PLAYSORTS

PROJECT REPORT

Project submitted in partial fulfillment of the degree

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

M. PranayKumarReddy (O161064)

B. Jeenath (O161615)

M. Sreelatha (O161315)

M. Jithendar (O161050)

G. Elumalai (O162089)

Under the supervision of

Ms. N. Madhavi Latha

Dept. of Computer Science and Engineering,

RGUKT Ongole Campus,

Prakasam Dist - 523001,

November 2021.

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the project report entitled **"PLAYSORTS"** submitted by **PranayKumar Reddy M,** Roll No: **O161064, Jeenath B,** Roll No: **O161615, Sreelatha M,** Roll No: **O161315**, **Jithendar M,** Roll No: **O161050**, **Elumalai G,** Roll No: **O162089** to the Department of Computer Science and Engineering, Rajiv Gandhi University of Knowledge Technologies, RGUKT Ongole Campus, during the academic year 2020-2021 is a partial fulfillment for the award of Undergraduate degree of Bachelor of Technology in Computer Science and Engineering, is a bonafide record carried out by him under my supervision. The project has fulfilled all the requirements as per regulations of this institute and in my opinion reached the standard for submission.

K. Sandhya
Branch HOD
Dept of CSE,
RGUKT Ongole,
Ongole - 523001.

N. Madhavi Latha
Asst. Professor
Dept of CSE,
RGUKT Ongole,
Ongole - 523001.

Date :
Place : Ongole.

ACKNOWLEDGEMENT

We would like to express our special thanks to **Ms. N. Madhavi Latha** madam and our Head of the Department **Ms. K. Sandhya** madam who gave us the golden opportunity to do this project on the topic **PlaySorts**. It helped us a lot in doing a lot of research and we came to know about a lot of things related to this topic.

We are highly indebted to our project mentor **Ms. N. Madhavi Latha** for their guidance and constant encouragement as well as for providing necessary information regarding the project and also for their kind cooperation, encouragement and their support in completing the project.

We would like to express our special gratitude and thanks to branch Head of The Department **Ms. K. Sandhya** madam for giving us such attention and time.

We have taken efforts in this project, to successfully complete this project, many people have helped us. We would like to thank all those who are related to this project. My special thanks and appreciation also go to my friends in developing the project and people who have willingly helped me out with their abilities.

ABSTRACT

PlaySorts is a website that illustrates the internal working theorem of sorting algorithms. PlaySorts is a visualization of the sorting algorithms like Bubble Sort, Merge Sort, Insertion Sort, Quick Sort, Heap Sort and Selection Sorts. PlaySorts helps students from any stream to easily understand the working process of sorting so that it enhances learning the subject and to master it by having a picturization proposal. Students can easily access PlaySorts and can learn by playing by selecting various sorting algorithms and can understand easily. PlaySorts has different User Interfaces that are needed to work with algorithms like List Size, Delay Speed, Generate, Sort, Get Last unsorted list and a menu to choose a sort option.It shows the Time and Space complexities in their worst,average and best cases to their respective sorting algorithms. Thus, PlaySorts sorts the generated elements as per the selected sorting algorithm.

The Student can :

- Access PlaySorts.
- Input the list size.
- Choose a sort option.
- Can give the delay speed to generate.
- Can generate the list of elements using the Generate option.
- Finally, the Sort option would sort the listed elements based on the sorting algorithm.
- Users can have the last unsorted list.

1. INTRODUCTION

This section describes the main objectives and aims of the project. It also describes our motivation which lies behind our project implementation.

1.1 Project Aim and Objectives:

“PlaySorts” is a web application which is mainly implemented to know how the sort algorithms work internally and shows in a picturization form. There are six different sorting algorithms such as Bubble Sort, Insertion Sort, Quick Sort, Merge Sort, Selection Sort and Heap Sort. These six sorting algorithms work on eight different factors such as array size, space complexity, time complexity at their worst, average and best cases, choosing any sort algorithm, the delay speed, generating array elements, getting last unsorted elements and finally the sort option after choosing all the requirements from the options. There is no algorithm that has all the properties and so the choice of algorithm depends on the application.

Each sorting algorithm has its own properties and factors aligned to it. Each algorithm visualizes and shows how a sorting system works on elements and operates. Shows that there is no best sorting algorithm because every sorting algorithm best suits various implementations. We came to know the disadvantages and advantages of every sorting algorithm and when and how to use their implementations in real life projects.It shows that the worst-case asymptotic behavior is not always the deciding factor in choosing an algorithm.

Users can choose the different sort options available on the website like array size, sorting option,delay speed to generate or sort them, generate the array elements by the system automatically, sort option to sort them. Users can watch its functionality by graphing them using different colors that enhance the system in a better way.Thus, an array of random values is generated and are drawn as lines in the window.Different colors are used to indicate which elements are being compared, sorted and unsorted. Thus, this is the main aim of this project **“PlaySorts”.**

1.2. MOTIVATION:

Now-a-days having a degree is not a matter, having knowledge and skills is such a big task in a student's life. The foundation of having a strong base in a subject is the way the students choose to learn new things. Most of them choose either rote learning or byharding. However, that is not a good practice. It makes students remember concepts for a very short period and thus leads to not having a grip on the subject.

