A Minor Project Report

on

THE SORTING VISUALIZER

submitted in partial fulfilment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

by

P. SHIVA SAI PRASAD	(19211A05M5)
N. RAVI TEJA	(19211A05J1)
S. SAI KIRAN	(19211A05P8)

Under the guidance of

Mrs. CH SREEDEVI,
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

B.V.RAJU INSTITUTE OF TECHNOLOGY

(UGC Autonomous, Accredited by NBA & NAAC)

Vishnupur, Narspur, Medak(Dist.), Telangana State, India - 502313

2020 – 2021

B. V. Raju Institute of Technology

(UGC Autonomous, Accredited by NBA & NAAC)

Vishnupur, Narsapur, Medak (Dist.),

Telangana State, India - 502313

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Minor Project entitled **"SORTING VISUALIZER"**, being submitted by

P. SHIVA SAI PRASAD	**(19211A05M5)**
N. RAVI TEJA	**(19211A05J1)**
S. SAI KIRAN	**(19211A05P8)**

In partial fulfilment of the requirements for the award of degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING to B.V.RAJU INSTITUTE OF TECHNOLOGY is a record of bonafide work carried out during a period from May 2020 to July 2021 by them under the guidance of **Mrs. Ch. Sreedevi**, Assistant Professor, CSE Department.

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

Mrs. Ch. Sreedevi
Assistant Professor

The Project Viva-Voce Examination of this team has been held on _____.

Dr. Ch. Madhu Babu
Professor & HoD-CSE

B. V. Raju Institute of Technology
(UGC Autonomous, Accredited by NBA & NAAC)
Vishnupur, Narsapur, Medak (Dist.),
Telangana State, India - 502313

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project entitled **"SORTING VISUALIZER"** in partial fulfillment of the requirements for the award of Degree of Bachelor of Technology and submitted in the Department of Computer Science and Engineering, B. V. Raju Institute of Technology, Narsapur is an authentic record of my own work carried out during a period from May 2020 to July 2021 under the guidance of **Mrs. Ch. Sreedevi**, Assistant Professor. The work presented in this project report has not been submitted by us for the award of any other degree of this or any other Institute/University.

P. SHIVA SAI PRASAD (19211A05M5)

N. RAVI TEJA (19211A05J1)

S. SAI KIRAN (19211A05P8)

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely fortunate to have got this all along the completion. Whatever we have done is due to such guidance and assistance. We would not forget to thank them.

We thank **Mrs. Ch. Sreedevi** for guiding us and providing all the support in completing this project. We are thankful to **Mr. Abdus Subhahan**, our section project coordinator for supporting us in doing this project. We are thankful to Mr. V. Pradeep Kumar, project coordinator for helping us in completing the project in time. We thank the person who has our utmost gratitude is **Dr. Ch. Madhu Babu**, Head of CSE Department.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all the staff members of CSE Department.

P. Shiva Sai Prasad (19211A05M5)

N. Ravi Teja (19211A05J1)

S. Sai Kiran (19211A05P8)

SORTING VISUALIZER

ABSTRACT

- This is the project which is useful for computer science students. For every computer science student understanding Data Structures is very important. And we all know that Sorting techniques plays a vital role in data structures.
- There are many different sorting techniques which have there significance depending on the situation. All those sorting techniques have different approaches. Many students don't understand the core idea of the particular algorithm, because they are unable to visualize how they work. The most important thing of these sorting algorithms is visualization.
- That's why we are making this project to let everyone understand how these sorting algorithms work visually. Through this project we also get deep understanding about these sorting algorithms.

TEAM MEMBERS:

1. P. Shiva Sai Prasad – 19211A05M5
2. N. Ravi Teja – 19211A05J1
3. S. Sai Kiran – 19211A05P8

GUIDE:

Mrs. Ch. Sreedevi, Assistant Professor

CONTENTS

Chapter 1

INTRODUCTION

1.1 Motivation

Analysis and design of algorithms is a great challenge for both computer and information science students. Fear of programming, lack of interest, and the abstract nature of the programming concepts are the main cause of the high drop out and failure rates in introductory programming courses. With an aim to motivate and help students, many researchers have proposed different kinds of tools. Although it has been reported that some of these tools have a positive impact on gaining programming skills, the problem still remain mostly unsolved.
To develop one of such tools which helps the students understand the sorting algorithms, has become our main motive. In this project the students can learn the sorting algorithms easily. Because they are depicted visually how they work.

1.2 Problem Definition

As we all know that we learn more easily through visualization than by reading. And I saw many of my friends and myself faced a lot of difficulty in understanding the sorting algorithms by listening, reading, and by hand tracing. So to make it easy to understand all the sorting techniques I thought of making this web application of sorting visualizer. Sorting Visualizer will produce the visualization of the sorting algorithms how they work visually by using the vertical bars representing the value of the elements or anything representing the data for sorting. In this we can visualize the sorting algorithms and we can understand them easily. We can also get better understanding about their working. We have produced the visualization of the five sorting algorithms.

1.3 Objective of Project

Our project Sorting Visualizer, a web application or tool. The main objective of our project is to make the students of computer and information science easily understand the sorting techniques. Because it is not easy for all the students to learn and understand these sorting techniques. Few students may understand easily and many will not. These many students are not understanding because of not understanding the working process of the sorting algorithms. These working process can be understood easily through visualization. As we know that "The brain processes visual information 60,000 times faster than text." Our project will visualize the working process of different sorting techniques.

1.4 Limitations of Project

There are few limitations in our project. They are:

- Our project is limited to only five sorting techniques for now.
- We cannot skip the sorting operation, we have to wait till the completion of the sorting operation.
- Animations used to show the sorting are not very smooth.

