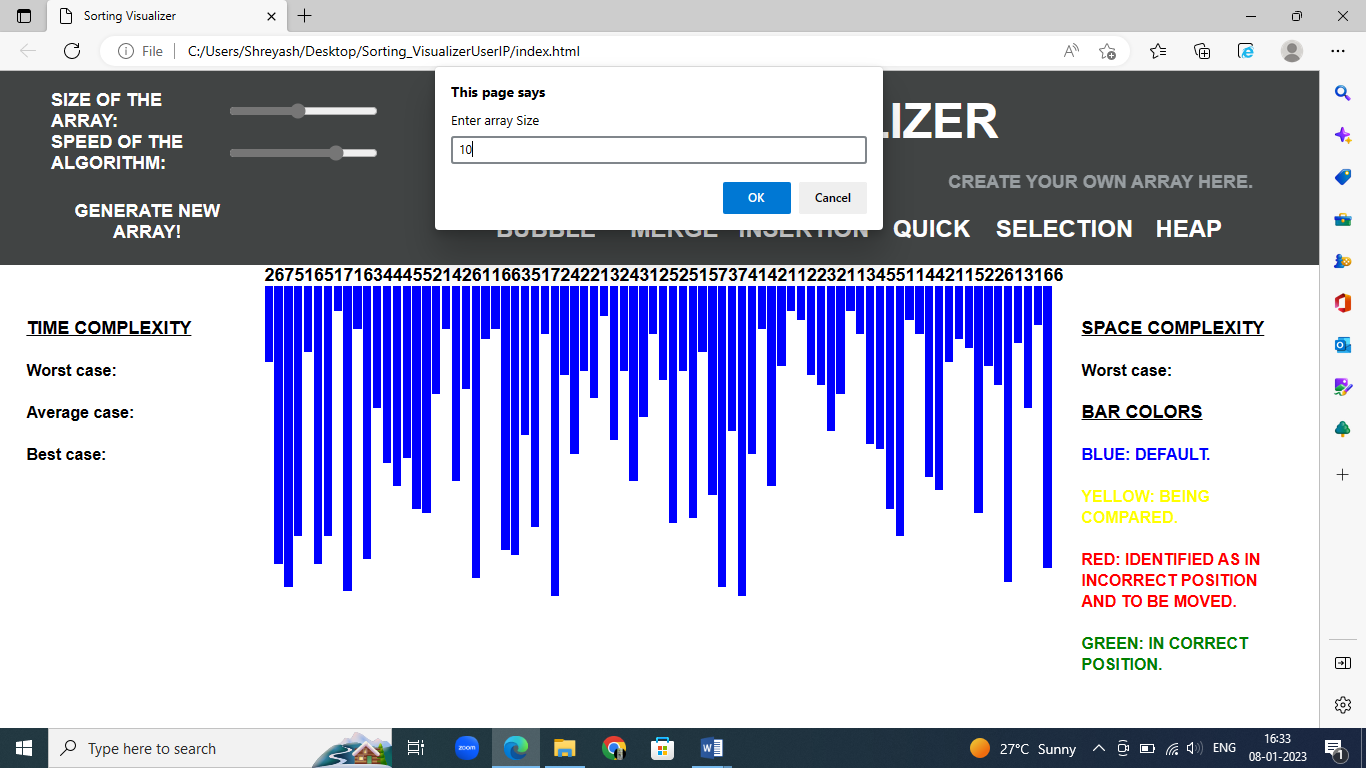
Step 1:

Click on “create your own array here”.



