SORTING VISULAZIER

Done By
Mr. Chinmay A. Shete
Mr. Gandharv N. Kulkarni

OBJECTIVE

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.

PROJECT CONTEXT

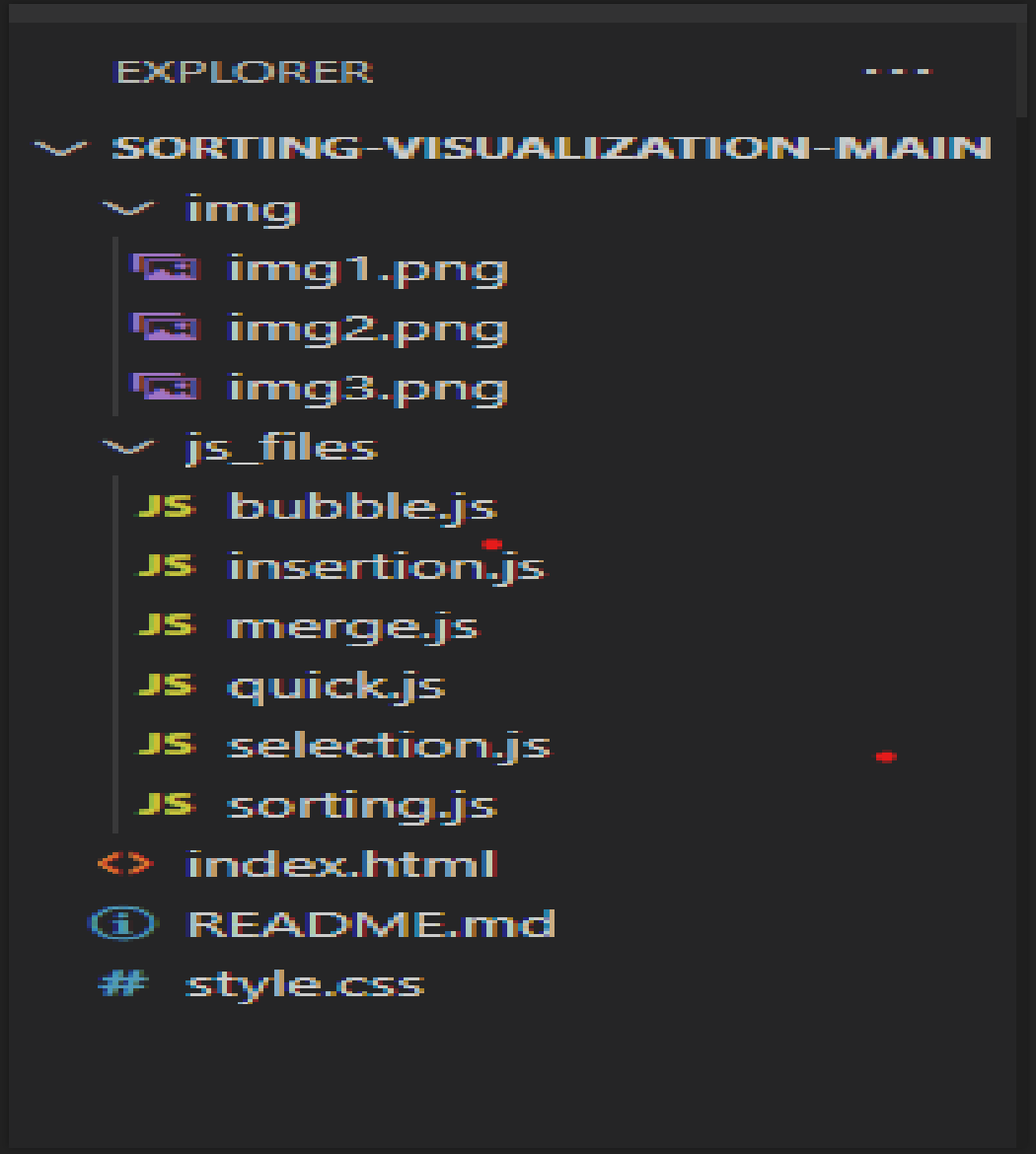
We have learnt sorting algorithms like bubble sort, selection sort, insertion sort, quick sort. But often we fail to understand the core idea of a particular algorithm maybe because we are unable to visualize how they work. So the most important thing to understand about these algorithms is visualization.

