A Project/Dissertation Review Report

on

SORTING VISUALIZER

Submitted in partial fulfillment of the requirement for the award of the degree of

Computer Science and Engineering in Cloud Computing Virtualization

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under the Supervision of

Dr. Ganga Sharma
Assistant Professor

Submitted By:

Neha Merlin Lobo (18021050138)
Supriya Giri (18021011495)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING / DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
DECEMBER , 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled " **Sorting Visualizer** " in partial fulfillment of the requirements for the award of the **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the **School of Computing Science and Engineering** of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY,2021** to **DECEMBER,2021** under the supervision of **Dr. Ganga Sharma, Assistant Professor**, **Department of Computer Science and Engineering**, of School of Computing Science and Engineering , Galgotias University, Greater Noida .

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

18SCSE1050034-NEHA MERLIN LOBO

18SCSE1010257-SUPRIYA GIRI

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Ganga Sharma

Assistant Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **18SCSE1050034-NEHA MERLIN LOBO, 18SCSE1010257- SUPRIYA GIRI** has been held on ________________ and his/her work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING.**

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December,2021

Place: Greater Noida

ACKNOWLEDGEMENT

Apart from the efforts of our, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

We would like to show my greatest appreciation to “Dr. Ganga Sharma” and also our dean “Dr. Munish Sabarwal”. We can’t say thank you enough for his tremendous support and help. We feel motivated and encouraged every time, We attend his meeting. Without his encouragement and guidance this project would not have materialized.

The guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project. We are grateful for their constant support and help.

ABSTRACT

We have learnt sorting algorithms like bubble sort, selection sort, insertion sort, quick sort. But often we fail to understand the core idea of a particular algorithm maybe because we are unable to visualize how they work. So the most important thing to understand about these algorithms is visualization. That’s why we are making this project to let everyone understand how these algorithms work and through this project you also will get a deep understanding of such sorting algorithms. At the end of this project you will have an immense grip on some core concepts of Javascript as well. Algorithm analysis and design is a great challenge for both computer and information science students. Fear of programming, lack of interest and the abstract nature of programming concepts are main causes of the high dropout and failure rates in introductory programming courses. With an aim to motivate and help students, a number of researchers have proposed various tools. Although it has been reported that some of these tools have a positive impact on acquiring programming skills, the problem still remains essentially unresolved. This project describes Sorting Visualizer, a tool for visualization of sorting algorithms. Sorting Visualizer is an easy-to-set-up and fully automatic visualization system with step-by-step explanations and comparison of sorting algorithms. Design principles and technical structure of the visualization system as well as its practical implications and educational benefits are presented and discussed.

TABLE OF CONTENT

LIST OF FIGURES

INTRODUCTION

Creating a web application using HTML, CSS, Javascript to visualize how various sorting algorithms work. This project's functionality will be similar to this application.

We have learnt sorting algorithms like bubble sort, selection sort, insertion sort, quick sort. But often we fail to understand the core idea of a particular algorithm maybe because we are unable to visualize how they work. So the most important thing to understand about these algorithms is visualization.

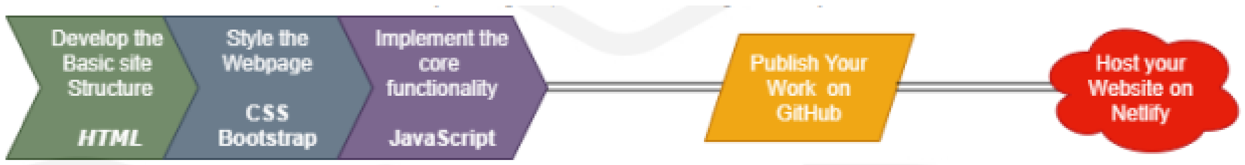
That's why we are making this project to let everyone understand how these algorithms work and through this project you also will get a deep understanding of such sorting algorithms.

This project is a good start for beginners and a refresher for professionals who have dabbled in data structures and algorithms using Javascript before and also web developers. The methodology can be applied to showcase any algorithm of one's choosing, so feel free to innovate.

High-Level Approach

- Creating the website's User Interface (UI) using HTML, CSS and enhancing it further using Bootstrap; without actually implementing any of the app's core features.
- Implementation of animations, effects and core functionalities (sorting algorithms) using JavaScript.

- Publish to GitHub and host your project live using Netlify.


Develop the Basic site Structure
HTML
Style the Webpage
CSS Bootstrap
Implement the core functionality
JavaScript
Publish Your Work on GitHub
Host your Website on Netlify

LITERATURE SURVEY

It is web-based App that can be used to create a web application using HTML, CSS, Javascript to visualize how various sorting algorithms work. This project's functionality will be similar to this application.

Pre-requisites

1. Code editor VSCode
2. HTML
3. CSS
4. JS

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.Visual Studio Code was first announced on April 29, 2015, by Microsoft at the 2015 Build conference.

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python and C++.It is based on the Electron framework,which is used to develop Node.js Web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports a number of programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette. Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support.A notable feature is the ability to create extensions that add support for new languages, themes, and debuggers, perform static code analysis, and add code linters using the Language Server Protocol.

Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

There are many web languages available, however we’re just going to look at three of them. They are HTML, CSS and JavaScript and they are considered to be the backbone of the web. When it comes to web development there is front-end web development and back end-web development. These three languages are for front-end web development and are responsible for what you can see and do on a website. They are referred to as client side languages as they run in the browser (Google Chrome, Firefox etc.) of your computer. The browser translates these languages and the result of this translation is the visual web page.

It’s important to note HTML and CSS are not considered to be programming languages. HTML is a markup language and CSS is a styling language. JavaScript, however, is a programming language. Hence, they are all web languages, but they perform different jobs.

HTML

- The Body
- HyperText Markup Language (HTML)
- Content and basic structure
- Describes and defines
- Made up of tags
- Tells the browser what to display

Hyper Text Markup Language (HTML) can be broken down into Hyper Text, which is what grants access to other texts through links, and Markup which outlines the basic structure and appearance of raw text. What this means is that HTML describes and defines the content and basic structure of the website. It does this through a means of special tags or codes which tell the browser what to do. HTML is the bare basics of a website.

HTML or Hyper Text Markup Language is the main markup language for creating web pages and other information that can be displayed in a web browser.

HTML is written in the form of HTML elements consisting of tags enclosed in angle brackets (like <html>), within the web page content.

HTML tags most commonly come in pairs like <h1> and </h1>, although some tags represent empty elements and so are unpaired, for example <img>. The first tag in a pair is the start tag, and the second tag is the end tag (they are also called opening tags and closing tags). In between these tags web designers can add text, further tags, comments and other types of text-based content.

The purpose of a web browser is to read HTML documents and compose them into visible or audible web pages. The browser does not display the HTML tags, but uses the tags to interpret the content of the page.
HTML elements form the building blocks of all websites. HTML allows images and objects to be embedded and can be used to create interactive forms. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. It can embed scripts written in languages such as JavaScript which affect the behavior of HTML web pages.

An HTML only website can be compared to a functioning human body. Note, I didn’t say fully-functional. An HTML only website has all of its body parts, although it doesn’t offer much to look at because it doesn’t have any accessories or personal style. At this stage, it’s also a body which is not capable of moving or speaking. A website which consists of only HTML would probably look a little like this:

This is where CSS comes in.

CSS

CSS stands for Cascading Style Sheets. It is the language for describing the presentation of Web pages, including colors, layout, and fonts, thus making our web pages presentable to the users.

CSS is designed to make style sheets for the web. It is independent of HTML and can be used with any XML-based markup language. Now let’s try to break the acronym:

- Cascading: Falling of Styles
- Style: Adding designs/Styling our HTML tags
- Sheets: Writing our style in different documents

CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML

- The Accessories
- Cascading Style Sheet (CSS)
- Gives style and structure to the content
- Link the CSS file to the HTML
- Tells the browser how to display

A Cascading Style Sheet is the website's accessories. It's responsible for outlining the colors, font and positioning of the content on a website. It adds some style and structure to the content. In order to make use of the CSS capabilities it needs to be linked within the HTML content so that style can be added to the website. CSS will tell the browser how to display the existing HTML.

CSS can be compared to adding personal style to the body. When you link CSS to HTML, it's like dressing up the body. For example, you can choose a specific color shirt and match it with a specific color pair of trousers. On a website, you can choose the color of the background or the font size of a heading and much more. It's important to note that CSS cannot live without HTML as there would be nothing to style. Just like clothes or shoes would be pointless without someone to wear them.