Chapter 2

LITERATURE SURVEY

2.1 Introduction

"It has been said that 80% of what people learn is visual " said by Allen Klein

- The human brains can easily process visuals instead of long codes to understand the algorithms.
- In this project, for visualization firstly we generate a random array and represent them using the bars.
- Different colors are used to indicate which elements are being compared, sorted and unsorted. We can also increase and decrease the size of the array and speed of the sorting operation.
- We use buttons representing the type of sorting technique (Bubble, Selection, Merge …). Then according to the button pressed, its corresponding sorting algorithm will be executed and we can see the sorting operation visually.

2.2 Existing System

- There are many existing systems related to this work.
- They did include the different types of the sorting techniques.
- They use different colors for the elements that are being sorted, compared and unsorted.
- We can increase and decrease the size of the no of elements.
- We can also increase and decrease the speed of the sorting operation going on.

2.3 Disadvantages of Existing System

There are existing systems related to this project.

The disadvantages of those project include the following points:

- Most of them are not included all the sorting techniques.
- They are not provided with the complexities of the sorting techniques.
- Most of them are also not compatible with the mobiles and tabs.

2.4 Proposed System

The proposed system will include the following:

- We will provide the sorting visualization of all the basic sorting techniques
- This project is compatible with both desktop and the mobiles.
- We may also include the complexities of the sorting techniques.
- User interface will be user friendly.

Chapter 3

ANALYSIS

3.1 Introduction

In this chapter, we will discuss about the analysis of the development process of our project including software requirements to do the project. Hardware requirements which make the webpage run smoothly. The requirements are included to provide complete description and over view of the system requirements of the user and the admin so that it will be very useful. Software requirement specification is divided into three components as follows, user requirements, software requirements and hardware requirements for convenience.

3.2 Software Requirement Specification

3.2.1 User Requirements

The User Requirements are as follows :

- Operating system :

  Laptop: Windows 7, windows 8, windows, Linux and Mac compatible.

- Browser :

  Chrome, Firefox, Microsoft edge or safari

- Internet connection: Required

3.2.2 Software Requirements

Languages : -

- HTML5
- CSS
- JavaScript
- Bootsrap

3.2.3 Hardware Requirements

- OS:

  Microsoft Windows 7 or 8 or 10 (32 bit or 64 bit)

- Processor:

  Intel core i3 or above recommended

- RAM:

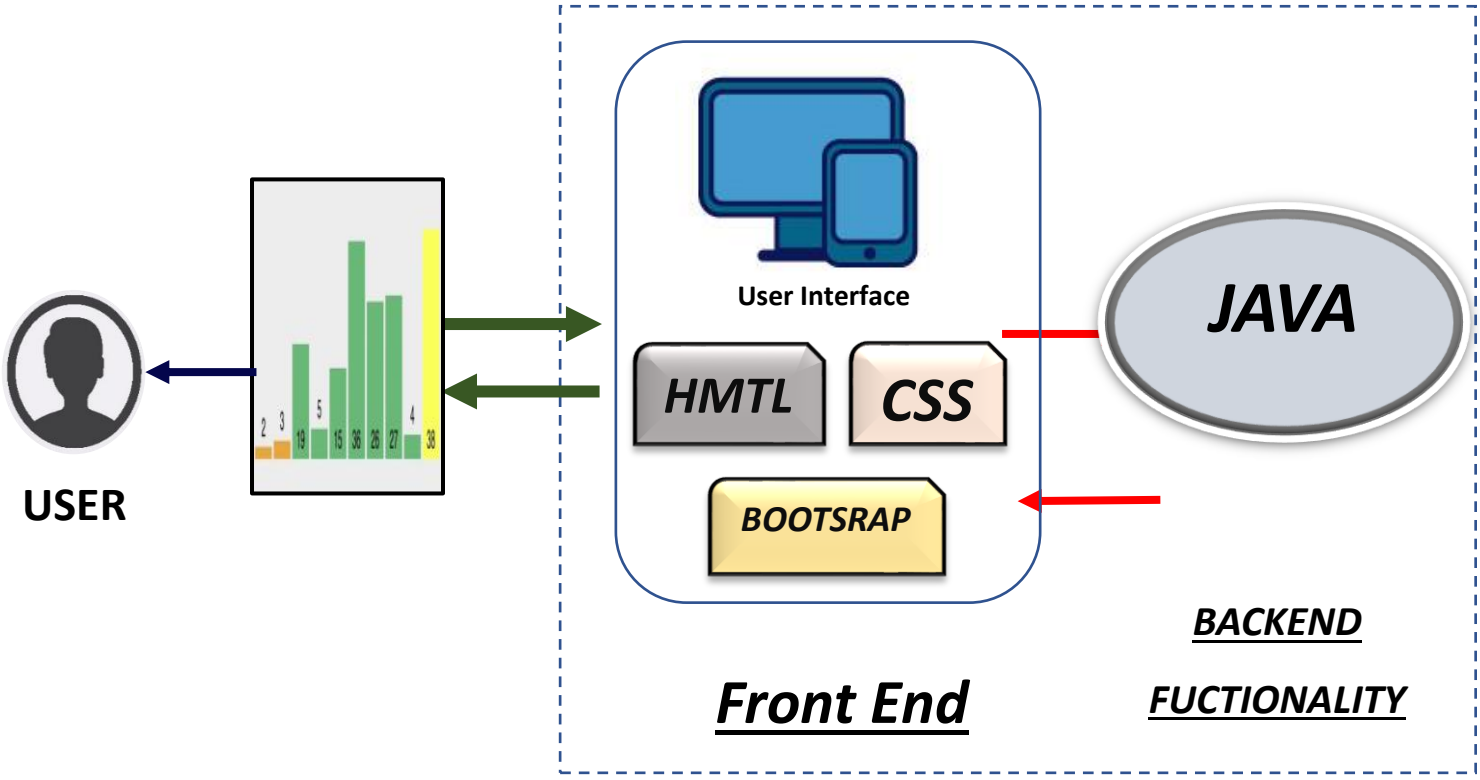
  Minimum 4GB RAM and 8GB recommended

- Memory:

  Internal memory is enough.