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.

DIRECTORY STRUCTURE

This is a typical JavaScript Project , so you need a code editor like VS-code (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.



SORTING ALGORITHM COVERED IN PROJECTS

We have covered all the basic Sorting Algorithm in our project i.e,.

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Quick Sort
5. Merge Sort

So, Basically What is Sorting Algorithm?

A sorting algorithm is a method for reorganizing a large number of items into a specific order, such as alphabetical, highest-to-lowest value or shortest-to-longest distance.

Sorting algorithms take lists of items as input data, perform specific operations on those lists and deliver ordered arrays as output.

Bubble Sort

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order.

Time Complexity:- O(n^2).

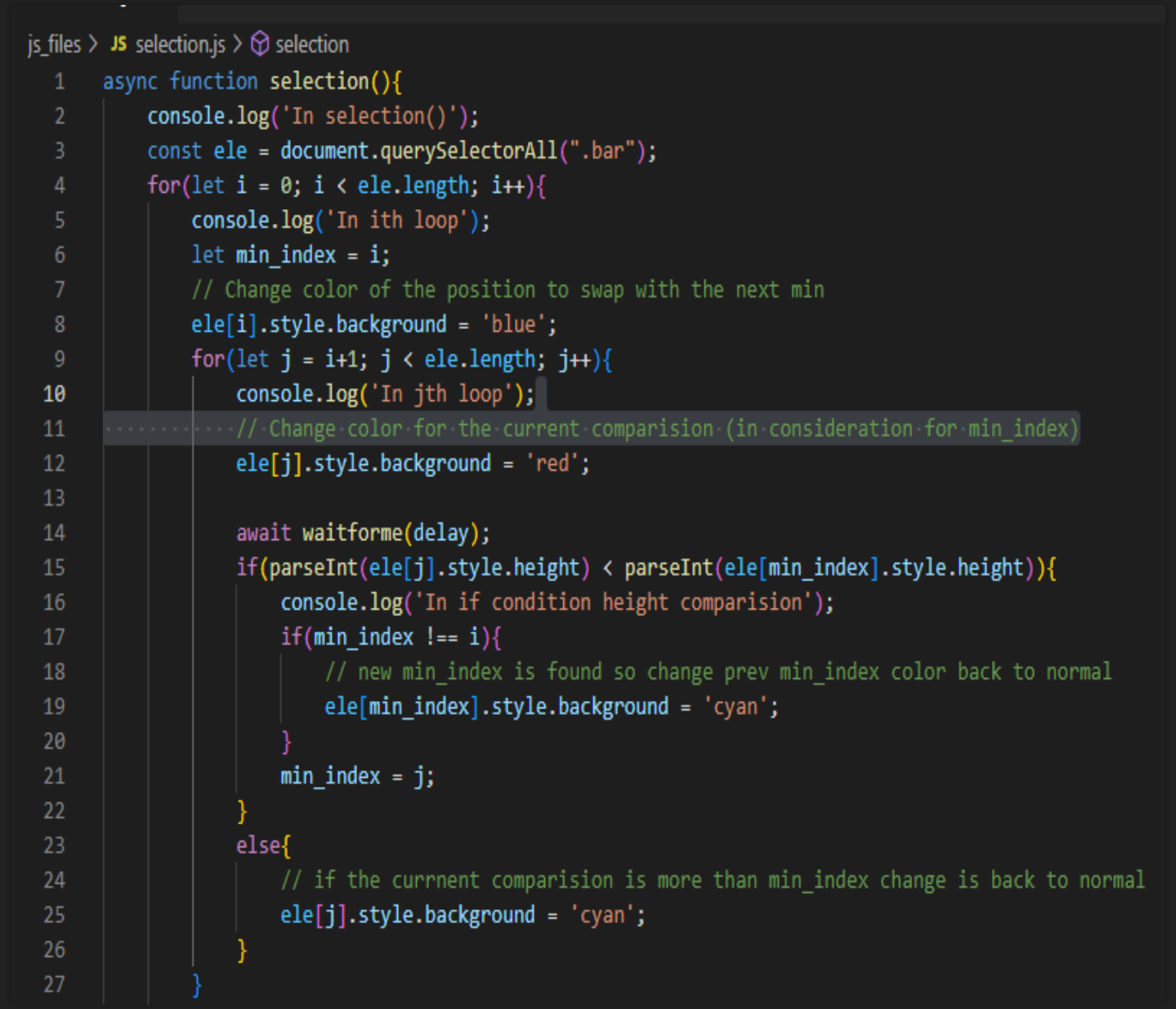
js_files > JS bubble.js > bubble

```
async function bubble() {
    console.log('In bubbe()');
    const ele = document.querySelectorAll(".bar");
    for(let i = 0; i < ele.length-1; i++){
        console.log('In ith loop');
        for(let j = 0; j < ele.length-i-1; j++){
            console.log('In jth loop');
            ele[j].style.background = 'blue';
            ele[j+1].style.background = 'blue';
            if(parseInt(ele[j].style.height) > parseInt(ele[j+1].style.height)){
                console.log('In if condition');
                await waitforme(delay);
                swap(ele[j], ele[j+1]);
            }
            ele[j].style.background = 'cyan';
            ele[j+1].style.background = 'cyan';
        }
        ele[ele.length-1-i].style.background = 'green';
    }
    ele[0].style.background = 'green';
}
```