It is easier to remember a picturization than a written format. A human brain can easily format visuals instead of long codes to understand the algorithms. Thus, the implementation of "**PlaySorts"** happened with the visualization of sorting algorithms and their working implementation with a user friendly environment. Users can easily understand the website and learn the sorting algorithms which are the topics that play a major role in computer science. Anyone from other specializations also can easily understand this process and can master them. It helps students to learn them, not by rote learning, thus they can remember them for a long time as they understand the concepts well. Thus, a student learns the subject not a single topic by byharding it.

2. SYSTEM ANALYSIS

Analysis is the detailed study of the various operations performed by a system and their relationships within and outside of the system. A key question is: What must be done to solve the problem? One aspect of analysis is defining the boundaries of the system and determining whether the system can be implemented within the specified conditions and does not interrupt other systems. During analysis, data are collected on the available files, decision points, and problems handled by the present system.

2.1 Software Requirements Specification:

A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform it. It also describes the functionality of the product needs to fulfill all stakeholders (business, users) needs. A typical SRS document includes the purpose of the project, overall description, and specific requirements to build the software.

2.1.1 Problem Statement:

It is hard for anyone to learn concepts by byharding or rote learning. It is mainly applicable to students who depend on rote learning and learn concepts. It is not such a good practice to learn the subject and to learn anything new as it remains in the brains of students for such a short period of time. It helps temporarily to attempt exams or anything but does not provide good practice to learn the actual subject. A student without subject knowledge but with good grades is nothing but a book without content. It doesn’t make sense anymore. Such that the students' hard work over years will be bin.

2.1.1 Project Description:

Learning something just by reading books might be difficult.In such cases, Visualization methods help in a better way. So we chose the most important concepts for a student i.e; Sorting Algorithms. Those are difficult to understand and it's easy to get confused. We believe that visualizing sorting algorithms can be a great way to better understand their functionality while having fun.

2.1.3 Existing System:

There are many ways where students can learn academic subjects. It is a traditional way for students to learn anything from books. Books provide the students all the needed information to learn a subject. The bridge between book knowledge and a student is always a teacher. A lecturer physically chooses the subject and teaches the student on boards. The sorting algorithms or any subject will be learned by students when teached by lecturers.The lecturer chooses a topic and discusses it on the board with the students. Students need to clarify their doubts instantly. This is the general existing system.

2.1.4 Proposed System:

With the problems arising in the existing system, we implemented **PlaySorts,** a visualization of sorting algorithms.

- PlaySorts is a web application that can be accessible to anyone who wants to learn sorting algorithms while having fun.
- It enables students to learn sortings in a visualization mode.
- Users can learn any sorting algorithm by choosing what is available on the website.
- Many options available on the website enhance the students to learn sorting algorithms in an easy way.

2.1.5 Functional Requirements :

Functional Requirements in an SRS document (software requirements specification) indicate what a software system must do and how it must function; they are product features that focus on user needs.

User Management :

User management deals with the responsibilities of selecting a sorting algorithm and all the required options related to the individual algorithm and performing sorting operation over the array elements and observing the output.

Admin Management :

As an admin, he/she needs to update and add new techniques to the system. Adds additional algorithms and makes them available to the users.

2.1.6 Non- Functional Requirements :

Non Functional Requirements define system attributes. They serve as constraints or restrictions on the design of the system across the different backlogs. These are also known as system qualities, they ensure the usability and effectiveness of the entire system.

Security:

PlaySorts is so secure that no one else other than admin can change the algorithms code and the system's functionality.

Performance:

Performance of the system is high because of its simplicity and can be accessible to any number of users. In case of huge accessibility, it endures the traffic and performs well over a number of systems.

Maintainability:

Maintaining PlaySorts is very easy for both users and admins. Users just need to access it directly without any authentication and admins can add additional algorithms.So, maintenance gets easier.

2.1.7 Software Requirements:

The software requirements used for PlaySorts are :

- Operating System : Linux or Windows version 7 or above.
- User Interface : UI and UX using HTML and CSS.
- Programming Language : Python
- Software : Visual Studio

2.1.8 Hardware Requirements:

The Hardware requirements used for PlaySorts are:

- Processor: Intel i3 or higher versions.
- RAM : Minimum 2GB
- Space on Hard Disk: Minimum 5GB
- Device : Any device which supports web browsers.

2.1.9 Software Tools Used:

HTML :

HTML stands for HyperText Markup Language. It is used to design web pages using a markup language. HTML is the combination of Hypertext and Markup language. Hypertext defines the link between the web pages. A markup language is used to define the text document within a tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most markup languages (e.g. HTML) are human-readable. The language uses tags to define what manipulation has to be done on the text.

HTML is a markup language used by the browser to manipulate text, images, and other content, in order to display it in the required format. HTML was created by Tim Berners-Lee in 1991. The first-ever version of HTML was HTML 1.0, but the first standard version was HTML 2.0, published in 1999.