3.3 Content Diagram of Project

Architecture Diagram of Sorting Visualizer



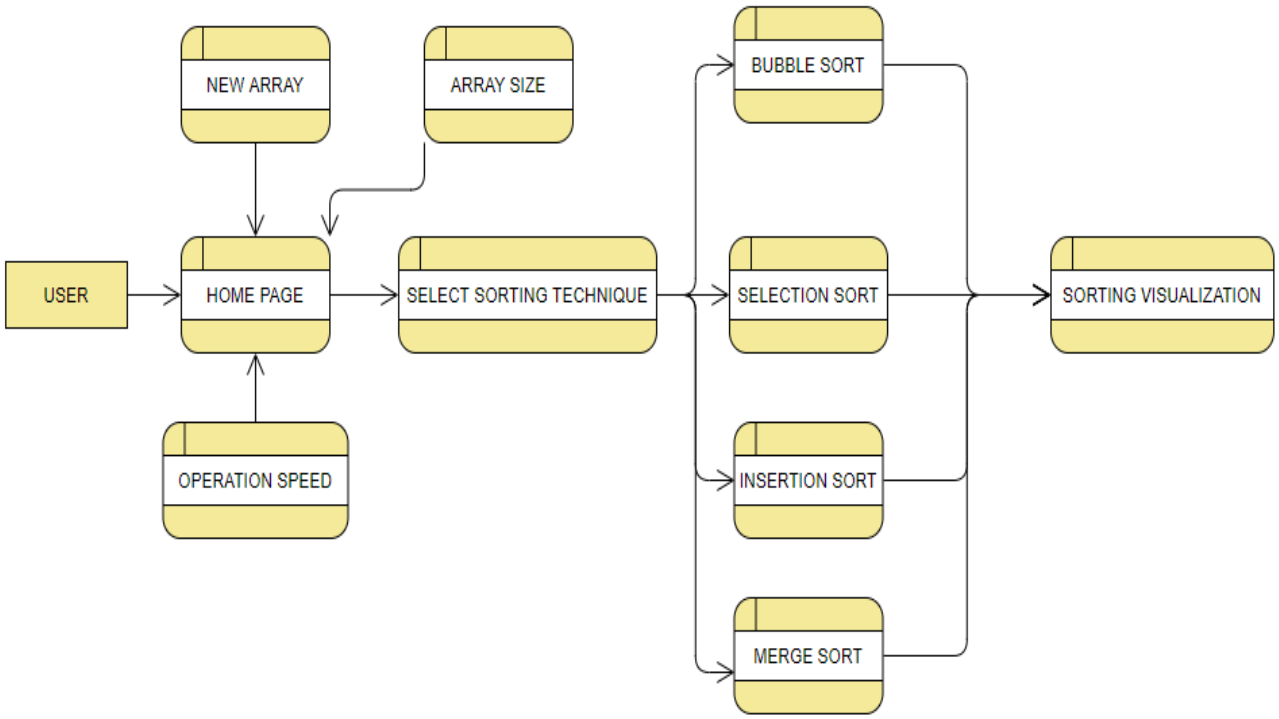
Chapter 4

DESIGN

4.1 Introduction

- The design of our project is quite simple.
- As soon as the user visits our website, he or she will be able to see the user interface of our project.
- UI contains Buttons like Radomize array to rearrange the elements of array. Buttons representing the sorting operation like Bubble sort, Selection sort, Quick sort, Merge sort, Insertion sort.
- UI also contains two range sliders. One for increasing and decreasing the size of the array elements. One for increasing and decreasing the speed of the sorting visualization.
- We cannot increase or decrease array size during the sorting visualization.
- We can increase and decrease speed of sorting visualization before or during the sorting visualization.
- When one sorting operation is started then all the buttons and sliders will get disabled except the speed slider.

4.2 DFD /ER Diagrams



4.3 Module design and organization

1. HOME PAGE

When the user opens this site, home page will appear with several options.

They are:

- New array : To generate a new array.
- Speed : To change the speed of the operation.
- Size : To change the size of the array.

2. SELECT SORTING TECHNIQUE

Bubble sort, Selection sort, Insertion sort, Merge Sort, Quick Sort …

3. SORTING VISUALIZATION

Now the user can watch the sorting technique he selected visually.

Chapter 5
IMPLEMENTATION & RESULTS

5.1 Introduction

Implementation deals with the systematic, disciplined and quantifiable approach to the development, operation and maintenance of our web application. The process involved with the development of web applications is significantly different from the process of developing conventional web application.

CODES :-

HTML Code:

index.html :

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width">
  <title> Sorting Visualizer!</title>
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/css/bo
otstrap.min.css">

  <scriptsrc="https://ajax.googleapis.com/ajax/libs/jquery/3
.5.1/jquery.min.js"></script>

  <scriptsrc="https://cdnjs.cloudflare.com/ajax/libs/popper.
js/1.16.0/umd/popper.min.js"></script>
```

```
  <scriptsrc="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.
2/js/bootstrap.min.js"></script>

  <link href="style.css" rel="stylesheet" type="text/css" />
</head>
<heading>
  <h1 align="center"> Sorting Visualizer </h1>
</heading>
<nav class="row">
  <div class="col gap-2 d-sm-flex">
    <button value="New array" class="btn btn-outline-success
newArray">Randomize Array </button><br>
    <span id="input">
        <span id="speed" > <label>speed</label>
        <input type="range" min="1" max="550" value="150"
        step="10" id="speedInput">
        </span><br>
        <span id=" size "><label>size</label>
        <input type="range" min="4" max="80" step="2"
value="15" id="arraySize" >
        </span>
      </span>
  </div>

      <div class="col gap-2 d-sm-flex justify-content-left">
        <button value="bubble" class="btn btn-primary
bubbleSort button" > BubbleSort</button>
        <button value="selection" class="btn btn-primary
selectionSort button">SelectionSort</button>
        <button value="insertion" class="btn btn-primary
insertionSort button" > InsertionSort</button>
        <button value="merge" class="btn  btn-primary
mergeSort button"> MergeSort</button>
        <button value="quick" class="btn btn-primary
quickSort button"> QuickSort </button>
      </div>
    </nav>
   <body  class="p-3 mb-2 bg-dark">