Selection Sort

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from unsorted part and putting it at the beginning.

Time complexity:- O(n^2).



Insertion Sort

Insertion sort is a sorting algorithm that places an unsorted element at its suitable place in each iteration.

Time complexity:- O(n^2).

js_files > JS insertion.js > insertion

```
async function insertion(){
    console.log('In insertion()');
    const ele = document.querySelectorAll(".bar");
    // color
    ele[0].style.background = 'green';
    for(let i = 1; i < ele.length; i++){
        console.log('In ith loop');
        let j = i - 1;
        let key = ele[i].style.height;
        // color
        ele[i].style.background = 'blue';

        await waitforme(delay);

        while(j >= 0 && (parseInt(ele[j].style.height) > parseInt(key))){
            console.log('In while loop');
            // color
            ele[j].style.background = 'blue';
            ele[j + 1].style.height = ele[j].style.height;
            j--;

            await waitforme(delay);

            // color
            for(let k = i; k >= 0; k--){
                ele[k].style.background = 'green';
            }
        }
        ele[j + 1].style.height = key;
        // color
        ele[i].style.background = 'green';
    }
```

Quick Sort

Quick Sort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot.

Time Complexity:- O(nlogn).

js_files > JS quick.js > ...

```
1
2   async function partitionLomuto(ele, l, r){
3       console.log('In partitionLomuto()');
4       let i = l - 1;
5       // color pivot element
6       ele[r].style.background = 'red';
7       for(let j = l; j <= r - 1; j++){
8           console.log('In partitionLomuto for j');
9           // color current element
10          ele[j].style.background = 'yellow';
11          // pauseChamp
12          await waitforme(delay);
13
14          if(parseInt(ele[j].style.height) < parseInt(ele[r].style.height)){
15              console.log('In partitionLomuto for j if');
16              i++;
17              swap(ele[i], ele[j]);
18              // color
19              ele[i].style.background = 'orange';
20              if(i != j) ele[j].style.background = 'orange';
21              // pauseChamp
22              await waitforme(delay);
23          }
24          else{
25              // color if not less than pivot
26              ele[j].style.background = 'pink';
27          }
28      }
29      i++;
```

Merge Sort

Merge Sort is a Divide and Conquer algorithm. It divides the input array into two halves, calls itself for the two halves, and then it merges the two sorted halves.