Elements and Tags: HTML uses predefined tags and elements which tell the browser how to properly display the content. Remember to include closing tags. If omitted, the browser applies the effect of the opening tag until the end of the page.

HTML page structure: The basic structure of an HTML page is laid out below. It contains the essential building-block elements (i.e. doctype declaration, HTML, head, title, and body elements) upon which all web pages are created.

<DOCTYPE! html>: This is the document type declaration (not technically a tag). It declares a document as being an HTML document. The doctype declaration is not case-sensitive.

<html>: This is called the HTML root element. All other elements are contained within it.

<head>: The head tag contains the "behind the scenes" elements for a webpage. Elements within the head aren't visible on the front-end of a webpage. HTML elements used inside the <head> element include:

- <title>
- <base>
- <noscript>
- <script>
- <meta>

<body>: The body tag is used to enclose all the visible content of a webpage. In other

words, the body content is what the browser will show on the front-end.

An HTML document can be created using any text editor. Save the text file using **.html** or **.htm**. Once saved as an HTML document, the file can be opened as a web page in the browser.

Features of HTML:

- It is easy to learn and easy to use.
- It is platform-independent.
- Images, videos, and audio can be added to a web page.
- Hypertext can be added to the text.
- It is a markup language.

CSS:

CSS (Cascading Style Sheets) is a stylesheet language used to design the webpage to make it attractive. The reason for using this is to simplify the process of making web pages presentable. It allows you to apply styles to web pages. More importantly, it enables you to do this independent of the HTML that makes up each web page.

JavaScript:

JavaScript is the world's most popular lightweight, interpreted compiled programming language. It is also known as scripting language for web pages. It is well-known for the development of web pages, many non-browser environments also use it. JavaScript can be used for Client-side developments as well as Server-side developments.

JavaScript can be added to your HTML file in two ways:

- **Internal JS:** We can add JavaScript directly to our HTML file by writing the code inside the <script> tag. The <script> tag can either be placed inside the <head> or the <body> tag according to the requirement.
- **External JS:** We can write JavaScript code in other file having an extension .js and then link this file inside the <head> tag of the HTML file in which we want to add this code.

What can we build using JavaScript ?

JavaScript is a widely-used programming language. Given below are some domains/products that can be built using JavaScript:

- **Websites:** JavaScript helps us to add behavior of our website. It helps users to interact with the website. For eg. clicking on buttons, saving details, uploading details on the website, etc.
- **Web Servers:** We can make robust server applications using JavaScript. To be precise we use JavaScript frameworks like Node.js and Express.js to build these servers.
- **Game Development:** In the Game Development industry, JavaScript is used widely. With the addition of HTML5 Canvas, it's now possible to make 2D and 3D games in JavaScript very efficiently.
- **3D Drawings:** JavaScript in addition with HTML Canvas is used to make three-dimensional graphics.

Advantages of JavaScript:

1. Speed:

Since JavaScript is an ‘interpreted’ language, it reduces the time required by other programming languages like *Java* for compilation. JavaScript is also a client-side script, speeding up the execution of the program as it saves the time required to connect to the server.

2. Simplicity:

JavaScript is easy to understand and learn. The structure is simple for the users as well as the developers. It is also very feasible to implement, saving developers a lot of money for developing dynamic content for the web.

3. Popularity:

Since all modern browsers support JavaScript, it is seen almost everywhere. All the famous companies use JavaScript as a tool including Google, Amazon, PayPal, *etc.*

4. **Interoperability*:***

JavaScript works perfectly with other programming languages and therefore numerous developers prefer it in developing many applications. We can embed it into any webpage or inside the script of another programming language.

5. Server Load:

As JavaScript operates on the client-side, data validation is possible on the browser itself rather than sending it off to the server. In case of any discrepancy, the whole website needs not to be reloaded. The browser updates only the selected segment of the page.