      <div id="bars" class="container"> </div>
      <!-- <div class="description bub"> This is description
of bubble sort </div> -->
```

```
  </body>
  <script src="js_files/script.js" ></script>
  <script src="js_files/bubbleSort.js" ></script>
  <script src="js_files/selectionSort.js" ></script>
  <script src="js_files/insertionSort.js"></script>
  <script src="js_files/mergeSort.js"></script>
  <script src="js_files/quickSort.js"></script>

</html>
```

CSS Code:

style.css :

```
*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body{

  /* background: linear-gradient(90deg,#2c3e50, #fd746c) */
}
h1{
  color: white;
}
.btn{

  margin: 2px;
  border-radius: 20px 10px 20px 10px;
}
div span{
  color: white;
  font-weight: bold;
  font-size: 0.5cm;
}
.container{
  margin-top: 20px;
  display: flex;
  flex-wrap: nowrap;
  height: 450px;
  width: 100%;
```

```
  justify-content: center;
  transition: 2s all ease;
}
.bar{
  border: 1px solid black;
  background-color: rgb(255,0,234);
  margin: 0.5px;
  width: 20px;
  height: 300px;
  font-weight: bold;
  border-radius: 5px 5px 0px 0px;
  transition: 0.1s all ease;
}
/* .description{
  color: white;
  font-size: 30px;
  float: right;
  display: none;
} */

#input{
  padding: 10px;
}
```

JAVASCRIPT Code

js_files/script.js :

```
var array_size_slider =
document.querySelector("#arraySize");
var speed_input_slider =
document.querySelector("#speedInput");
var new_array_btn = document.querySelector(".newArray");
var container = document.querySelector(".container");
let array = [];

//function to disable the sorting buttons during sorting
function disable_buttons(){

  console.log("disabling buttons");
  document.querySelector(".bubbleSort").disabled = true;
```

```
  document.querySelector(".selectionSort").disabled = true;
  document.querySelector(".insertionSort").disabled = true;
  document.querySelector(".mergeSort").disabled = true;
  document.querySelector(".quickSort").disabled = true;

}

// function to enable the sorting buttons after completion
of sorting
function enable_buttons(){

  console.log("enabling buttons");
  document.querySelector(".bubbleSort").disabled = false;
  document.querySelector(".selectionSort").disabled = false;
  document.querySelector(".insertionSort").disabled = false;
  document.querySelector(".mergeSort").disabled = false;
  document.querySelector(".quickSort").disabled = false;

}

function disable_size_slider(){
  array_size_slider.disabled = true;
}

function enable_size_slider(){
  array_size_slider.disabled = false;
}

function disable_newArray_btn(){
  new_array_btn.disabled = true;
}

function enable_newArray_btn(){
  new_array_btn.disabled = false;
}


function swap(bar1,bar2){
  [bar1.style.height,bar2.style.height] =
[bar2.style.height,bar1.style.height];
}

createNewArray(15);
```

```
var delay = 250;
function wait(milliSec){
  return new Promise((resolve,reject)=>{
    setTimeout(()=>{resolve("")},milliSec);
  });
}

speed_input_slider.addEventListener("input",function(){
  console.log(speed_input_slider.value);
  delay = 600 - parseInt(speed_input_slider.value);
});

array_size_slider.addEventListener("input", function(){
  console.log(array_size_slider.value);
  createNewArray(parseInt(array_size_slider.value));
});
new_array_btn.addEventListener("click",function(){
  createNewArray(parseInt(array_size_slider.value));
});

function createNewArray(noOfBars){

  // var bar = document.createElement("div");
  // container.appendChild(bar);
  array = [];
  bars.innerHTML = "";
  for(let i = 0; i < noOfBars; i++){
    array[i] = Math.floor(Math.random()*200)+10;
  }
  console.log(array);
  for(let i = 0; i<noOfBars; i++){
    var bar = document.createElement("div");
    bar.classList.add("bar");
    bar.style.alignSelf = "flex-end";
    bar.style.height = `${array[i]*2}px`;
    bars.appendChild(bar);
   }
}
```

js_files/bubbleSort.js :

```
var bubbleSort_btn = document.querySelector(".bubbleSort");
async function bubble(){
  console.log("bubble is working");
  var bars = document.querySelectorAll(".bar");
  console.log(typeof(bars.length));
  for(let i=0; i<bars.length-1; i++){
    //console.log("in i loop");
    for(let j=0; j<bars.length-i-1; j++){
      //console.log("in j loop");
      bars[j].style.backgroundColor = "cyan";
      bars[j+1].style.backgroundColor = "cyan";

if(parseInt(bars[j].style.height)>parseInt(bars[j+1].style.h
eight)){
          //console.log("In if block");
          await wait(delay);
          swap(bars[j],bars[j+1]);
      }
      else{
        await wait(delay);
        console.log("");
      }
      if(array[j]>array[j+1]){
        [array[j],array[j+1]] = [array[j+1],array[j]];
      }
      await wait(delay);
      bars[j].style.backgroundColor = "rgb(255,0,234)";
      bars[j+1].style.backgroundColor = "rgb(255,0,234)";

    }
    bars[(bars.length-i-1)].style.backgroundColor = "green";
  }
  console.log(array);
  bars[0].style.backgroundColor = "green";
}

bubbleSort_btn.addEventListener("click",async function(){
  disable_newArray_btn();
  disable_size_slider();
  disable_buttons();
  await bubble();
```

```
  enable_newArray_btn();
  enable_size_slider();
  enable_buttons();
});
```

js_files/selectionSort.js :

```
var selectSort_btn =
document.querySelector(".selectionSort");

selectSort_btn.addEventListener("click",async function(){

  disable_newArray_btn();
  disable_size_slider();
  disable_buttons();
  await select();
  enable_newArray_btn();
  enable_size_slider();
  enable_buttons();