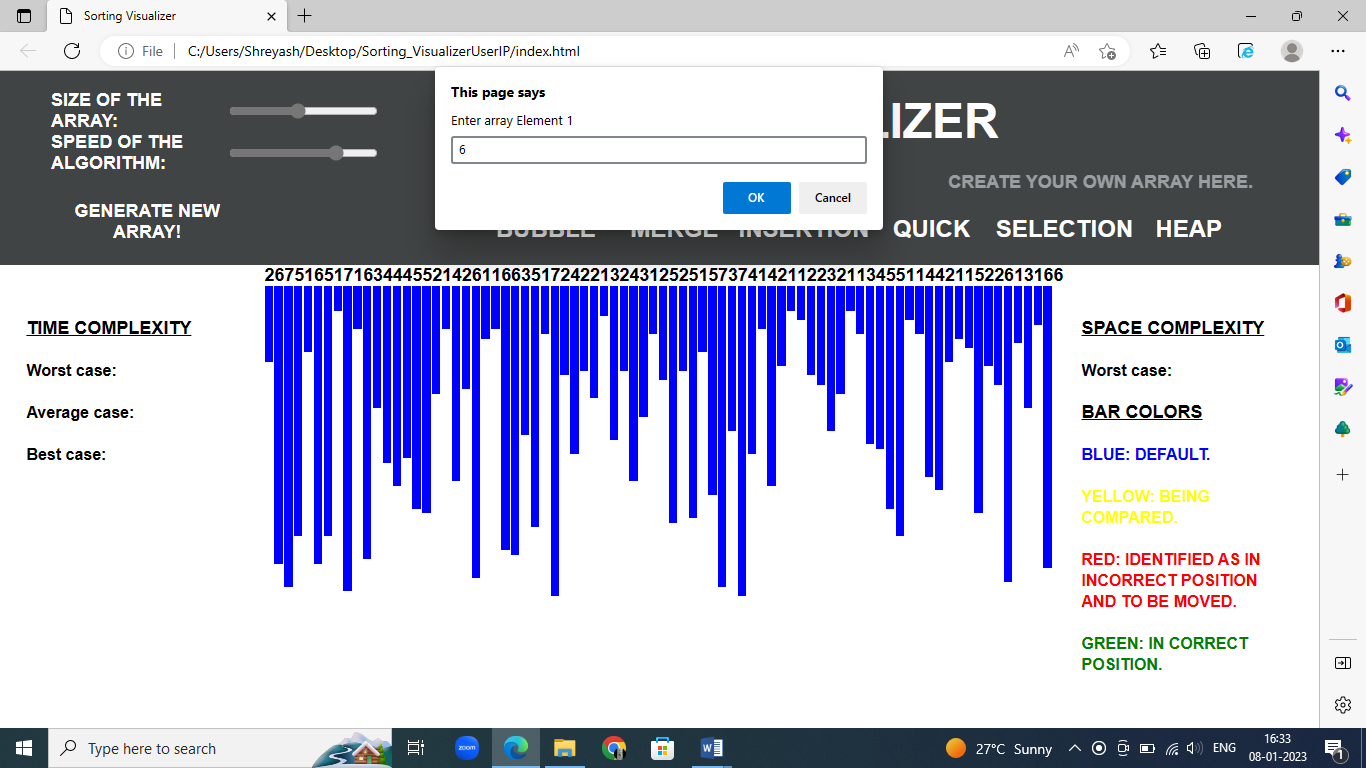
Step 2:

Enter the size of array.