So by now you should have an understanding of how structure and style are constructed on a website. A website that consists of HTML and CSS might looks like this:

However, you can’t help but notice that something is missing. The web page is lacking certain functions like a search box or options to comment. Right now the body, with all its accessories, looks more like a mannequin in a store window than a real human being.

That’s where JavaScript comes in.

JavaScript

JavaScript (often shortened to JS) is a lightweight, interpreted, object-oriented language with first-class functions, and is best known as the scripting language for Web pages, but it's used in many non-browser environments as well. It is a prototype-based, multi-paradigm scripting language that is dynamic, and supports object-oriented, imperative, and functional programming styles.

JavaScript runs on the client side of the web, which can be used to design / program how the web pages behave on the occurrence of an event. JavaScript is an easy to learn and also powerful scripting language, widely used for controlling web page behavior.

Contrary to popular misconception, JavaScript is not "Interpreted Java". In a nutshell, JavaScript is a dynamic scripting language supporting prototype based object construction. The basic syntax is intentionally similar to both Java and C++ to reduce the number of new concepts required to learn the language. Language constructs, such as if statements, for and while loops, and switch and try ... catch blocks function the same as in these languages (or nearly so).

- The body’s ability to perform actions
- JavaScript is not Java
- Behaviour of the website
- Used for interactive functionality
- Allows for the user to interact with the browser

JavaScript controls the behaviour of the website. It’s important to note that JavaScript and Java are two different things. JavaScript was designed to manipulate web pages and it is used to create interactive functionality. Without JavaScript a website will still be functional, but in a limited way. JavaScript is what animates HTML and CSS, and it’s what brings your website to life.

JavaScript can be compared to the body’s ability to perform actions such as walking or talking. So when you add JavaScript to HTML and CSS, it transforms the body from being a beautifully dressed mannequin into a real-life walking talking human being. It animates the body, giving it lifelike qualities. JavaScript can also be compared to a fully functional body that has the ability to interact. As we all know, having an interactive website is critical, otherwise its just a boring page filled with information. Here we see a website which consists of HTML, CSS and JavaScript:

If you look at this example of twitter, JavaScript allows you to expand the tweet to see re-tweets, to set a tweet as a favourite and more. A popular JavaScript App is Google Maps.

Basic Requirements (Hardware)

- Processor: Minimum 2.0GHz requires.
- Ram: 4 GB.
- Hard Disk: 100 GB.
- Input device: Standard Keyboard and Mouse.
- Output device: VGA and High-Resolution Monitor.
- Operating System: Windows1

What's inside the website

1. Currently available sorting algorithms- Bubble sort, Selection sort, Insertion sort, Merge sort, Quicksort (I will plan to bring more algorithms in action to visualize & more changes).

2. You can change the size of the array

3. You can change the speed of the visualization

Algorithms:

Bubble Sort:

Bubble sort, sometimes incorrectly referred to as sinking sort, is a simple sorting algorithm that works by repeatedly stepping through the list to be sorted, comparing each pair of adjacent items and swapping them if they are in the wrong order. The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted. The algorithm gets its name from the way smaller elements "bubble" to the top of the list. Because it only uses comparisons to operate on elements, it is a comparison sort. Although the algorithm is simple, most of the other sorting algorithms are more efficient for large lists.

Selection Sort:

Selection sort is a sorting algorithm, specifically an in-place comparison sort. It has O(n2) time complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity, and it has performance advantages over more complicated algorithms in certain situations, particularly where auxiliary memory is limited.

The algorithm divides the input list into two parts: the sublist of items already sorted, which is built up from left to right at the front (left) of the list, and the sublist of items remaining to be sorted that occupy the rest of the list. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list. The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted sublist, exchanging it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

Insertion sort:

Insertion sort is a simple sorting algorithm that builds the final sorted array (or list) one item at a time. It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort. However, insertion sort provides several advantages:

Simple implementation

Efficient for (quite) small data sets Adaptive (i.e., efficient) for data sets that are already substantially sorted: the time complexity is O(n + d), where d is the number of inversions More efficient in practice than most other simple quadratic (i.e., O(n2)) algorithms such as selection sort or bubble sort; the best case (nearly sorted input) is O(n) Stable; i.e., does not change the relative order of elements with equal keys In-place; i.e., only requires a constant amount O(1) of additional memory space Online; i.e., can sort a list as it receives it.

When humans manually sort something (for example, a deck of playing cards), most use a method that is similar to insertion sort.

Quick sort:

Quicksort, or partition-exchange sort, is a sorting algorithm developed by Tony Hoare that, on average, makes O(n log n) comparisons to sort n items. In the worst case, it makes O(n2) comparisons, though this behavior is rare. Quicksort is often faster in practice than other O(n log n) algorithms.Additionally, quicksort's sequential and localized memory references work well with a cache. Quicksort is a comparison sort and, in efficient implementations, is not a stable sort. Quicksort can be implemented with an in-place partitioning algorithm, so the entire sort can be done with only O(log n) additional space used by the stack during the recursion.

There are eight tasks that are needed to complete the sorting visualizer project

<u>Task 1 :Getting Started</u>

First validate the idea by doing a low level implementation (Proof of concept) of the components involved in the project.

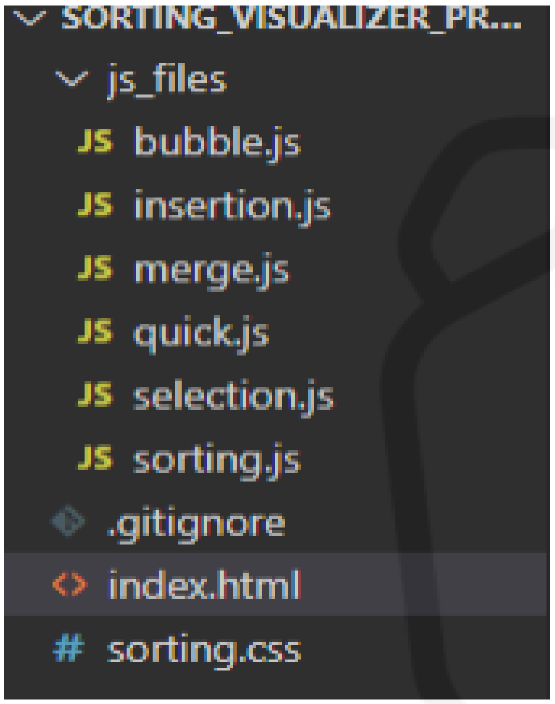
- Get more clarity around the unknowns. Eg: XML vs HTML and why to use HTML5/CSS3 not the other versions, advantages of using Bootstrap.

- Get a better understanding of the stages involved in the project. Eg: By doing a proof of concept you will understand that there are multiple stages such as creating the basic layout, styling it and implementing the functionalities.

Requirements

- This is a typical JavaScript Project , so you need a code editor like VScode (recommended), Atom, Sublime text, etc. with necessary plugins.

- Then create an appropriate project folder with essential files. It’s a good practice to follow the suggested file structure (shown below).



```
SORTING_VISUALIZER_PR...
  js_files
    bubble.js
    insertion.js
    merge.js
    quick.js
    selection.js
    sorting.js
  .gitignore
  index.html
  sorting.css
```

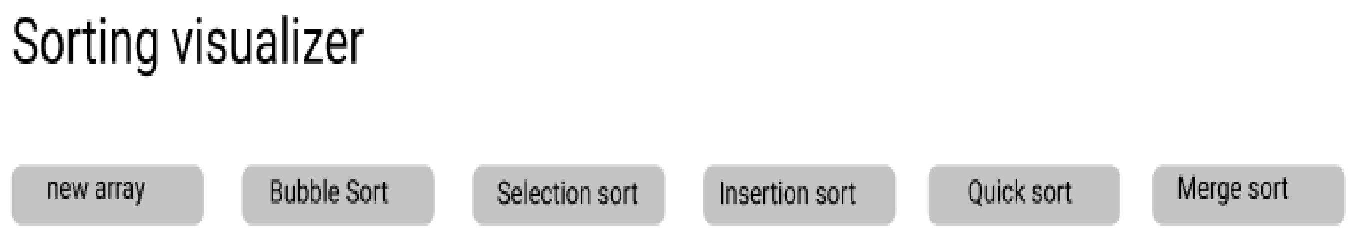
Task 2 : Create the website's UI

In this milestone the basic structure of this website will be made. In this milestone you will mainly use HTML. Then in the next milestone we will add Bootstrap and CSS for styling purposes.