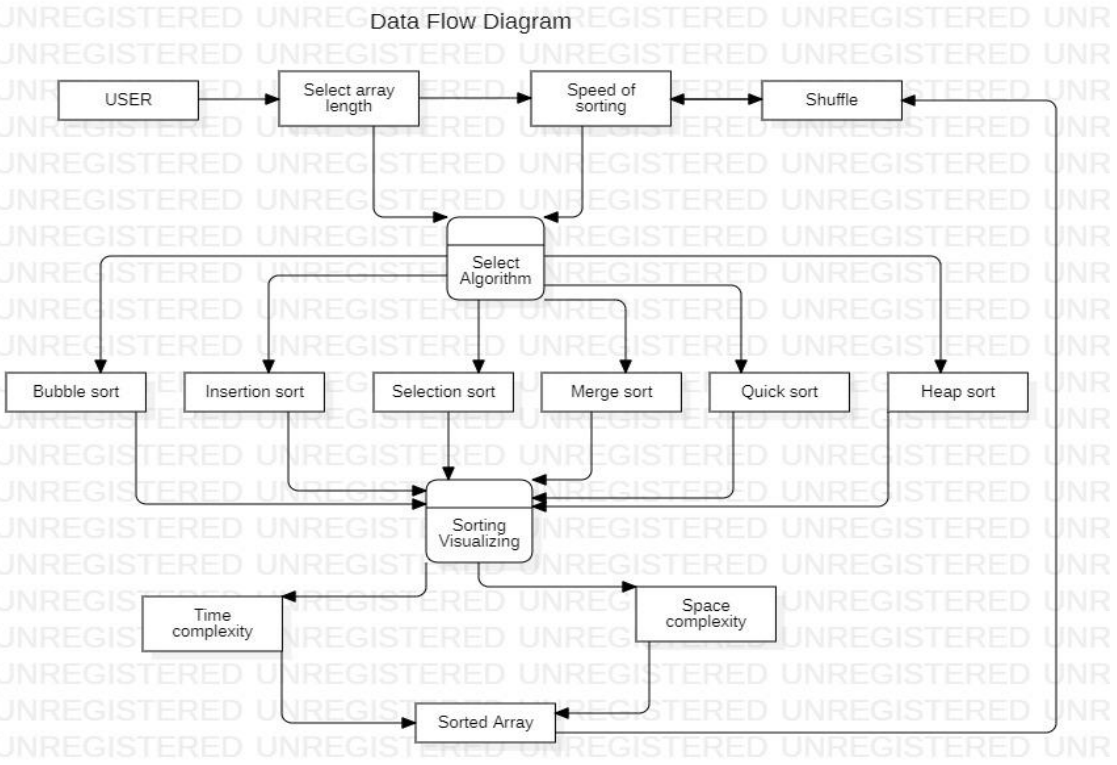
3.System Design

System Design is the process of defining the components, modules, interface and data for a system to satisfy specified requirements. It is the process of creating or altering systems along with the processes, practices, models and methodologies used to develop them. System Design is the process of designing architecture of the developing software application. The purpose of System Design is to provide sufficient detailed data and information about the system and its elements to enable it to be consistent with architectural entities as defined in modules and views of the system architecture.

3.1 Data Flow Diagram:

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation. DFD graphically represents the functions, or processes, which capture, manipulate, store, and distribute data between a system and its environment and between components of a system. The visual representation makes it a good communication tool between User and System designer. Structure of DFD allows starting from a broad overview and expanding it to a hierarchy of detailed diagrams.

The below diagram shows the Data Flow Diagram of PlaySorts. User is an external entity where he can perform many operations based on the requirements he needs. Where the user can choose a sorting algorithm, element size, generating the elements, time and space complexity labels etc. Finally the users get a sorted array by having a sort button.