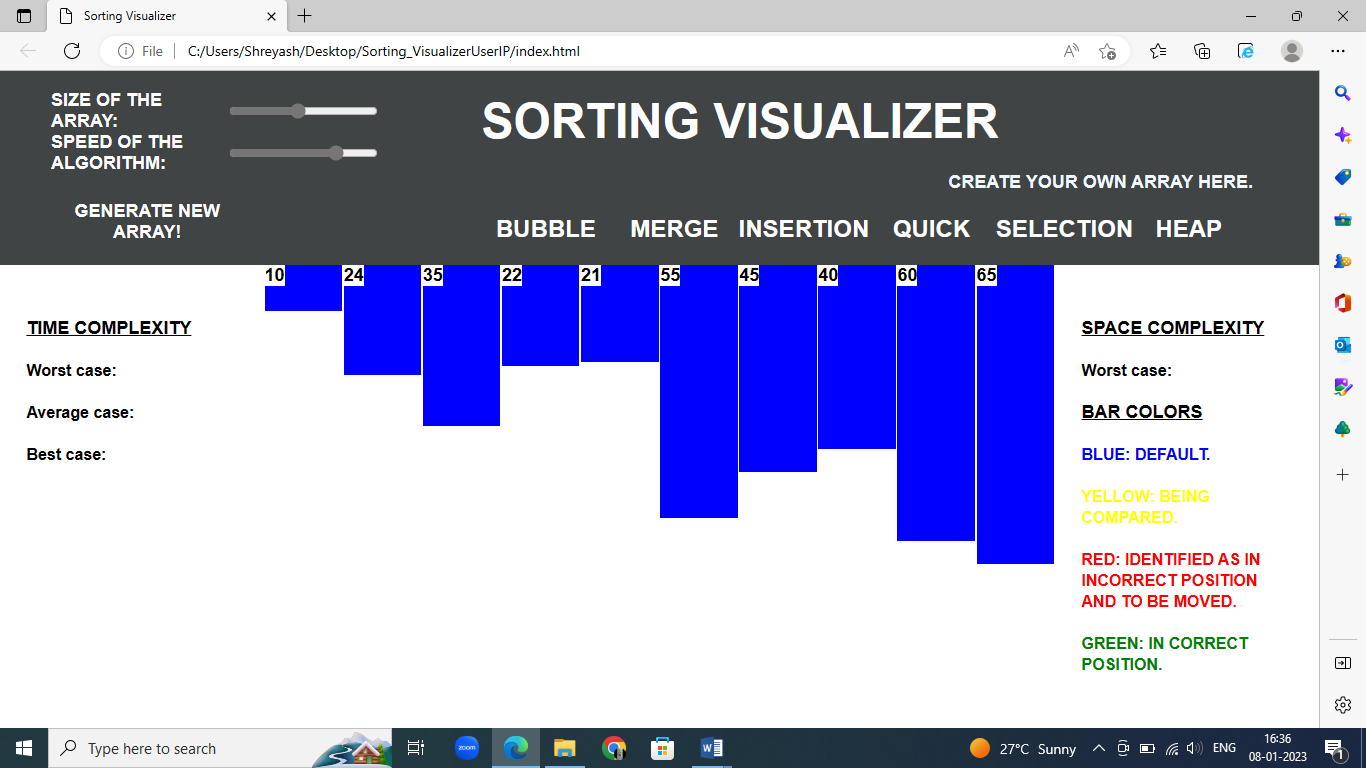
Step 3:

Enter array elements using enter key on your keyboard to enter next element. Then click “ok”.



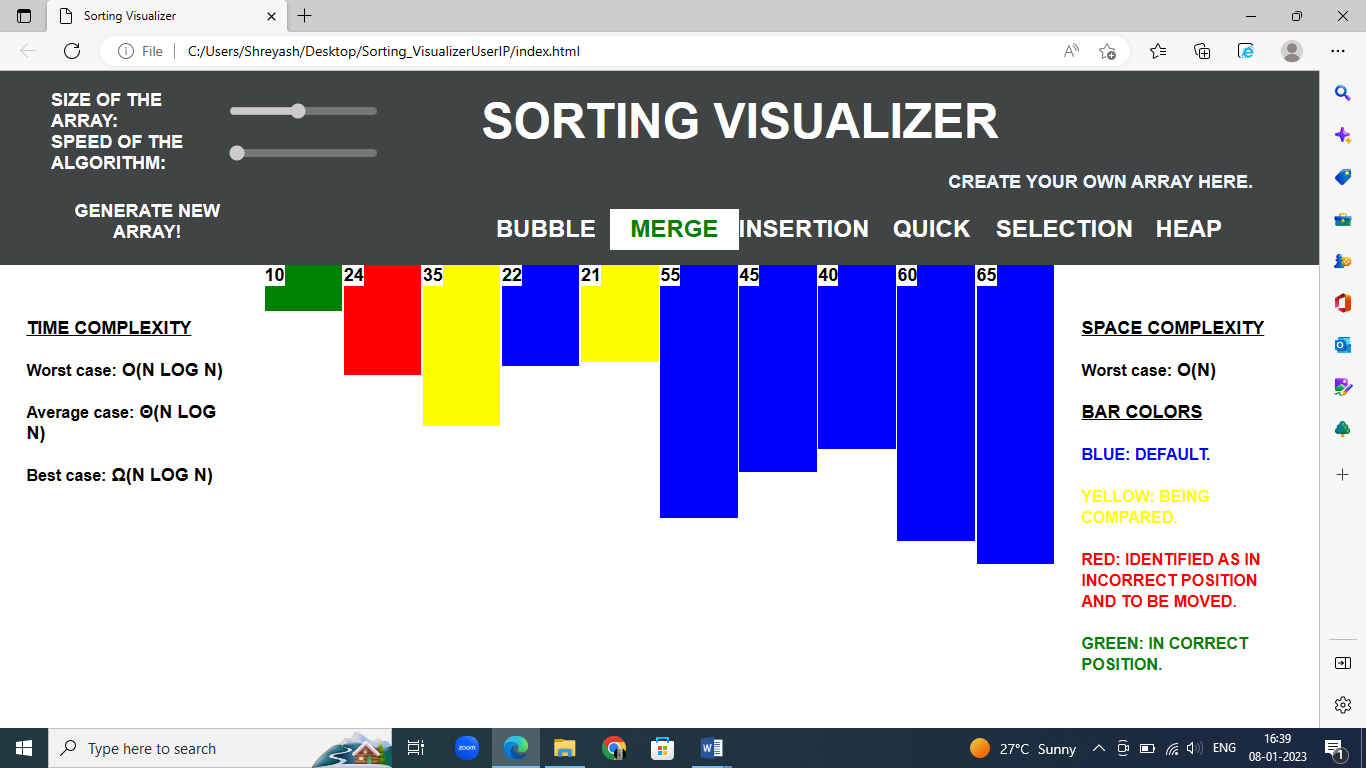
Step 4:

Your array will look something this.



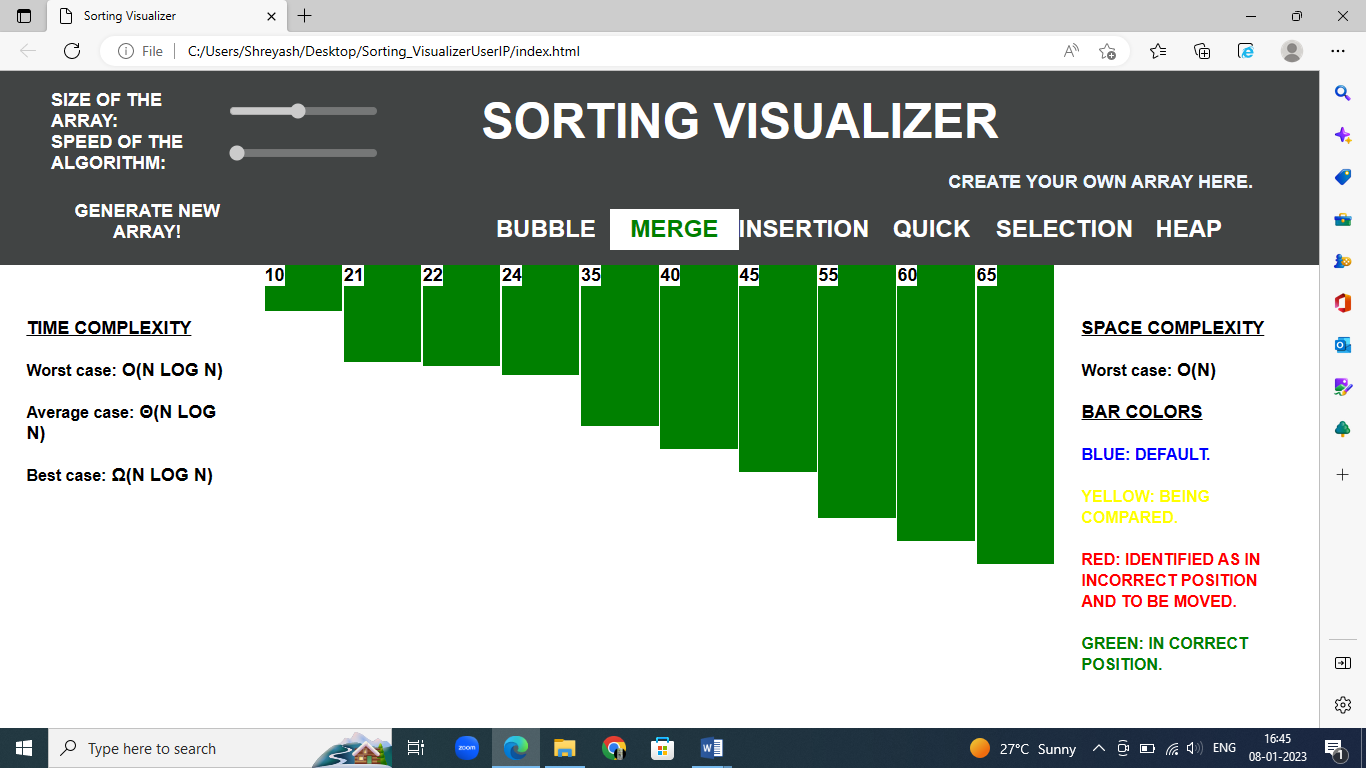
Step 5:

Choose any one sorting algorithm you want to visualize

****

Step 6:

Your sorted array will look like this after sorting.