Requirement

- First component of the website is to give a heading using the HTML heading tag.
- Then the main components are to create 5 buttons for running the sorting algorithms (bubble sort, selection sort, insertion sort, quick sort, merge sort) and another button to generate new arrays. Create all these buttons using the HTML button tag.
- And wrap them with the appropriate id's and classes which will then be used for reference in styling in CSS and to select them and also to add event listeners in Javascript code (to be done in the upcoming milestones).

Expected Outcome - Since only HTML has been used the site should look something like this.

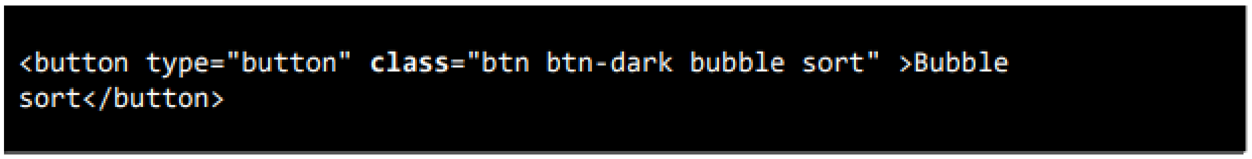


Task 3 :Improving UI using CSS and Bootstrap

The web app's basic skeleton UI was created in the previous task. To make the app more attractive and interactive we will employ CSS and Bootstrap for styling purposes.

Requirement

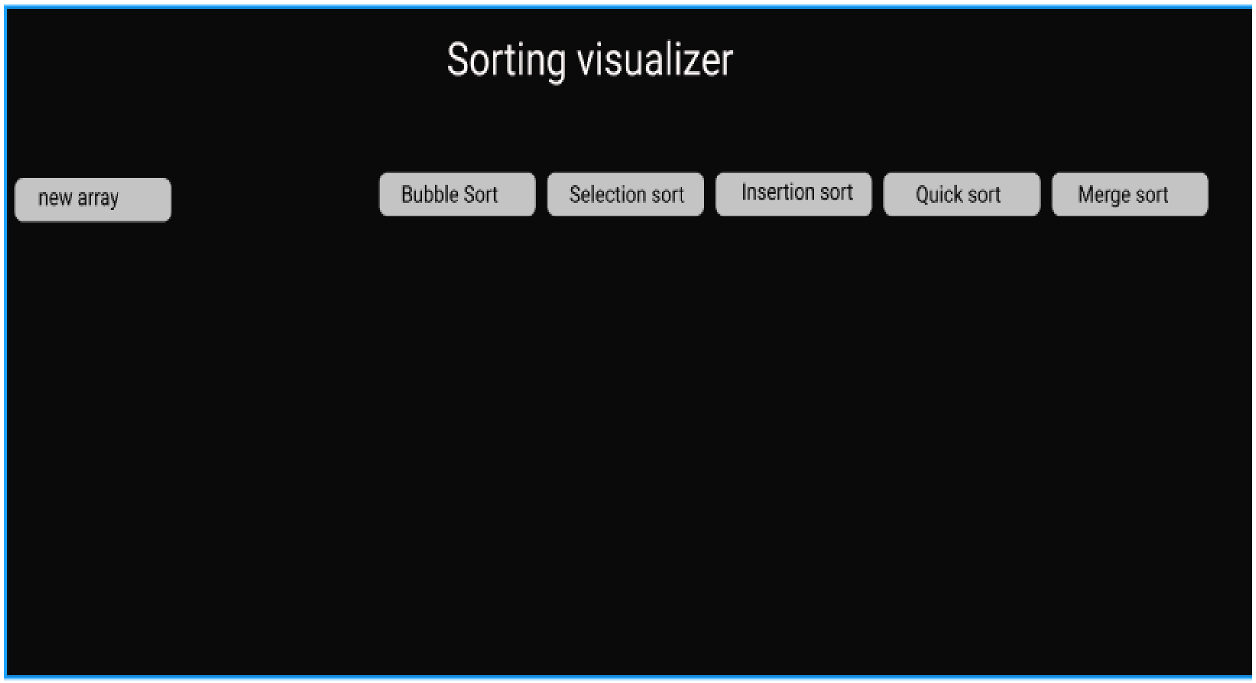
- Give a background color to the website using CSS.
- Use Bootstrap to add a navbar for the top part of the web app and inside this navbar class provide all the buttons.
- Give all the appropriate class names and id to all the relevant substructures like this (to be done in HTML code).



```
<button type="button" class="btn btn-dark bubble sort" >Bubble
sort</button>
```

- For styling purposes, you can refer to the image in the Expected Outcome section as your starter template. Do not think about the bars and other components except the buttons. Bars and other components will be addressed in the upcoming milestones. Feel free to innovate and come up with your own styles.

Expected Outcome - After styling using CSS and Bootstrap the site should look something like this.



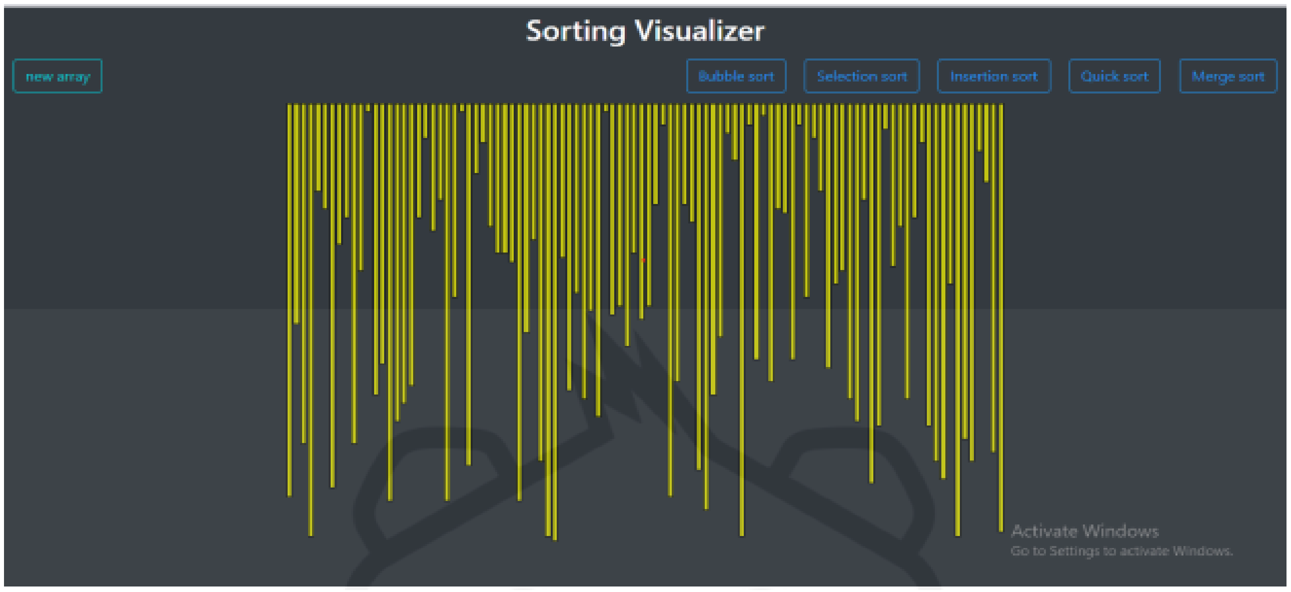
<u>Task 4 Creating Bars Using JavaScript</u>

From this milestone onwards we will start implementing the animations and other core functionalities of the application. In this milestone, we will create bars of different heights; which basically indicates the array that we will sort. Through these bars, we will visualize how sorting algorithms work.

Requirement

- In the JS file just create an array and push 100 numbers (Don’t worry we will implement the number of bars changing functionality in the upcoming modules).
- Create 100 numbers using a random function, convert those numbers to an integer number in the range 0-100 (you may take any range).
- The array integers should be the height of bars.
- Now inside the HTML file under the navbar create a division (div) and also give an id in this division where we will be placing all the bars' components.
- Now coming to the JS file we will create 100 div elements (creating elements using JS).
- Using JS add a particular class to all divs (so that we can add styles to all the div in CSS) and all divs will have different heights equal to array elements (choose an appropriate scale) (Changing the CSS property using JS).
- Push every bar in that particular div, defined in the HTML file under navbar using JS.
- Wrap all this in a function and make a call to that function.
- Also, add event listeners to the new array button and inside that call the function. So that you can use that button to create a new bar every time without refreshing the page.
- In the CSS file, you can add styles to the bars inside the class for bars.