});

async function select(){

  var bars = document.querySelectorAll(".bar");
  for(let i = 0; i < bars.length-1; i++){
    var min_index = i;
    bars[i].style.backgroundColor = "blue";
    for(let j = i+1; j < bars.length; j++){
      bars[j].style.backgroundColor = "cyan";
      await wait(delay);
      if(parseInt(bars[min_index].style.height) >
parseInt(bars[j].style.height)){
        if(min_index != i){
        bars[min_index].style.backgroundColor =
"rgb(255,0,234)";
        }
        min_index = j;
      }
      else{
        bars[j].style.backgroundColor = "rgb(255,0,234)";
      }
    }
```

```
    bars[min_index].style.backgroundColor = "blue";
    await wait(delay);
    console.log("");
    await wait(delay);
    swap(bars[min_index],bars[i]);
    await wait(delay);
    bars[min_index].style.backgroundColor =
"rgb(255,0,234)";
    bars[i].style.backgroundColor = "green";
  }
  bars[bars.length-1].style.backgroundColor = "green";
}
```

js_files/insertionSort.js :

```
var insertSort_btn =
document.querySelector(".insertionSort");
insertSort_btn.addEventListener("click",async function(){
  disable_newArray_btn();
  disable_size_slider();
  disable_buttons();
  await insert();
  enable_newArray_btn();
  enable_size_slider();
  enable_buttons();
});

async function insert(){

  var bars = document.querySelectorAll(".bar");
  for(let i = 1; i < bars.length; i++){

    bars[i-1].style.backgroundColor = "green";
    let j = i;
    while(j>0 && (parseInt(bars[j].style.height)  <
parseInt(bars[j-1].style.height))) {
      bars[j].style.backgroundColor = "#f3f3f3";
      await wait(delay);
      //bars[j-1].style.backgroundColor = "#a3a3a3";
      await wait(delay);
      bars[j].style.backgroundColor = "green";
      bars[j-1].style.backgroundColor = "#f3f3f3";
```

```
      swap(bars[j],bars[j-1]);
      await wait(delay);
      bars[j-1].style.backgroundColor = "green";
      bars[j].style.backgroundColor = "green";

      j--;
    }
    await wait(delay);
  }
  bars[bars.length-1].style.backgroundColor = "green";
}
```

js_files/mergeSort.js :

```
var mergeSort_btn = document.querySelector(".mergeSort");

mergeSort_btn.addEventListener("click",async function (){

  disable_newArray_btn();
  disable_size_slider();
  disable_buttons();
  var bars = document.querySelectorAll(".bar");
  await mergeSort(bars,0,bars.length-1);
  enable_newArray_btn();
  enable_size_slider();
  enable_buttons();
});

async function mergeSort(bars,low,high){


  // for(let i=0; i<=high; i++){
  //   bars[i].style.backgroundColor = "cyan";
  // }
  if(low < high){

    let mid = Math.floor((low+high)/2);
   // let left = bars.slice(low,mid);
   await mergeSort(bars,low,mid);
   await mergeSort(bars,mid+1,high);
   await merge(bars,low,mid,mid+1,high);
  //  let right = bars.slice(mid);
```

```
    // mergeSort(left);
    // mergeSort(right);
    // merge(left,right);
  }
}

async function merge(bars,low,mid,midPlus,high){

  let result = [];
  let i,j,k;
  i = low;
  j = midPlus;
  k=0;

  for(let x=low; x<=mid; x++){
    await wait(delay);
    bars[x].style.backgroundColor = "yellow";
  }
  for(let x=midPlus; x<=high; x++){
    await wait(delay);
    bars[x].style.backgroundColor = "orange";
  }
  while(i <= mid && j <= high){

    if(parseInt(bars[i].style.height) <=
parseInt(bars[j].style.height)){
      result[k] = bars[i].style.height;
      i++;
    }
    else{
      result[k] = bars[j].style.height;
      j++;
    }
    k++;
  }
  while(i <= mid){

    result[k] = bars[i].style.height;
    i++;
    k++;
  }
  while(j <= high){
```

```
    result[k] = bars[j].style.height;
    j++;
    k++;
  }
  for(i = low,j=0; i <=high; i++,j++){
    await wait(delay);
    bars[i].style.height = result[j];
    if(low+high===bars.length-1)
      bars[i].style.backgroundColor = "green";
    else{
      bars[i].style.backgroundColor = "lightGreen";
    }
  }
  console.log("result: "+result);

}
```

js_files/quickSort.js :

```
var quickSort_btn = document.querySelector(".quickSort");

quickSort_btn.addEventListener("click",async function(){
  var bars = document.querySelectorAll(".bar");
  disable_newArray_btn();
  disable_size_slider();
  disable_buttons();
  await quickSort(bars,0,bars.length-1);
  enable_newArray_btn();
  enable_size_slider();
  enable_buttons();
});

async function quickSort(bars,low,high){
  console.log("quick sort called for values low: "+low+"
high: "+high);
  if(low < high){

    let i,j,pivot;
    pivot = low;
    i = low;
    j = high;
     await wait(delay);
     bars[pivot].style.backgroundColor = "red";
```

```
    while(i < j){
      while(parseInt(bars[i].style.height) <=
parseInt(bars[pivot].style.height) && i<high){
        if(i != pivot)
          bars[i].style.backgroundColor = "yellow";
        await wait(delay);
        if(i != pivot)
          bars[i].style.backgroundColor = "rgb(255,0,234)";
        await wait(delay);
        i++;
      }
      await wait(delay);
      bars[i].style.backgroundColor = "yellow";

      while(parseInt(bars[j].style.height) >
parseInt(bars[pivot].style.height) && j > low){
        bars[j].style.backgroundColor = "orange";
        await wait(delay);
        bars[j].style.backgroundColor = "rgb(255,0,234)";
        await wait(delay);
        j--;
      }
      await wait(delay);
      bars[j].style.backgroundColor = "orange";


      if(i < j){
        await wait(delay*2);
        swap(bars[i],bars[j]);
      }
  }

  await wait(delay);
  swap(bars[j],bars[pivot]);
  bars[pivot].style.backgroundColor = "rgb(255,0,234)";
  bars[j].style.backgroundColor = "green";
  await quickSort(bars,low,j-1);
  await quickSort(bars,j+1,high);
  }
  else{
    await wait(delay);
    if((low <= 0 && low < bars.length) || low==high)
      bars[low].style.backgroundColor = "green";
```