Time Complexity:- O(nlogn).

js_files > JS merge.js > ...

```
1   //let delay = 30;
2   async function merge(ele, low, mid, high){
3       console.log('In merge()');
4       console.log(`low=${low}, mid=${mid}, high=${high}`);
5       const n1 = mid - low + 1;
6       const n2 = high - mid;
7       console.log(`n1=${n1}, n2=${n2}`);
8       let left = new Array(n1);
9       let right = new Array(n2);
10
11      for(let i = 0; i < n1; i++){
12          await waitforme(delay);
13          console.log('In merge left loop');
14          console.log(ele[low + i].style.height + ' at ' + (low+i));
15          // color
16          ele[low + i].style.background = 'orange';
17          left[i] = ele[low + i].style.height;
18      }
19      for(let i = 0; i < n2; i++){
20          await waitforme(delay);
21          console.log('In merge right loop');
22          console.log(ele[mid + 1 + i].style.height + ' at ' + (mid+1+i));
23          // color
24          ele[mid + 1 + i].style.background = 'yellow';
25          right[i] = ele[mid + 1 + i].style.height;
26      }
```

SOFTWARE TESTING

Our software application is incomplete without testing it.

Software testing is the process of verifying and validating that software application is bug free and meets the technical and user requirements as guided by its design and development.

TESTING METHODOLOGY USED FOR OUR PROJECT.

Our project is gone through the following software testing types:-

1. Unit Testing
2. Integration Testing
3. Usability Testing
4. Compatibility Testing

Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. This testing methodology is done during the development process by the software developers and sometimes QA staff.

The main objective of unit testing is to isolate written code to test and determine if it works as intended.

Unit testing is an important step in the development process, because if done correctly, it can help detect early flaws in code which may be more difficult to find in later testing stages.

Integration Testing

Integration testing -- also known as integration and testing (I&T) -- is a type of software testing in which the different units, modules or components of a software application are tested as a combined entity. However, these modules may be coded by different programmers.
The aim of integration testing is to test the interfaces between the modules and expose any defects that may arise when these components are integrated and need to interact with each other.

Usability Testing

Usability Testing also known as User Experience(UX) Testing, is a testing method for measuring how easy and user-friendly a software application is. A small set of target end-users, use software application to expose usability defects. Usability testing mainly focuses on user's ease of using application, flexibility of application to handle controls and ability of application to meet its objectives.

Compatibility Testing

Compatibility testing is **a form of non-functional software testing** -- meaning it tests aspects such as usability, reliability and performance -- that is used to ensure trustworthy applications and customer satisfaction. Compatibility tests are crucial to the successful performance of applications.

It is not done for each application; we will do it only for that application where we don't have control over the platform used by users.

Final Output of our Project

Now here comes the final output of our project i.e., Sorting Visualizer.