Expected Outcome - After adding the bars the site looks something like this,

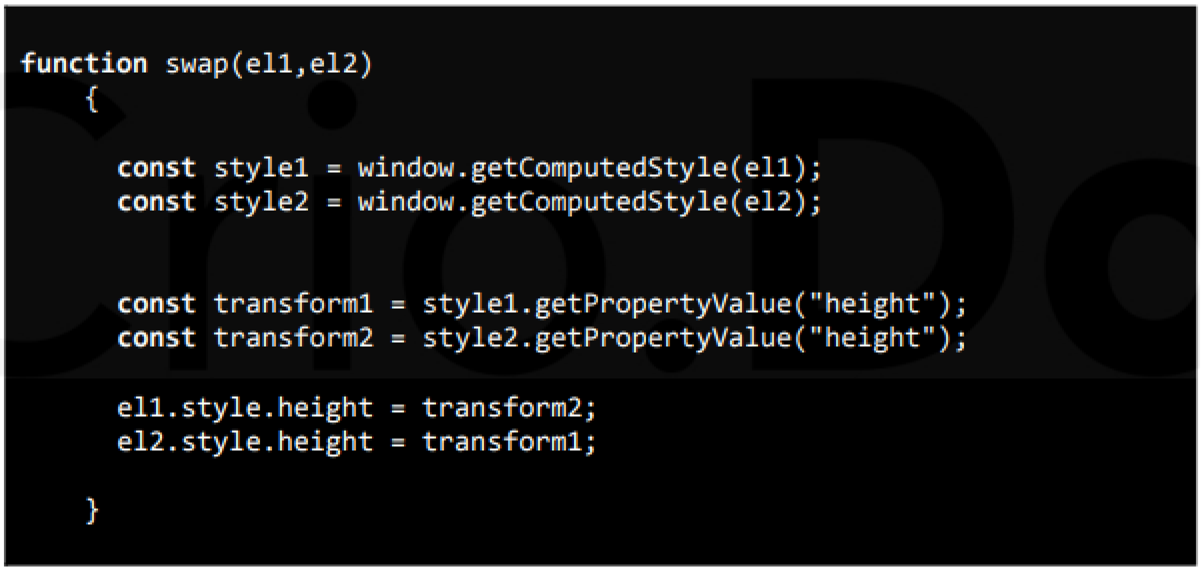


<u>Task 5 : Implementing Bubble Sort Algorithm</u>

Before starting this task, understand the Bubble Sort algorithm thoroughly.

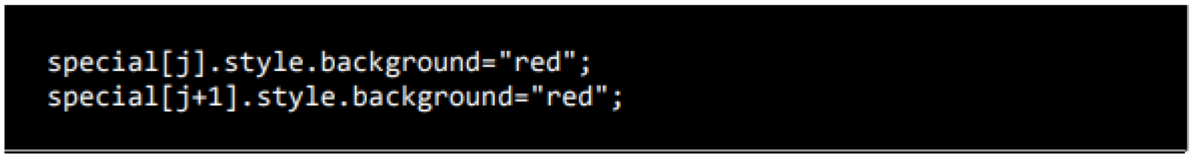
Requirements

- The most important thing to do in every sorting algorithm is to swap elements. To make swap two elements in HTML using JS you can do it this way.



```
function swap(el1,el2)
    {

      const style1 = window.getComputedStyle(el1);
      const style2 = window.getComputedStyle(el2);


      const transform1 = style1.getPropertyValue("height");
      const transform2 = style2.getPropertyValue("height");

      el1.style.height = transform2;
      el2.style.height = transform1;

    }
```

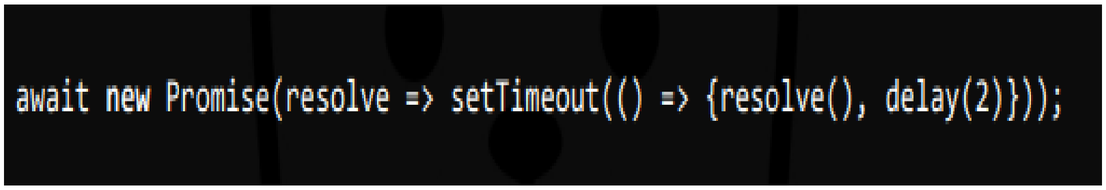
- Now apply the simple bubble sort algorithm. During the comparison of two elements make the background color red for both the bars and after the comparison convert the background color again to the default one for both the bars. You may use the following logic –



```
special[j].style.background="red";
special[j+1].style.background="red";
```

- At the end of every iteration when the highest bar will be taken to the right corner then to show that this bar is placed at its perfect position make the background color Green in the above way.

- Now when you run this you will notice that there is no delay in swapping and the other iterations. So you have to add a delay before the swaps in order to watch how changes are happening. For delay may use the following logic


```
await new Promise(resolve => setTimeout(() => {resolve(), delay(2)}));
```

- Wrap this whole thing in a function and pass this into the event listener of the bubble sort button.

Task 6 :Implementation of remaining Sorting function

Again before starting this task understand the Selection Sort, Insertion Sort, Quick Sort, Merge sort algorithms thoroughly.

Tips

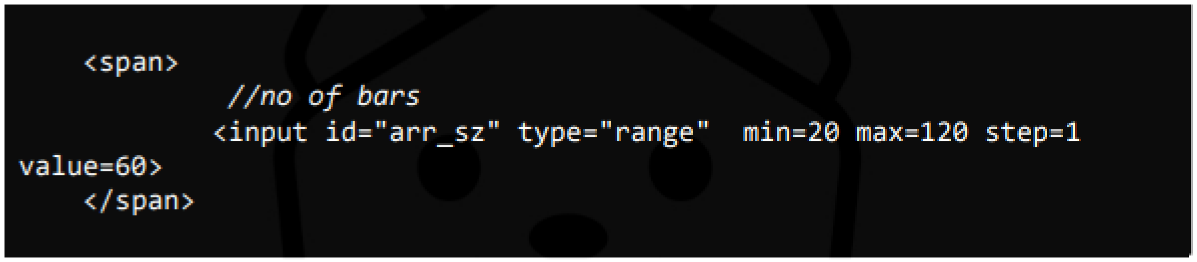
- These sorting algorithms' implementations are the same as for the Bubble sort algorithm. In every algorithm's implementation to distinguish every comparison, swaps and iterations just change the colors and animation effects in your own way.

Task 7 : Changing the number of bars and speed

Now as you must have observed from the earlier app's demo we need to change the number and speed of the bars. This can be done mainly by attributing each bar with a relative value, so that it becomes a pictorial representation of the array's elements that are being sorted.

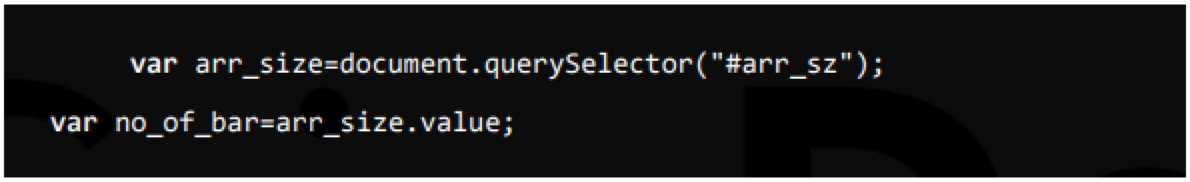
Requirements

- For this, we can use the input element in the navbar. In the HTML file input should be like this.



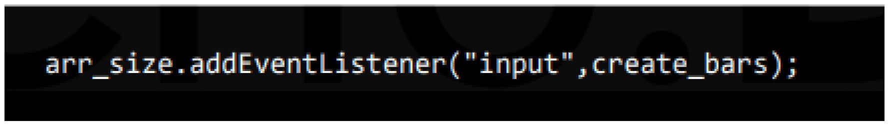
```
    <span>
            //no of bars
           <input id="arr_sz" type="range"  min=20 max=120 step=1
value=60>
    </span>
```

- And then in the javascript code part, using DOM we will select the input tag and take the value from that and pass it onto the create bar function. Instead of 100 we will use the number of bars as the inputted (to be taken) value from the input tag and in the delay function pass the delay time as the taken input from the speed input like the below code.



```
      var arr_size=document.querySelector("#arr_sz");
   var no_of_bar=arr_size.value;
```

- Also, add event listeners with no bar and pass create bar function this way.



```
  arr_size.addEventListener("input",create_bars);
```

Tips :

- Add an event listener to the number of bars because when we will use that the no of bars should change instantly.

Expected Outcome After implementing all the functionalities the end result of your app should be like this.