```
    }
}
```

5.2 Explanation of Key functions:

Key functions in our project are :

Speed slider : For increasing or decreasing the speed of visualization.

Size slider : For increasing or decreasing the size of the array.

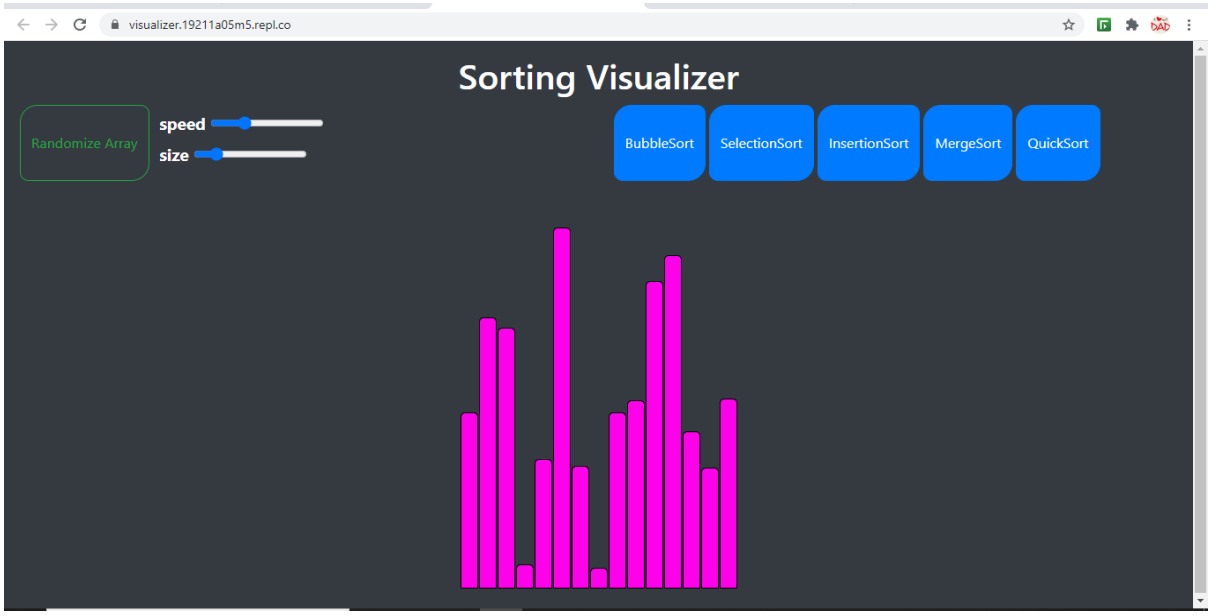
Buttons : Five buttons representing the five sorting techniques.

On clicking them, the corresponding sorting visualization will be done

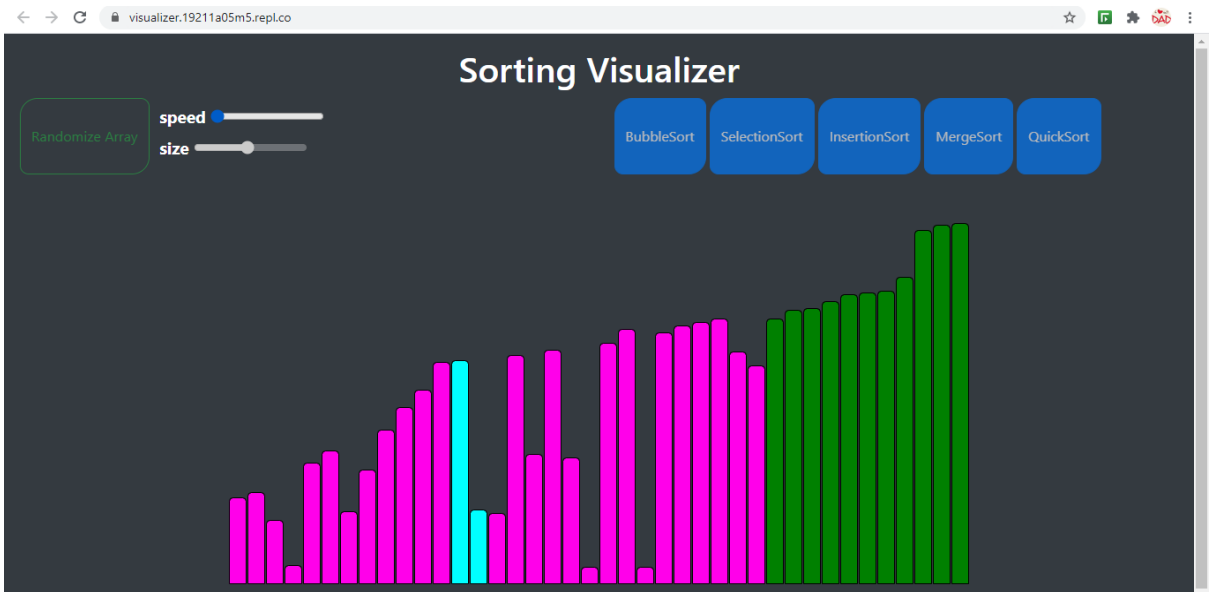
Randomize array button, to rearrange the elements of the array.

5.3 Method of Implementation

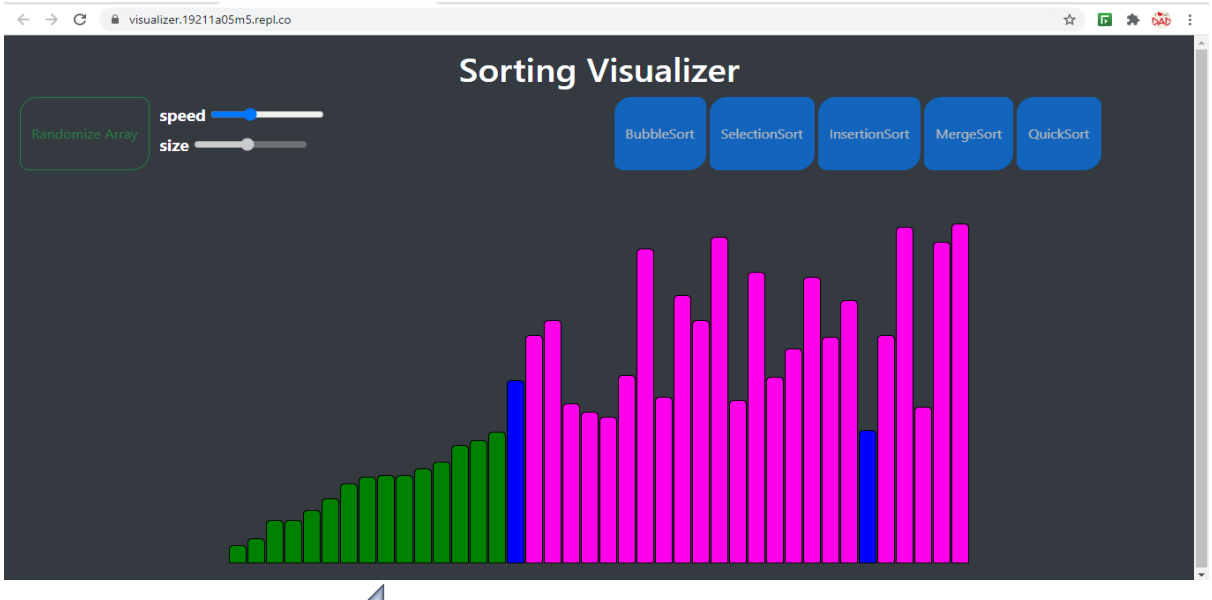
5.3.1 Output Screens



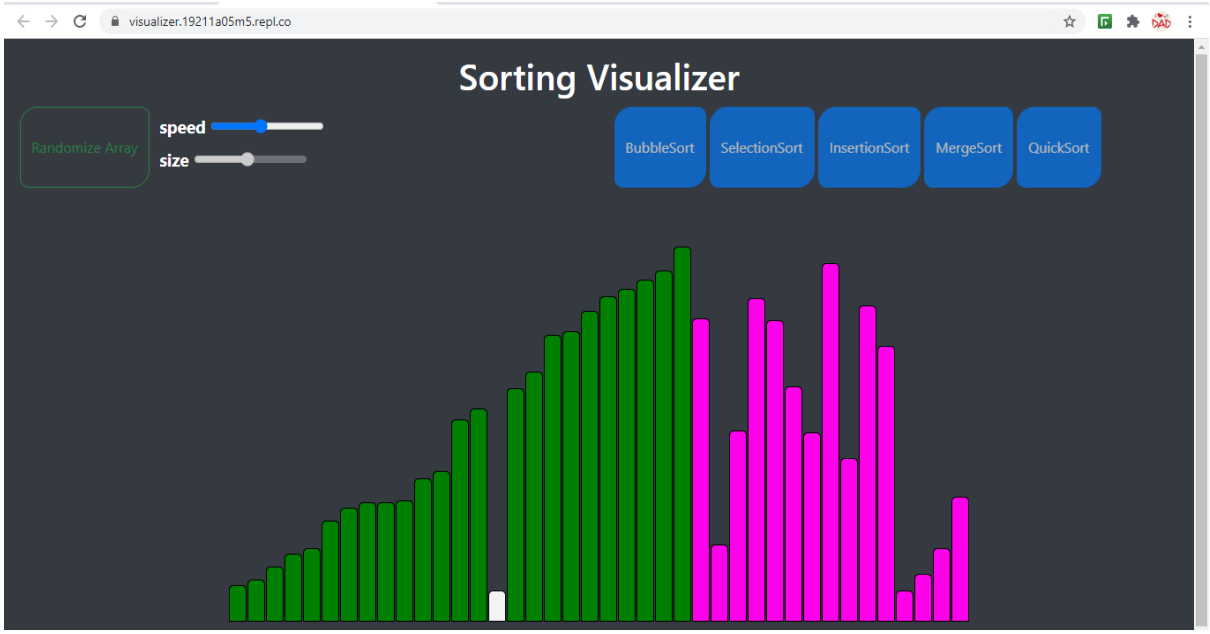
This is UI as soon as user opens this website



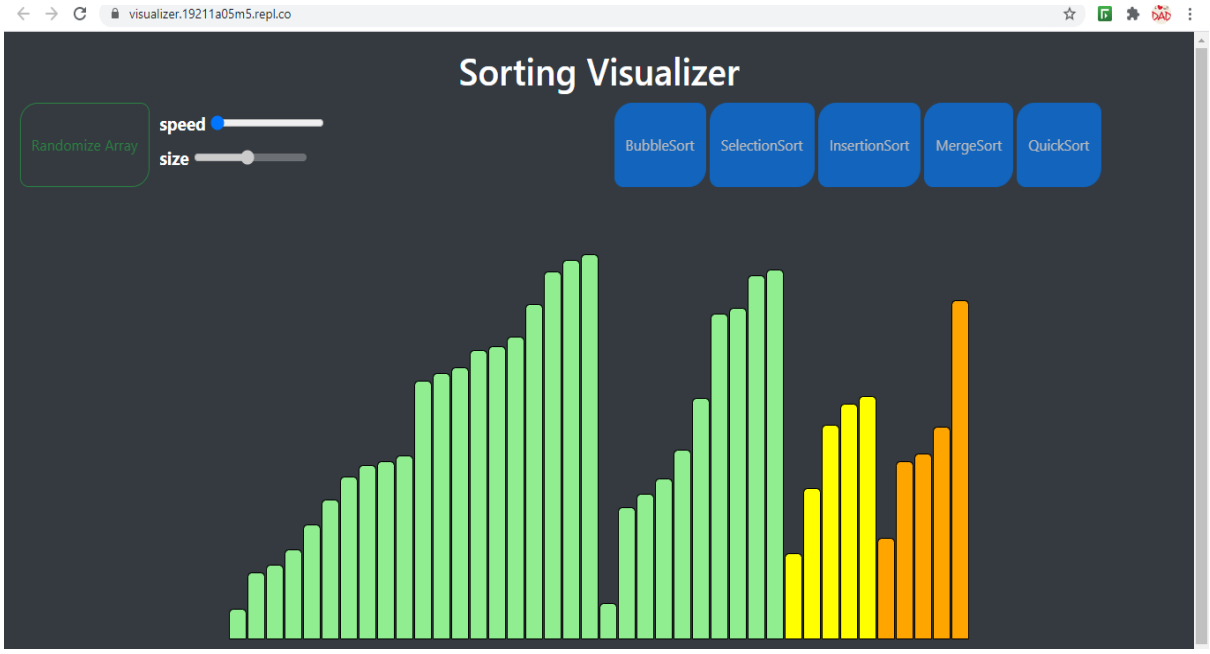
During Bubble Sort



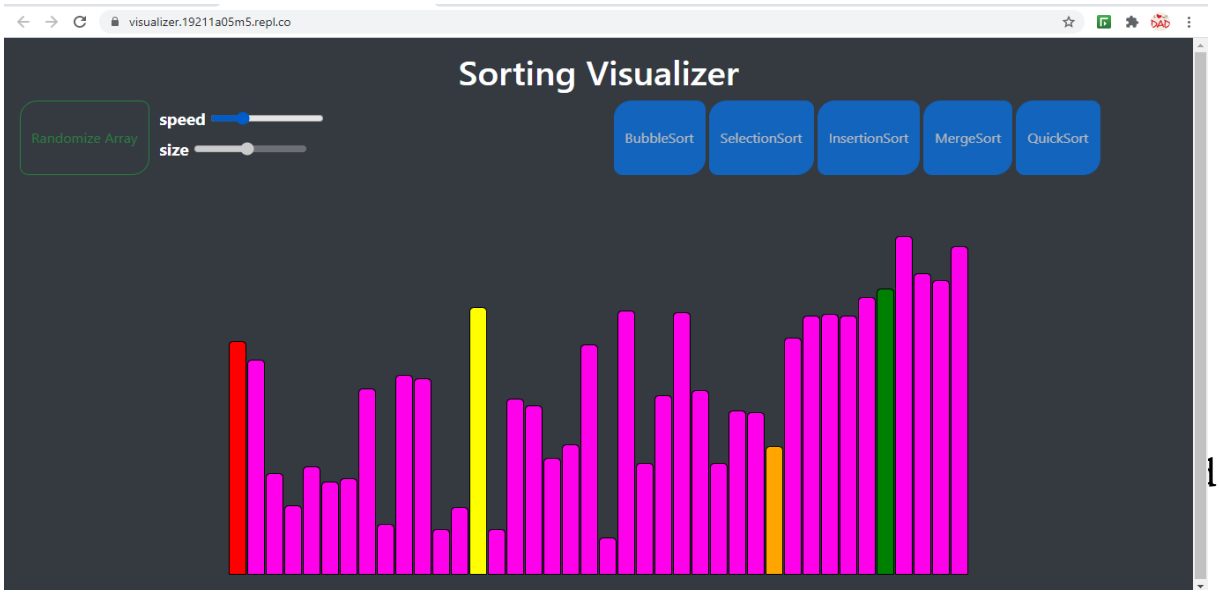
During Selection Sort



During Insertion Sort



During Merge Sort



During Quick Sort

5.3.3 Result Analysis

- After user visiting our website Sorting Visualizer
- Firstly he can randomize the array if he want.
- Or He can also increase or decrease the size of the array
