

**Sorting Algorithms Visualizer**

**Design and Analysis of algorithms**

**Section: E**

**Group Members:**

**Syed Muhib Ahmed (20K-0405)**

**Sarim Latif Khan(20K-1644)**

**SORTING ALGORITHM VISUALIZER**

**ABSTRACT**

Many students find Sorting Algorithms difficult because it requires abstract thinking. So, to ease up the hardships of students this idea of the project was formed. Our website Sim- ulation of Sorting Algorithms is both interactive and attractive to students. It is a web-based animation tool to visualise common sorting algorithms like Selection Sort, Bubble Sort, Insertion Sort, Quick Sort and Merge Sort. It gives the students hands on experience of the algorithms’ implementation. It feeds into their imagination to help them get a better understanding while also helping teachers to help make their students understandbetter. Through this project every student can learn at their own pace with their own speed. This interface is designed to make one feel fully engaged and concentrated. We have made use of HTML, CSS and JavaScript language for our project. The aimof this project is to make learning less of a burden and more ofan incredible experience which leaves students with the want to learn more

**INTRODUCTION**

When we talk about complex subject topics like Algorithms, it becomes extremely necessary for students to have a strong grip over the topic as it would form the foundation of their computational thinking and programming skills. We had observed that through conventional methods of teaching, it becomes a little difficult for students to understand the concept and also for teachers to explain their thoughts. a): So, we developed a method of learning through visualization and hand-on experience over different sorting algorithms which is bound to help the students and teachers. Good visualizations bring algorithms to life by graphically representing their various states and animating the transitions between those states. Visualization allows the human visual system to extend human intellect; we can use it to better understand these important conceptual processes, other things too.

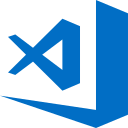
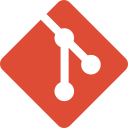
**DESIGN**

The programming design of the project includes HTML, CSS and Bootstrap for the creation of user interface and JavaScript was used at the back to visualize the data bars and calculate the time taken for each function to run

****. ** **

**Experimental Setup**

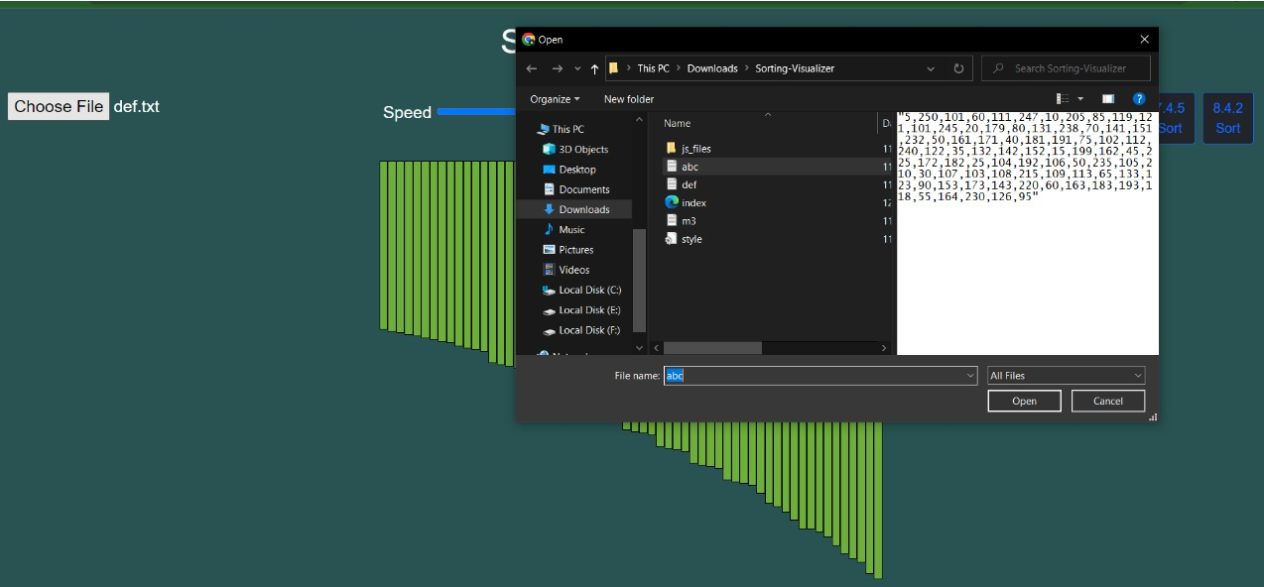
The code editor used to build this project was Visual Studio code and the version control system used to coordinate with the team members was Git for better understanding of the changes made by each member and making a clear label on the work done by each member.