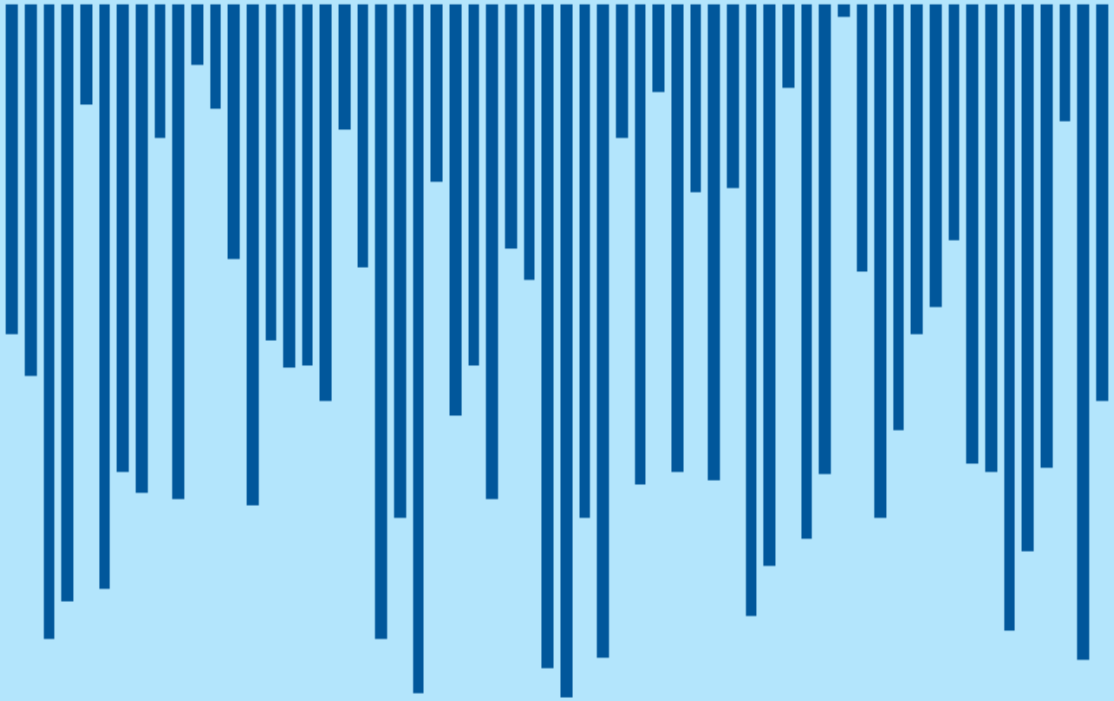
Sorting Visualizer

New Array

Size

Speed

Bubble Sort | Selection Sort | Insertion Sort | Quick Sort | Merge Sort

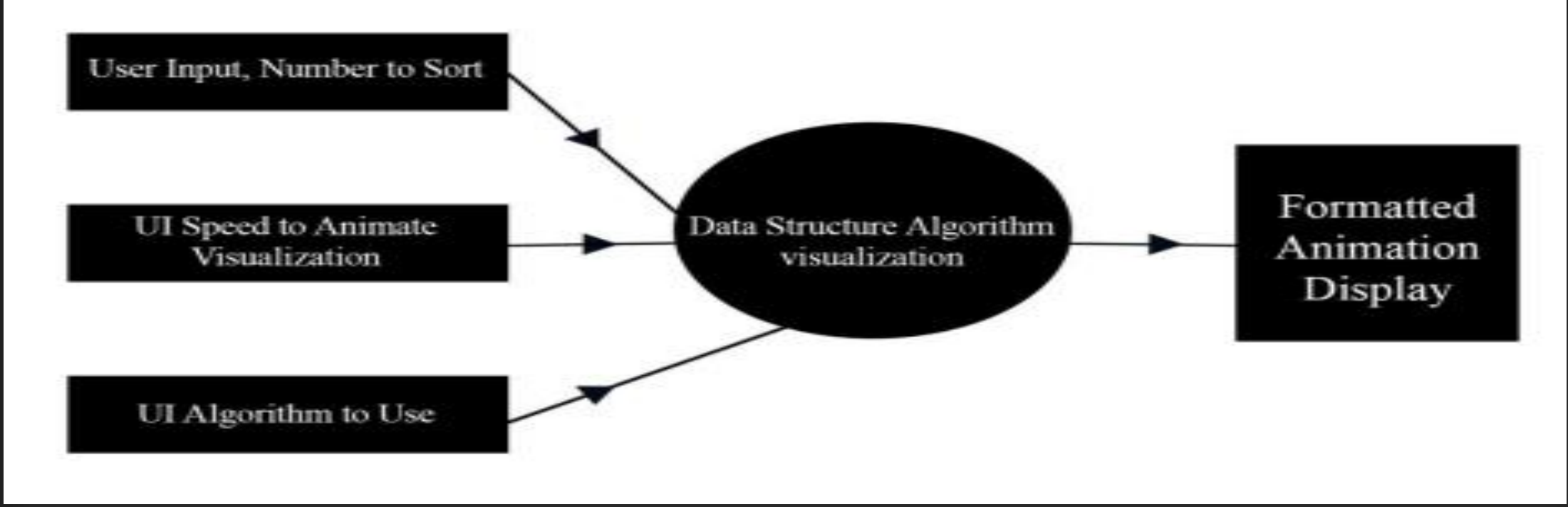


Enter Custom Array

Generate Custom Array

e.g 10,20,40,50,60

Data Flow Diagram Of Our Project



Level 1 DFD for Sorting Visualizer.

Future Development In Our Project

- Not all the sorting algorithm are covered, so one can implement the other sorting algorithms in JavaScript file for visualization.
- One can make the new .html file and add the explanation of algorithm all algorithm with code.
- One can change the .css file update the design of project and make it more attractive.

Thank You!