- Then he have to select the sorting technique which he wanted to visualize from these sorting techniques Bubble sort, Selection sort, Insertion sort, Merge sort, and Quick sort.
- As soon as he selects the sorting by clicking on the button, then the visualization of that sorting will be done.
- During the sorting, the user can increase or decrease the speed of the sorting visualization by using the speed slider.
- After the completion of the sorting visualization, all the elements will turn into green colour.

Chapter 6
TESTING & VALIDATION

6.1 Introduction

- The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.
- Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

6.2 Design of test cases and scenarios

- Testing a web application is not easy than testing a static website. Functionality testing is the most important thing to be performed while testing a web application.
- The web application may contain much-complicated functionality so test cases need to be very careful while testing.
- Our project includes test cases which are accepted across maximum number of scenarios.

6.3 Validation

- Validation ensures that your website complies with the standards accepted by most web designers.
- Our web application can be validated as it is consisting of less amount of code when compared to other commercial web apps so it is bug free and is any bug is found is can be removed easily.
- That also means that it will be accessible to more people, across more web browsers and operation systems. Having an accessible website is also regarded as good web design practice.

Chapter 7

CONCLUSION & FUTURE WORK

Conclusion:

- In this project the user can see the visualization of the sorting technique he selected.
- We have implemented bubble sort, selection sort, insertion sort, merge sort and quick sort.

OUTCOMES:

- Able to see sorting process step by step.
- Understanding sorting techniques.
- Differentiating the different sorting techniques.

Future work:

- We will add brief description of the every sorting technique.
- We will add the visualization of some other sorting techniques.
- We will also add the visualization of algorithms like BFS, DFS, Dijkstras shortest path, Krushkal's minimum cost spanning tree etc.

References

- www.crio.do/projects
- W3schools
- Javatpoint
- GeeksforGeeks
- Tutorialspoint
- Github