<u>Task 8 :Host your website live</u>

After completing all the milestones we have our application ready to be deployed and hosted live onto the web.

Start off by pushing your code to your GitHub account with a good README.md to publish your project.

Host your app live using Netlify and share its link among your peers and finally do add this project to your resume.

<u>PROJECT DESIGN</u>

The design of Sorting Visualizer will look like this.

The User-Interface

Even though the underlying back-end code went through a drastic refactor midway through the implementation, the overall design and layout of the user-interface components has remained the same. The interface has twelve components: a canvas area, ten control buttons, and a volume on/off toggle button.

The user can select any of these algorithms to see the visualization of how that algorithm works. There is no algorithm selected by default, so the user will need to select one before starting the animation.
Before selecting an algorithm, the user must select the type of input data to be sorted. The three gray-bordered buttons on the left of the bottom row allow the user to choose between sorting input data that is already in order or in reverse and random orders

The default is in sorted order. Once the input and the sorting algorithm have been selected, the user can click the green-bordered "Start" button in the next row of buttons to see the sort run from beginning to end. To see the algorithm execution slowly step-by-step, the user can click the yellow-orange-bordered "Step" button.
The "Stop" button simply halts the auto-animating process if in progress.

Input Design

Input design include the creation of the text fields and the space required input the data dynamically. A text box, text field or text entry box is a kind of widget used when building a graphical user interface (GUI). A text box's purpose is to allow the user to input text information to be used by the program. User-interface guidelines recommend a single-line text box when only one line of input is required, and a multi-line text box only if more than one line of input may be required. Non-editable text boxes can serve the purpose of simply displaying text.

A typical text box is a rectangle of any size, possibly with a border that separates the text box from the rest of the interface. Text boxes may contain zero, one, or two scrollbars. Text boxes usually display a text cursor (commonly a blinking vertical line), indicating the current region of text being edited. It is common for the mouse cursor to change its shape when it hovers over a text box.

User Interface Design

User interface design (UID) or user interface engineering is the design of websites, computers, appliances, machines, mobile communication devices, and software applications with the focus on the user's experience and interaction. The goal of user interface design is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals—what is often called user-centered design. Good user interface design facilitates finishing the task at hand without drawing unnecessary attention to itself. Graphic design may be utilized to support its usability. The design process must balance technical functionality and visual elements (e.g., mental model) to create a system that is not only operational but also usable and adaptable to changing user needs.

Interface design is involved in a wide range of projects from computer systems, to cars, to commercial planes; all of these projects involve much of the same basic human interactions yet also require some unique skills and knowledge. As a result, designers tend to specialize in certain types of projects and have skills centered around their expertise, whether that be software design, user research, web design, or industrial design

Processes

User interface design requires a good understanding of user needs. There are several phases and processes in the user interface design, some of which are more demanded upon than others, depending on the project. (Note: for the remainder of this section, the word system is used to denote any project whether it is a website, application, or device.)

Functionality requirements gathering

Assembling a list of the functionality required by the system to accomplish the goals of the project and the potential needs of the users.

User analysis

Analysis of the potential users of the system either through discussion with people who work with the users and/or the potential users themselves.

Typical questions involve:

What would the user want the system to do?

How would the system fit in with the user's normal workflow or daily activities?

How technically savvy is the user and what similar systems does the user already use?

What interface look & feel styles appeal to the user?

Information architecture

Development of the process and/or information flow of the system (i.e. for phone tree systems, this would be an option tree flowchart and for web sites this would be a site flow that shows the hierarchy of the pages).

Prototyping

Development of wireframes, either in the form of paper prototypes or simple interactive screens. These prototypes are stripped of all look & feel elements and most content in order to concentrate on the interface.

Usability inspection

letting an evaluator inspect a user interface. This is generally considered to be cheaper to implement than usability testing (see step below), and can be used early on in the development process since it can be used to evaluate prototypes or specifications for the system, which usually can't be tested on users. Some common usability inspection methods include cognitive walkthrough, which focuses the simplicity to accomplish tasks with the system for new users, heuristic evaluation, in which a set of heuristics are used to identify usability problems in the UI design, and pluralistic walkthrough, in which a selected group of people step through a task scenario and discuss usability issues.

Usability testing

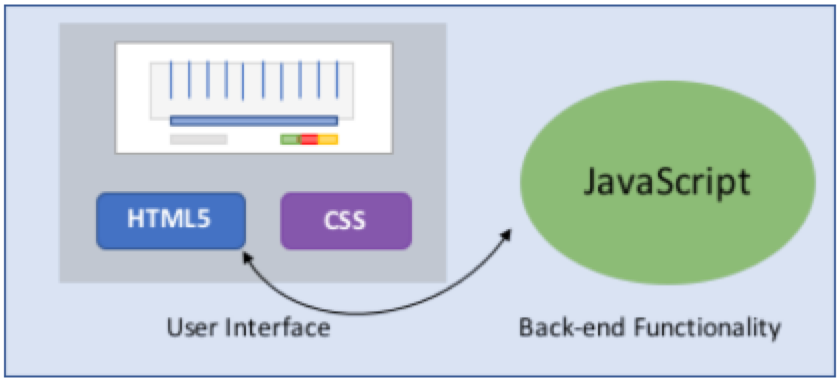
testing of the prototypes on an actual user—often using a technique called think aloud protocol where you ask the user to talk about their thoughts during the experience.

Graphic interface design

actual look and feel design of the final graphical user interface (GUI). It may be based on the findings developed during the usability testing if usability is unpredictable, or based on communication objectives and styles that would appeal to the user. In rare cases, the graphics may drive the prototyping, depending on the importance of visual form versus function. If the interface requires multiple skins, there may be multiple interface designs for one control panel, functional feature or widget. This phase is often a collaborative effort between a graphic designer and a user interface designer, or handled by one who is proficient in both disciplines.

System Architecture

The back-end code is comprised of HTML5, CSS, and JavaScript. All three types of code are contained in one .html file and can be run solely from this file. One of the advantages of HTML 5 is that it is not necessary to include different types of web languages in a single file. Therefore, each type could have been separated, making a total of three files (plus the miscellaneous sound and image files). This is good practice for readability and keeping related code together. However, I decided not to separate the code for two reasons: 1) to increase the portability of the project by only needing to worry about one project file instead of three, and 2) where in the project file, the change in coding languages is distinctly marked and therefore does not significantly reduce readability



As you can see, there are no major components besides the three coding languages. Most websites have tools or scripts that require a server on the back-end (like PHP), but it is not necessary in this case since JavaScript runs right in the user’s browser. HTML5 and CSS are used for the interface. The HTML5 communicates with the JavaScript code and vice versa to launch the appropriate algorithms and update the interface accordingly, as seen with a single, bidirectional arrow.

Throughout the project, the code for the HTML5 and CSS did not change much. As the JavaScript was modified from a functional programming focus to a more object-oriented one, the parts of the HTML5 that did change were the function calls for each button. All of the back-end interaction is abstracted to the various buttons for selecting algorithms and running the animation.