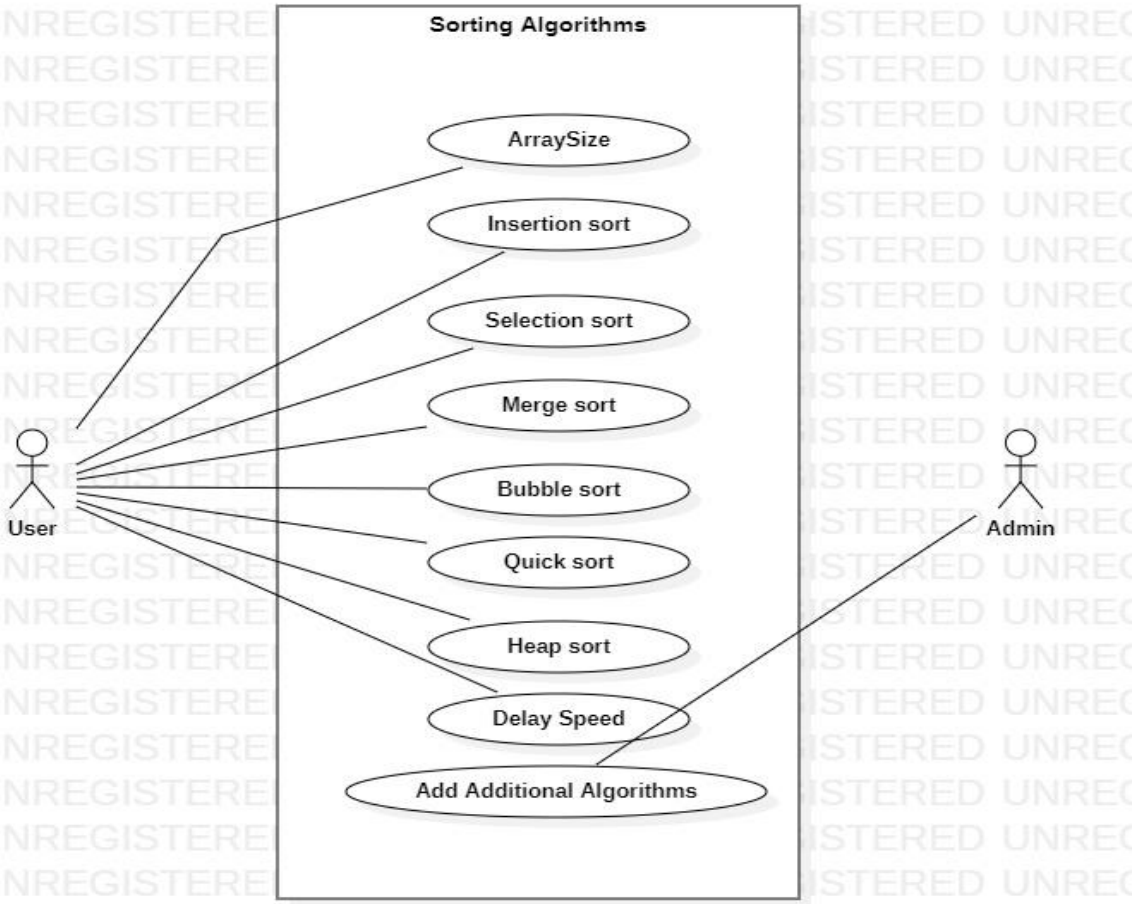


3.2. UML Diagrams:

Unified Modeling Language (UML) is a general purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. It is quite similar to blueprints used in other fields of engineering. UML is not a programming language, it is rather a visual language. We use UML diagrams to portray the behavior and structure of a system. UML helps software engineers, businessmen and system architects with modeling, design and analysis. UML is linked with object oriented design and analysis. UML makes the use of elements and forms associations between them to form diagrams

3.2.1 Use Case Diagram :

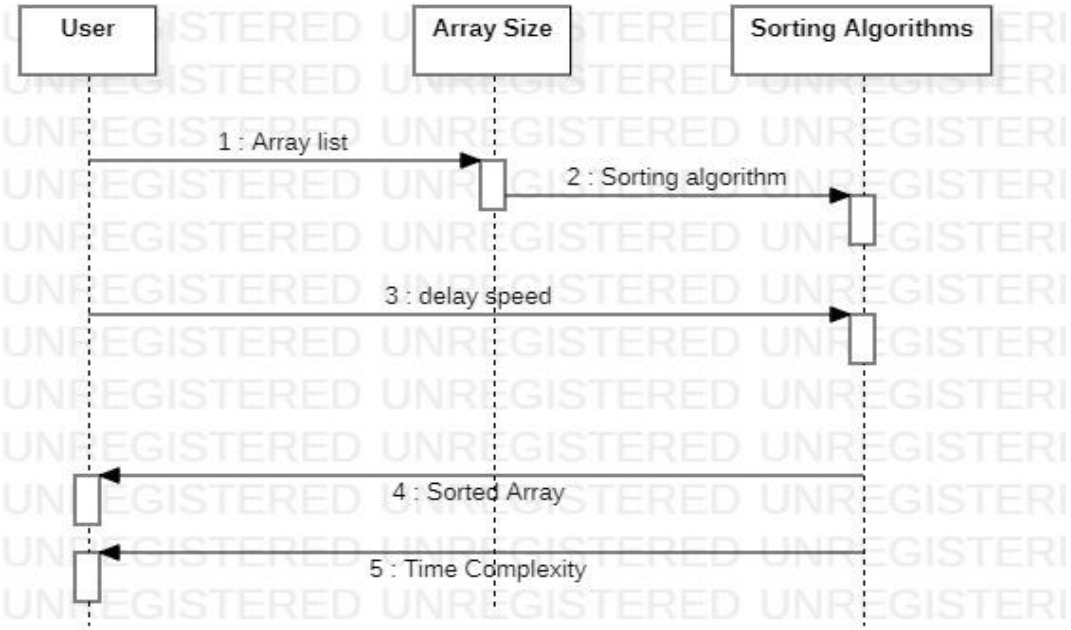
A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and different use cases in which the user is involved. A set of use cases describe the complete functionality of the system at a particular level of detail and it can be graphically denoted by the use case diagram.



In the above diagram, The user can select any sorting algorithm that is available, can choose array size, delay speed. The Admin's responsibility is to add additional algorithms.