** **

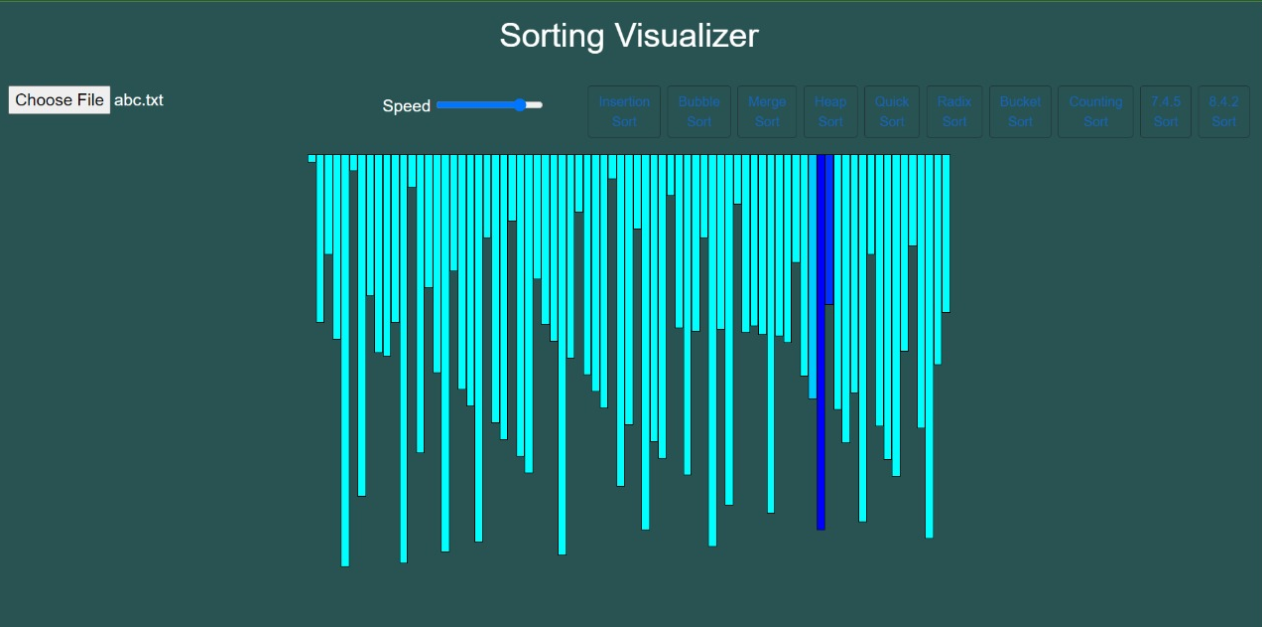
**Results**

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

**Selecting file**



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 due to the fact that 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.



**Conclusion**

In a nutshell, we identifies some issues by experiencing them ourselves in the present learning strategies in use and we tried to help better the scenario for aspiring students in this domain through or progressive web application. When we ourselves were learning the subject of algorithms in our curriculum, we found it a bit difficult to relate and understand the practical implementation of the algorithms owing to the difficulty in communication of the concepts from the teachers to the students. We found that there were no proper means that the teachers could adopt to portray their ideas in a better and easy manner in front of the students. So, we built an application which could help in the following ways:

* It has been found that it becomes easier for humans to retain the concepts when learnt through visuals than just textual or speech explanations.
* Application is extremely user friendly so people of any age can engage and start learning new things right away.

**REFERENCES**

1. “E-learning Tool for Visualization of Shortest Paths Algorithms” by Daniela Borissova and Ivan Mustakerov, ResearchGate, July 2015.
2. “Algorithm Visualization: The State” of the Field by Clifford A. Shaffer, Matthew L. Cooper, Alexander Joel D. Alon, Monika Akbar, Michael Stewart, Sean Ponce and Stephen H. Edwardsacm Transactions on Computing Education, Vol. 10, No. 3, Article 9, Pub. date: August 2010.
3. “Visualizing sorting algorithms” by Brian Faria, Rhode Island College, 2017.
4. “ViSA: Visualization of Sorting Algorithms” by Tihomir Orehovački, ResearchGate, May 2012.