OUTPUT DESIGN

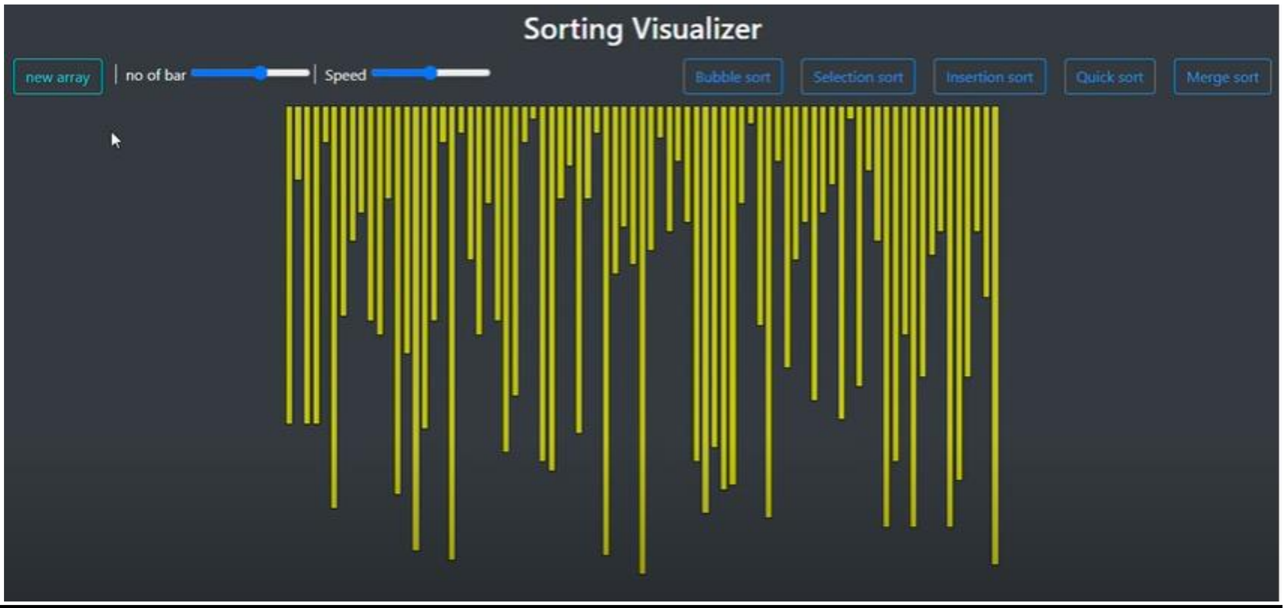
Designing computer output should proceed in an organized, well throughout manner; the right output element is designed so that people will find the system whether or executed. When we design an output we must identify the specific output that is needed to meet the system. The usefulness of the new system is evaluated on the basis of their output. Once the output requirements are determined, the system designer can decide what to include in the system and how to structure it so that the require output can be produced. For the proposed software, it is necessary that the output reports be compatible in format with the existing reports. The output must be concerned to the overall performance and the system’s working, as it should. It consists of developing specifications and procedures for data preparation, those steps necessary to put the inputs and the desired output, i.e. maximum user friendly. Proper messages and appropriate directions can control errors committed by users. The output design is the key to the success of any system. Output is the key between the user and the sensor.

The output must be concerned to the system's working, as it should. Output design consists of displaying specifications and procedures as data presentation.

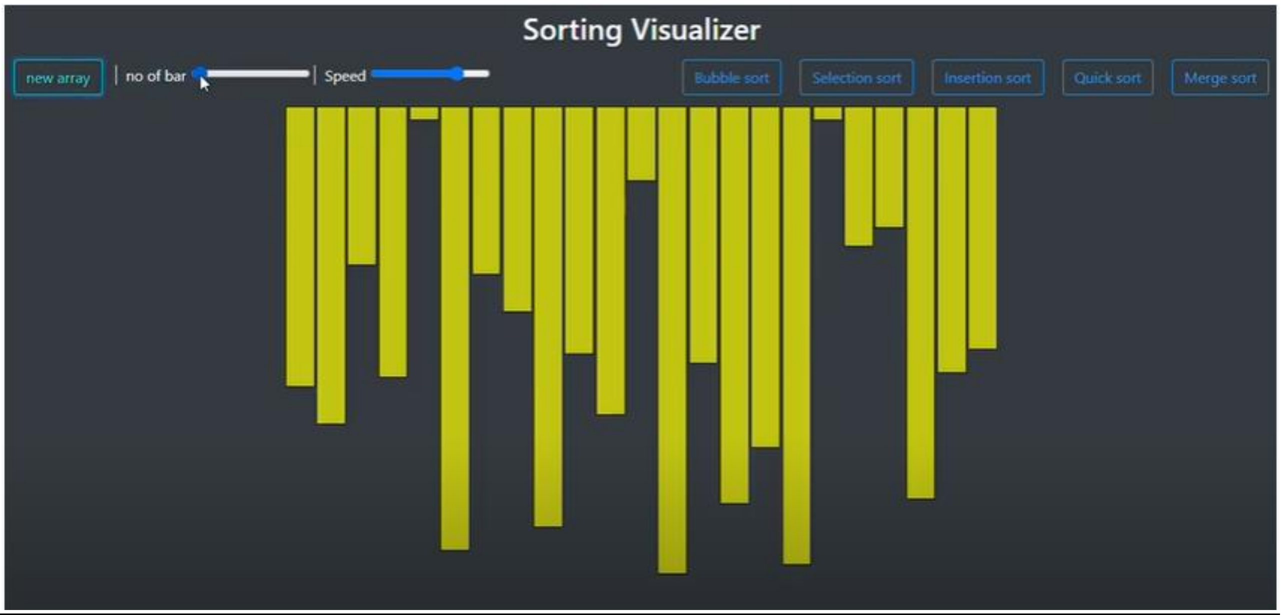
User never left with the confusion as to what is happening without appropriate error and acknowledges message being received.

Number of graph and speed can be changed.

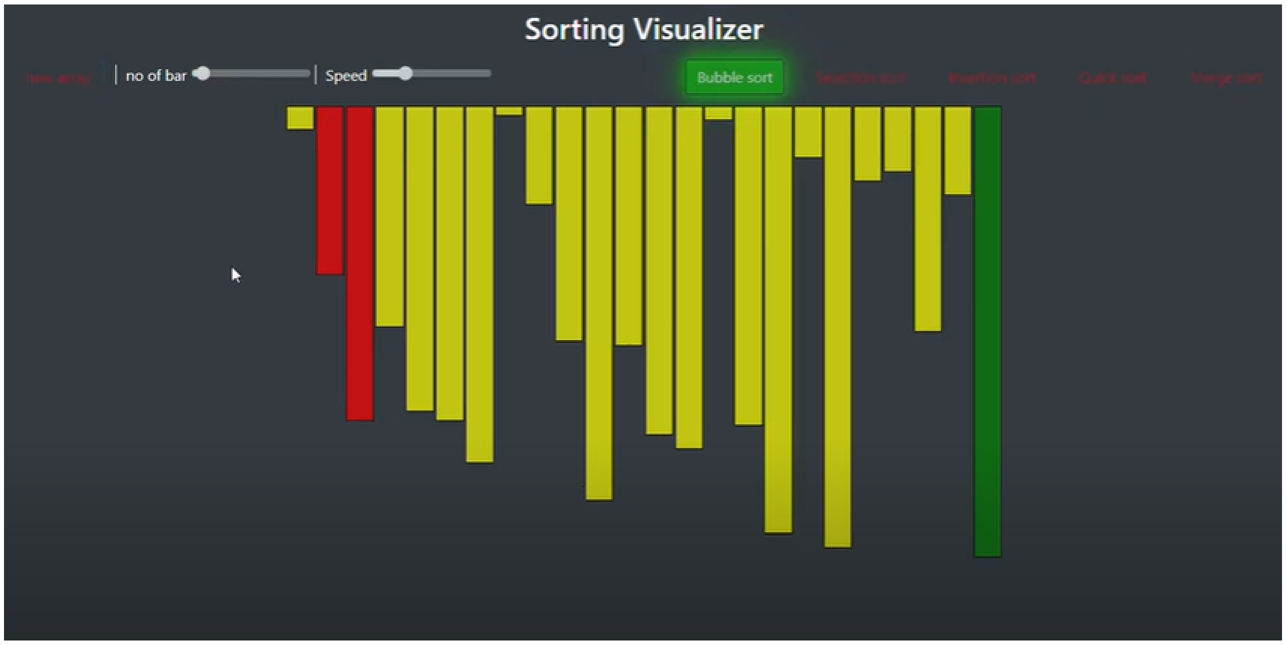
1.0) In below no. of bars is more.



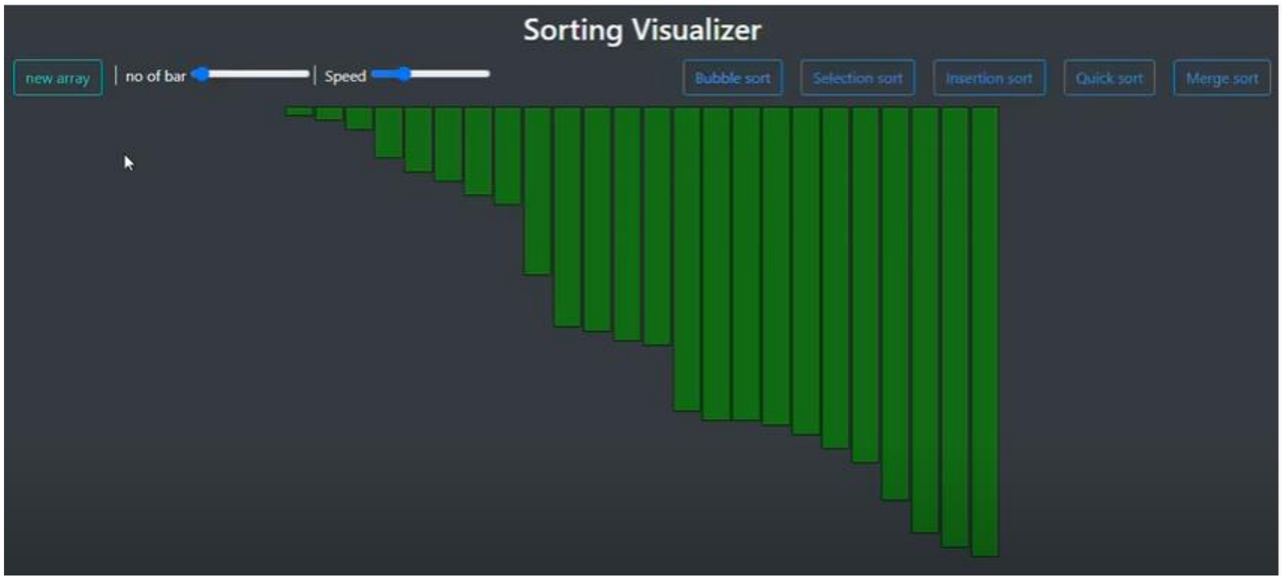
1.1) In below no. of bar is less.