3.2.2 Sequence Diagram:

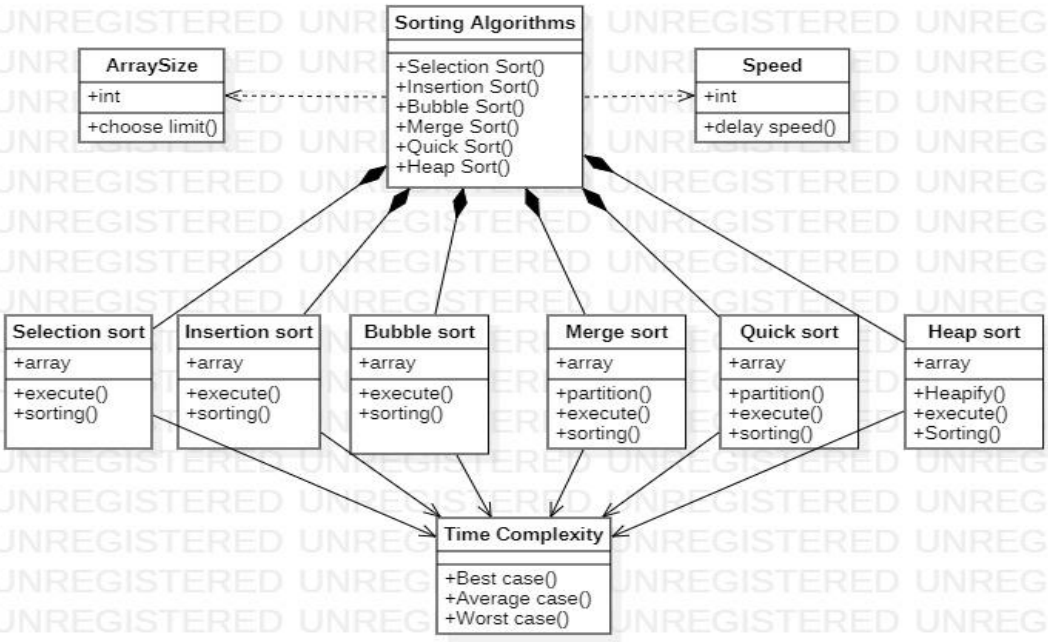
Sequence diagrams are an easy and intuitive way of describing the behavior of a system by viewing the interaction between the system and the environment. A sequence diagram shows an interaction arranged in a time sequence. A sequence diagram has two dimensions: vertical dimension represents time; the horizontal dimension represents the object's existence during the interaction.



This is the sequence diagram representation of PlaySorts having five stages. It represents the stages from where the user can choose array size,sorting algorithm, delay speed, sorted array to Time complexity.

3.1.3 Class Diagram:

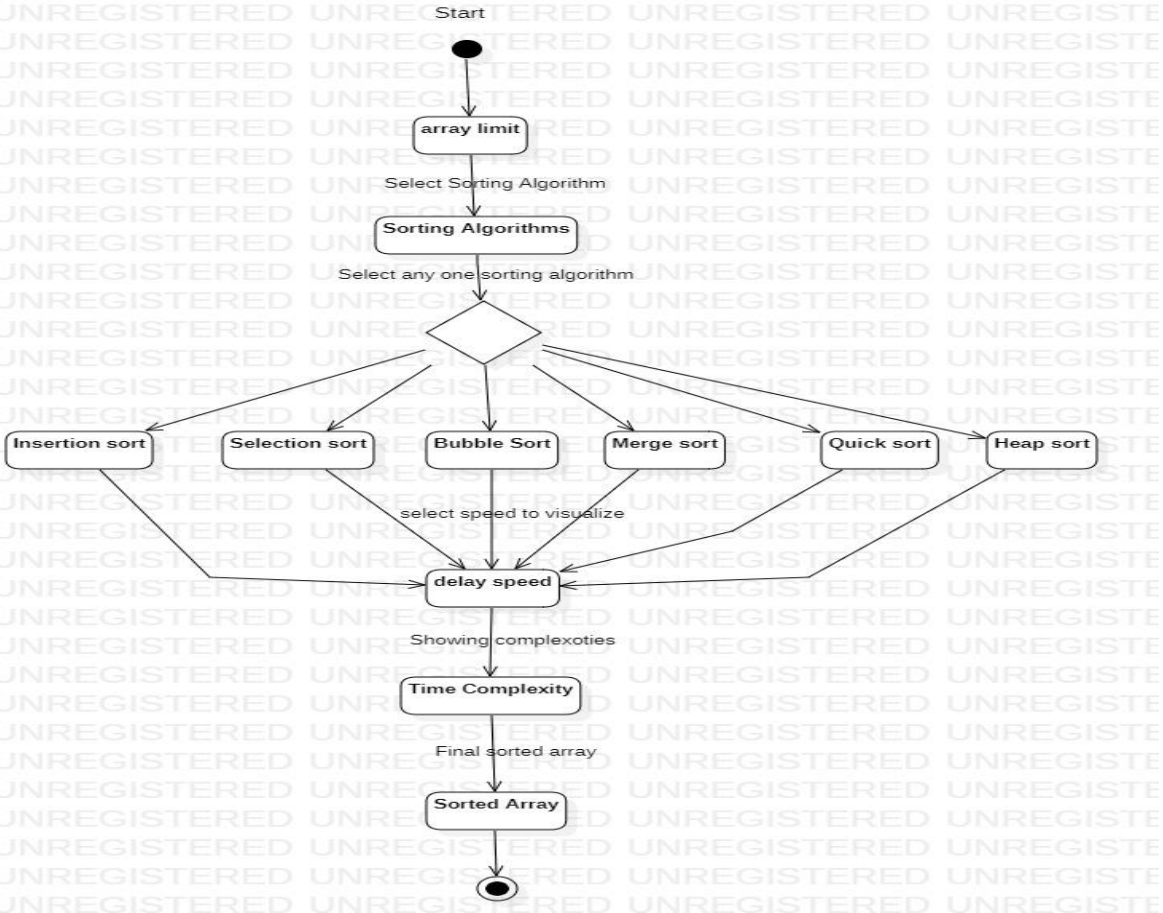
Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. It is also known as a structural diagram.