1. **UNIT TESTING**

In order to perform an experiment to evaluate our animation tool, I recruited coworkers to act as test subjects and survey respondents. To that end, the plan was to expose her Computer Science Data Structures class to the tool and get them to take part in a survey comprising questions that gave the students the ability to write down what they observed and learnt, even if it was little. Despite the class size being of twenty-one students, only thirteen were present; fortunately, all of the students who had completed the survey were there. The set-up here was ideal since it addressed the audience for whom I was designing this course: a group of college students who were taking their first course in computer science and who needed assistance with algorithms. a) Results Start by arranging the data, and then pick the visualization algorithm to use. Algorithm buttons provide sorting of data as it arrives on the interface. Asking to specify the ordering of elements takes precedence because when the algorithm has completed running the initialization process, the interface is now showing a new ordering, while the code has already completed running the initialization with the prior data set. There was considerable confusion caused by the way the ordering buttons and algorithm buttons were shown in the UI after the surveys were completed. When beginning the sorting process, the student noted that she was having problems starting because she believed that she was hitting the buttons in the wrong order. This then led to her failing to execute the animation. Overall, my animation tool did not aid with the understanding of sorting algorithms. Among those who answered question 3, which questioned if their knowledge of a particular algorithm changed after using the tool, 5 of the 13 students (38%) stated that they had in some way altered their previous knowledge of the algorithm. Many thought the tool was a good concept, while the other 7 did not find it useful at all. It was said that one student stated a false positive about the instrument (whom I did not include in the 5 that said it was helpful).

A shortcoming of the animation is that it does not provide comparisons of the data's motions that result in such movements. Selection Sort's performance advantage over the other sorting algorithms is because there are O(n) swaps, which eliminates superfluous computer movements. Comparing the data produces a runtime complexity of O (n squared) (the slowest overall). In response to question 5, where students were asked for input and thoughts, another student stated that Merge Sort is the best of the four kinds. The average runtime of Merge Sort is O(n log2 n), which is the best average runtime among all sorting algorithms. Integrating visualization of comparisons as well as motions would help fix this. A good technique to accomplish this is to use an algorithm that highlights the bars in red when it is examining data, requiring additional time in the animation. The following sorting algorithms, Selection Sort and Bubble Sort, would require a considerable amount of comparisons in order to finish

1. **CONCLUSION AND FUTURE WORK**

This web-based animation tool for viewing the following sorting algorithms functions in great part because of all the time and effort that we invested into it. Despite its memory overhead, the feedback given to it was mostly good from the students that worked with it. This is consistent with my prior research, which revealed that there was no substantial difference in learning the content. What we do agree with totally is the attitude that holds there is a great need to investigate and produce animated presentations to enhance education in the classroom. Overall, we are not concerned that a large rework to a different language will be required soon because JavaScript is still one of the most popular web languages.

Following this, finally, we would make the online tool available to the public, with the feature I want most, which is to make it available to the public. This might be tough as well. The application that created the animation tool knows that it is available locally, but because of concurrency, it can serve numerous requests to the web site by separate users. As we try to figure out how to make the code as efficient as possible, we would need to spend some time thinking about how to make it work with numerous people using it. This would be excellent, as it would enable a form of comparison study.

1. **Bibliography**
2. Analysis of different sorting techniques URL: <https://www.geeksforgeeks.org/analysis-of-different-sorting-techniques/>
3. What is Big O notation explained URL: <https://www.freecodecamp.org/news/big-o-notation-why-it-matters-and-why-itdoesnt-1674cfa8a23c/>
4. JavaScript History URL:

[https://www.w3schools.com/js/js\_history.asp](https://www.w3schools.com/js/js_history.asp%20)

4. Node.js – Introduction URL: <https://www.tutorialspoint.com/nodejs/nodejs_introduction.html>