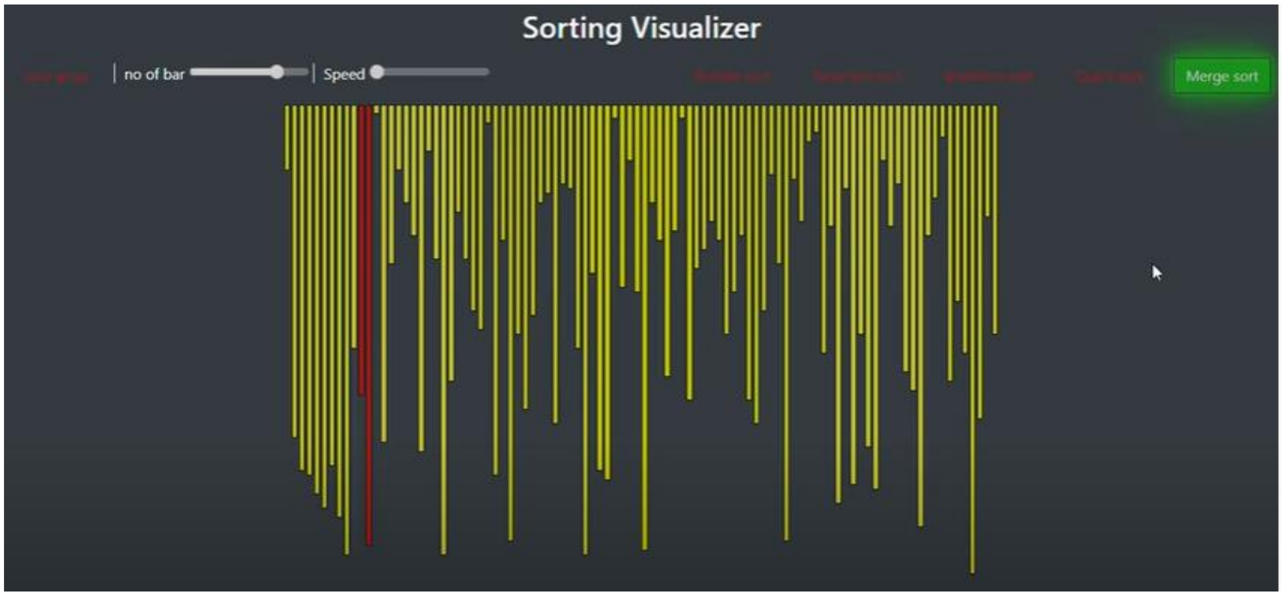
1.2) Bubble sort before using sorting



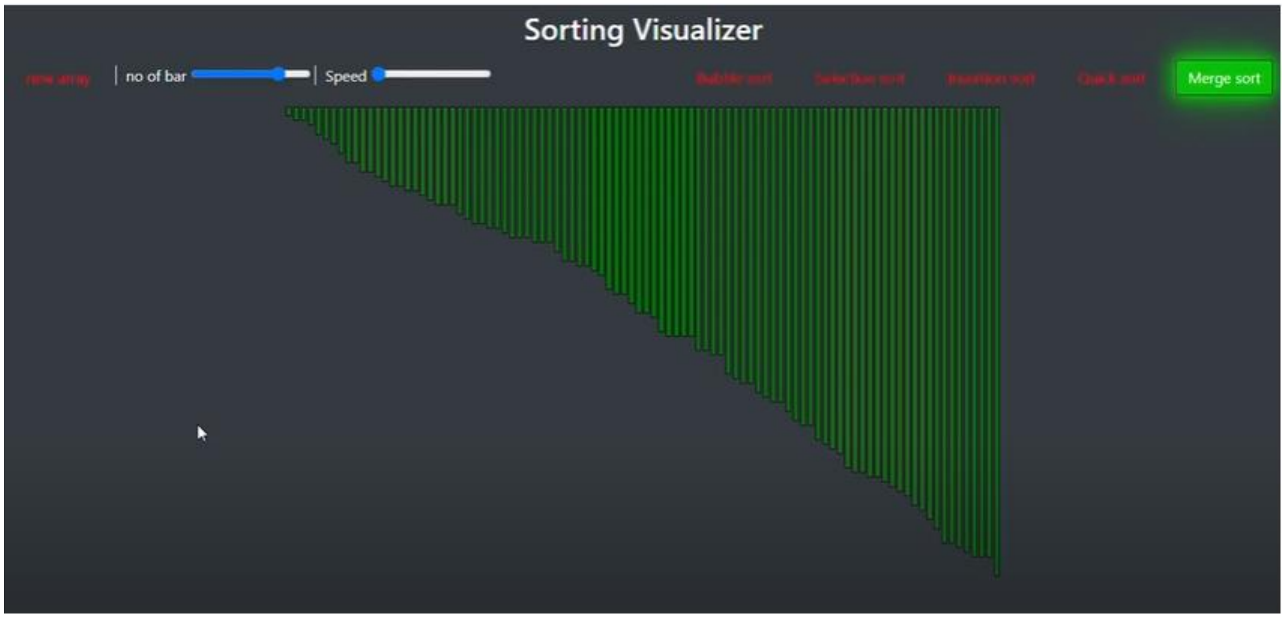
1.3) Bubble sort after using sorting



1.4) Merge sort before using sorting



1.5) Merge sort after using sorting



SYSTEM TESTING

System testing is the stage of implementation, which is aimed a ensuring that the system works accurately and efficiently before live operation commences. Testing is the process of executing the program with the intent of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied. The ultimate aim is quality assurance. Tests are carried out and the results are compared with the expected document. In the case of erroneous results, debugging is done. Using detailed testing strategies a test plan is carried out on each module. The various tests performed in “Network Backup System” are unit testing, integration testing and user acceptance testing.

Unit Testing

The software units in a system are modules and routines that are assembled and integrated to perform a specific function. Unit testing focuses first on modules, independently of one another, to locate errors. This enables, to detect errors in coding and logic that are contained within each module. This testing includes entering data and ascertaining if the value matches to the type and size supported. The various controls are tested to ensure that each performs its action as required.

Integration Testing

Data can be lost across any interface, one module can have an adverse effect on another, sub functions when combined, may not produce the desired major functions. Integration testing is a systematic testing to discover errors associated within the interface. The objective is to take unit tested modules and build a program structure. All the modules are combined and tested as a whole. Here the Server module and Client module options are integrated and tested. This testing provides the assurance that the application is well integrated functional unit with smooth transition of data.

User Acceptance Testing

User acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the system users at time of developing and making changes whenever required.

Results

The best way to go about using the tool is to first select the ordering of the data and then select which algorithm to visualize. When any one of the algorithm buttons are selected, it will sort the data as it appears on the interface. The ordering takes precedence, as selecting the ordering after the algorithm updates the interface momentarily, while the code has already run the initialization with the previous data set. After conducting the surveys, this sparked some confusion as the algorithm buttons are listed above the ordering buttons in the interface. One student commented on having difficulty trying to start sorting, thinking that it may be the cause of pressing the buttons in the wrong order, which in turn did not run the animation.