This is the Class Diagram representation of PlaySorts. It is mainly having ten modules. ArraySize module has arra size number, Sorting algorithms module has different sorting options, Speed module has speed delay number and sorting methods and finally the Time Complexity module is having best,average and worst cases.

3.1.4 State Chart Diagram:

State chart diagram is one of the five UML diagrams used to model the dynamic nature of a system. They define different states of an object during its lifetime and these states are changed by events. Statechart diagrams are useful to model the reactive systems. Reactive systems can be defined as a system that responds to external or internal events. Statechart diagram describes the flow of control from one state to another state. States are defined as a condition in which an object Dept.of CSE 26 exists and it changes when some event is triggered. The most important purpose of a Statechart diagram is to model the lifetime of an object from creation to termination. Statechart diagrams are also used for forward and reverse engineering of a system.



4. SYSTEM IMPLEMENTATION

System implementation is the process of defining how the information system should be built i.e., physical design of the application, ensuring that the information system is operational and ensuring that the information system meets quality standards. This phase is done after the completion of system design.

Below code shows the different implementation pages of PlaySorts. Each algorithm has its own Javascripted codes as shown in below.

4.1 Coding and Implementation:

4.1.1 BubbleSort.js :

```
async funcion doBubbtleSort(array) {
    var length = array.length;
    for (var i = 0; i < length; i++) {
        for (var j = 0; j < length - 1; j++) {
            if (isStopTriggered) return;
            await prepareSwappingStep(array[j], array[j + 1]);
            if (array[j] > array[j + 1]) {
                swapArrItem(array, j, j + 1);
                await swap(array[j], array[j + 1]);
            }
await endSwappingStep(array[j], array[j + 1]);
        }}
```

```
return array;}
async function doEnhancedBubbleSort(array) {
    var length = array.length;
    var isSwapped;
    var count = 1;
    do {
        isSwapped = false;
        for (var i = 0; i < length - count; i++) {
            if (isStopTriggered) return;
            await prepareSwappingStep(array[i], array[i + 1]);
            if (array[i] > array[i + 1]) {
                isSwapped = true;
                swapArrItem(array, i, i + 1);
                await swap(array[i], array[i + 1]);
            }
            await endSwappingStep(array[i], array[i + 1]);
        }
        count++;
    }
    while (isSwapped == true)
    return array;
}
```

4.1.2. HeapSort.js:

```
async function doHeapSort(array) {
    let length = array.length;
    let lastParentIndex = Math.floor(length / 2 - 1);
    while (lastParentIndex >= 0) {
        await heapify(array, length, lastParentIndex);
        lastParentIndex--;
    }
    for (let i = length - 1; i >= 0; i--) {
        if (isStopTriggered) return;
        await changeBackgroundColor(array[0], 'orange');
        await changeBackgroundColor(array[i], 'red');
        await swap(array[0], array[i]);
        swapArrItem(array, i, 0);
        await endSwappingStep(array[0], array[i]);
        await heapify(array, i, 0);
    }
}
async function heapify(array, length, index) {
    if (isStopTriggered) return;
    let largestIndex = lastLargestIndex = index;
    await changeBackgroundColor(array[largestIndex], 'orange');
    let left = index * 2 + 1;
    let right = left + 1;
    await changeBackgroundColor(array[left], 'red');
    await changeBackgroundColor(array[right], 'blue');
if (left < length && array[left] > array[largestIndex]) {
        largestIndex = left;
```

```
if (right < length && array[right] > array[largestIndex]) {
        // await changeBackgroundColor(array[largestIndex], '#343a40');
        largestIndex = right;
        // await changeBackgroundColor(array[largestIndex], 'orange');
    }
    await changeBackgroundColor(array[lastLargestIndex], '#343a40');
    await changeBackgroundColor(array[largestIndex], 'orange');
    if (largestIndex != index) {
        await swap(array[index], array[largestIndex]);
        swapArrItem(array, index, largestIndex);
        await resetHeapColor(array, index, left, right);
        await heapify(array, length, largestIndex);
    } else {
        await resetHeapColor(array, index, left, right);
    }
}
async function resetHeapColor(array, index, left, right) {
    await changeBackgroundColor(array[index], '#343a40');
    await changeBackgroundColor(array[left], '#343a40');
    await changeBackgroundColor(array[right], '#343a40');
}
```