The responses of all the students can be found in Appendix D. Overall, there was not a significant advantage in using my animation tool to help learning about sorting algorithms. By looking at the student responses for question 3, which asked if their understanding of a particular algorithm changed after using the tool, 5 of the 13 students (38%) said yes in some way. The other 7 did not find it very helpful, even though most appreciated the idea of the tool. One student, however, gave a false positive to the tool being helpful (whom I did not include in the 5 that said it was helpful). A drawback to the animation is that it only shows the movements without the comparisons that lead to the movements of the data. This student saw how Selection Sort completes quickly compared to the other algorithms, as there are O(n) swaps that take place, which is beneficial in avoiding unnecessary data moves the computer needs to make. In contrast, the process of comparing the data results in a O(n2) runtime (the slowest overall). Another student noted this discrepancy in response to question 5 that asked for comments and feedback, noting that Merge Sort is the best of the four sorts.

Merge Sort has an average runtime of O(n log2 n), which is the best average runtime out there. One way to resolve this would be to integrate visualizing the comparisons as well as the movements. This way, the bars would change color when an algorithm is comparing data, taking up more time in the animation. Selection Sort and Bubble sort use the most comparisons, so their time to complete would slow down and be more appropriate compared to the other algorithms.

Conclusions and Future Work

Through much time and effort, I have successfully created a working web based animation tool for visualizing the following sorting algorithms: Selection Sort, Bubble Sort, Insertion Sort, and Merge/Insertion Sort. Even with its memory overhead, it received overall positive feedback from the students who explored it. I am not surprised that there was not a significant difference in learning the material, which reflects what I found in my previous research. There remains, however, a strong mindset to research and create animations like these to improve learning in the classroom, which I agree with completely. Learning how to code a web platform was challenging, and I thank the tutorials on W3Schools.com for getting me there. I had a previous internship where I updated the JavaScript on a webpage, but it was much more concise and did not involve objects and HTML5 for visualizations. The good news is that JavaScript is still one of the most popular web languages, so I am not too worried about another big refactor soon for a language update. For my laundry list of future works, the elephant in the room is to resolve the memory issues. Next would be to modify Merge/Insertion Sort to reflect a true Merge Sort. After which, I would get Quick Sort up and running, as the code is already in a state where it would not be too difficult to integrate. Then, I would add the suggestions listed in the bulleted feedback of to further promote usability and understandability. Finally, I would make the web tool public, realizing my most desired feature of making it public. This would also present some new challenges. Even though the animation tool works locally, I have unintentionally avoided the issue of concurrency, where a server can handle multiple requests to the web site by different users. I would need to give more thought on how to optimize the code so that it can work with multiple people using it.

REFERENCES

[1] D. Radošević, T. Orehovački, and A. Lovrenčić, “Verificator: Educational Tool for Learning Programming”, Informatics in Education, vol. 8, no. 2, 2009, pp. 261-280.

[2] J. Bennedsen, and M. E. Caspersen, “Failure Rates in Introductory Programming”, ACM SIGCSE Bulletin, vol. 39, no. 2, 2007, pp. 32-36.

[3] S. Al-Imamy, J. Alizadeh, and M.A. Nour, "On the Development of a Programming Teaching Tool: The Effect of Teaching by Templates on the Learning Process", In Proceedings of JITE, 2006, pp.271-283.

[4] T. Naps, J. Eagan, and L. Norton, “JHAVÉ: An Environment to Actively Engage Students in Web-based Algorithm Visualizations”, In Proceedings of the 31st ACM SIGCSE Technical Symposium on Computer Science Education, Austin:ACM, 2000. pp. 109-113.

[5] A. Gomes, and A. J. Mendes, “Learning to program – difficulties and solutions”. In: Proceedings of the International Conference on Engineering Education. Coimbra, Portugal, 2007 http://icee2007.dei.uc.pt/proceedings/papers/411.pdf

[6] S. Hansen, N. H. Narayanan, and M. Hegarty, “Designing Educationally Effective Algorithm Visualizations”, Journal of Visual Languages and Computing, vol. 13, no. 3, 2002, pp. 291- 317.

[7] G. P. Waldheim, “Understanding How Students Understand”, Engineering Education, vol. 77, no. 5, 1987, pp. 306-308.

[8] T. L. Naps, G. Rößling, V. Almstrum, W. Dann, R. Fleischer, C. Hundhausen, A. Korhonen, L. Malmi, M. McNally, S. Rodger, and J. Á. Velázquez-Iturbide, “Exploring the Role of Visualization and Engagement in Computer Science Education”, In Working group reports

from ITiCSE on Innovation and technology in computer science education, Aarhus: ACM, 2002, pp. 131-152.

[9] M. Conway, S. Audia, T. Burnette, D. Cosgrove, and K. Christiansen, “Alice: lessons learned from building a 3D system for novices”, In Proceedings of the SIGCHI conference on Human factors in computing systems, The Hague: ACM, 2000, pp. 486 – 493.

[10] A. W. Lawrence, “Empirical Studies of the Value of Algorithm Animation in Algorithm Understanding. PhD thesis, Department of computer Science, Georgia Institute of Technology, 1993, http://www.dtic.mil/cgibin/GetTRDoc?AD=ADA275135&Location=U2 &doc=GetTRDo c.pdf

[11] Guido von Robot, http://gvr.sourceforge.net/

[12] J. Stasko, “Samba algorithm Animation System”, http://www.cc.gatech.edu/gvu/softviz/algoanim/samba.html

[13] A. Zeller, “Animating data structures in DDD”, In Proceedings of the SIGCSE/SIGCUE Program Visualization Workshop, 2000, Porvoo: ACM, pp. 69-78.

[14] A. I. Concepcion, N. Leach, and A. Knight, “Algorithma 99: an experiment in reusability & component based software engineering”, ACM SIGCSE Bulletin, vol. 32, no. 1, 2000, pp.. 162-166.

[15] W. C. Pierson, and S. H. Rodger, “Web-based animation od data structures using JAWAA”, ACM SIGCSE Bulletin, vol. 30, no. 1, 1998, pp. 267-271.

[16] H. Liberman, and C. Fry, “Zstep 95: A reversible, animated source code stepper”, In Software Visualization--Programming as a Multimedia Experience, 1998, pp. 277-292.

[17] C. D. Hundhausen, and S. A. Douglas, “Low-Fidelity Algorithm Visualization”, Journal of Visual Languages and Compunting 2002, vol. 13, no. 5, pp. 449-470.

[18] A. Moreno, N. Myller, E. Sutinen, and M. Ben-Ari, “Visualizing Programs with Jeliot 3, In Proceedings of the working conference on Advanced visual interfaces, Gallipoli: ACM, 2004. pp. 373-376.

[19] D. J. Barnes, and M. Kolling, “Objects First with Java: A Practical Introduction Using BlueJ”, Prentice Hall; 2 edition, 2004.

[20] D. Radosevic, and T. Orehovacki, “An Analysis of Novice Compilation Behavior using Verificator”, In Proceedings of the 33rd International Conference on Information Technology Interfaces (ITI), Cavtat: IEEE, 2011. pp. 325–330.

[21] G. Rößling, and B. Freisleben, "ANIMAL: A System for Supporting Multiple Roles in Algorithm Animation", Journal of Visual Languages & Computing, vol. 13, no. 3, 2002, pp. 341-354.

[22] J. C. Bradley, A. C. Millspaugh, “Advanced Programming Using Visual Basic .NET”, Mcgraw-Hill, 2
nd edition, 2003.

[23] L. J. Cronbach, “Coefficient Alpha and the Internal Structure of Tests”, Psychometrika, vol. 16, no. 3, 1951, pp. 297-334.

[24] F. D. Davis, “Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology”, MIS Quarterly, vol. 13, no. 3, 1989, pp. 319-340.

[25] M. Gong, Y. Xu, Y. Yu, “An Enhanced Technology Acceptance

Model for Web-Based Learning", Journal of Information Systems Education vol. 15, no. 4, 2004, pp. 365-374.

[26] J. C. Nunnally, "Psychometric Theory", Second Edition, McGraw Hill, New York, 1978.

[27] W. Dann, S. Cooper, and R. Pausch, "Using Visualization To Teach Novices Recursion", In Proceedings of the 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, Canterbury, England, 2001, pp. 109-112.

[28] M. Guzdial, and E. Soloway, "Log on education: teaching the Nintendo generation to program", Communications of the ACM, vol. 45, no. 4, 2002, pp. 17-21.

[29] R. B. Findler, C. Flanagan, M. Flatt, S. Krishnamurthi, and M. Felleisen, "DrScheme: A pedagogic programming environment for scheme", Lecture Notes in Computer Science, vol. 1292, 1997, pp. 369-388.

[30] B. Erwin, M. Cyr, and C. Rogers, "LEGO Engineer and RoboLab: Teaching Engineering with LabVIEW from Kindergarten to Graduate School", International Journal of Engineering Education, vol. 16, no. 3, 2000, pp. 181-192.