4.1.3. Insertion Sort:

```
async function doInsertionSort(array) {
    var length = array.length;
    for (var i = 1; i < length; i++) {
        var key = array[i];
        var keyHeight = $(`#item-${key}`).height();

        await changeBackgroundColor(array[i], 'red');
        await changeHeightAndId(key, 0, 'key');

        var j = i - 1;
        while (j >= 0) {
            if (isStopTriggered) return;
            if (array[j] > key) {
                await insert(array[j], 'key');
                array[j + 1] = array[j];
                j--;
            } else {
                break;
            }
        }
        array[j + 1] = key;

        await changeHeightAndId('key', keyHeight, key);
        await changeBackgroundColor(array[i], '#343a40');
    }
    return array;
}
```

```
async function changeHeightAndId(id, height, newId) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            $(`#item-${id}`).height(height);
            $(`#item-${id}`).prop('id', `item-${newId}`);
            resolve()
        }, window.delaySpeed);
    });
}
async function insert(current, next) {
    return new Promise((resolve, reject) => {
        setTimeout(() => {
            console.log($(`#item-${current}`).height());
            $(`#item-${next}`).height($(`#item-${current}`).height());
            $(`#item-${current}`).height(0);

            var valTemp = $(`#item-${current}`).html();
            if (valTemp != '' && valTemp != undefined && valTemp != null) {
                $(`#item-${current}`).html($(`#item-${next}`).html());
                $(`#item-${next}`).html(valTemp);
            }
            $(`#item-${current}`).prop('id', 'current');
            $(`#item-${next}`).prop('id', 'next');
            $(`#current`).prop('id', `item-${next}`);
            $(`#next`).prop('id', `item-${current}`);
            resolve()
        }, window.delaySpeed);
    });
}
```

4.1.4 MergeSort.js:

```
async function doMergeSort(array) {
    await mergeSort(array, array, 0);
}
async function mergeSort(originalArray, array, startIndex) {
    if (isStopTriggered) return;
    if (array.length < 2) {
        return array;
    }
    var middle = Math.floor(array.length / 2);
    var leftArray = array.slice(0, middle);
    var rightArray = array.slice(middle, array.length);

    var leftSortedArray = await mergeSort(originalArray, leftArray, startIndex);
    var rightSortedArray = await mergeSort(originalArray, rightArray, startIndex + leftSortedArray.length);

    return mergeSortedArrays(originalArray, leftSortedArray, rightSortedArray, startIndex);
}
async function mergeSortedArrays(originalArray, leftSortedArray, rightSortedArray, startIndex) {
    var sortedArray = [],
        leftArrIndex = 0,
        rightArrIndex = 0;
          while (leftArrIndex < leftSortedArray.length && rightArrIndex < rightSortedArray.length) {
        let leftMinValue = leftSortedArray[leftArrIndex],
            rightMinValue = rightSortedArray[rightArrIndex],
            minimumValue = 0;
```

```
            await changeBackgroundColor(leftMinValue, 'green');
            await changeBackgroundColor(rightMinValue, 'green');
            if (leftMinValue <= rightMinValue) {
                minimumValue = leftMinValue;
                leftArrIndex++;
            } else {
                minimumValue = rightMinValue;
                rightArrIndex++;
            }
            sortedArray.push(minimumValue);
            await changeBackgroundColor(leftMinValue, window.baseColor);
            await changeBackgroundColor(rightMinValue, window.baseColor);
        }
        if (leftArrIndex < leftSortedArray.length) {
            sortedArray = sortedArray.concat(leftSortedArray.slice(leftArrIndex));
        }
        if (rightArrIndex < rightSortedArray.length) {
            sortedArray = sortedArray.concat(rightSortedArray.slice(rightArrIndex));
        }
        originalArray.splice(startIndex, sortedArray.length, ...sortedArray);
        await renderList(originalArray);
        return sortedArray;
    }
```

4.1.5 QuickSort.js:

```
async function doQuickSort(array) {
    await quickSort(array, 0, array.length - 1);
}
async function quickSort(items, left, right) {
    if (isStopTriggered) return;
    var index;
    if (items.length > 1) {
        index = await partition(items, left, right); //index returned from partition
        if (left < index - 1) { //more elements on the left side of the pivot
            await quickSort(items, left, index - 1);
        }
        if (index < right) { //more elements on the right side of the pivot
            await quickSort(items, index, right);
        }
    }
    return items;
}
async function partition(items, left, right) {
    if (isStopTriggered) return;
    var pivot = await getPivot(items, left, right);
    debugger;
    await changeBackgroundColor(pivot, 'green');
    i = left, //left pointer
        j = right; //right pointer
    while (i <= j) {
        if (isStopTriggered) {
            await endSwappingStep(items[i], items[j]);
            return;
        }
```

```
    await changeBackgroundColor(items[i], 'blue');
    await changeBackgroundColor(items[j], 'red');
    while (items[i] < pivot) {
        await changeBackgroundColor(items[i], window.baseColor);
        i++;
        if (i > j) {
            await changeBackgroundColor(items[j], 'red');
        }
        await changeBackgroundColor(items[i], 'blue');
    }
    while (items[j] > pivot) {
        await changeBackgroundColor(items[j], window.baseColor);
        j--;
        if (j < i) {
            await changeBackgroundColor(items[i], 'blue');
        }
        await changeBackgroundColor(items[j], 'red');
    }
    if (i <= j) {
        await swap(items[i], items[j]);
        swapArrItem(items, i, j); //sawpping two elements

        await endSwappingStep(items[i], items[j]);
        i++;
        j--;
        await changeBackgroundColor(items[i], 'blue');
        await changeBackgroundColor(items[j], 'red');
    }
    await changeBackgroundColor(pivot, 'green');
}
await endSwappingStep(items[i], items[j]);
```

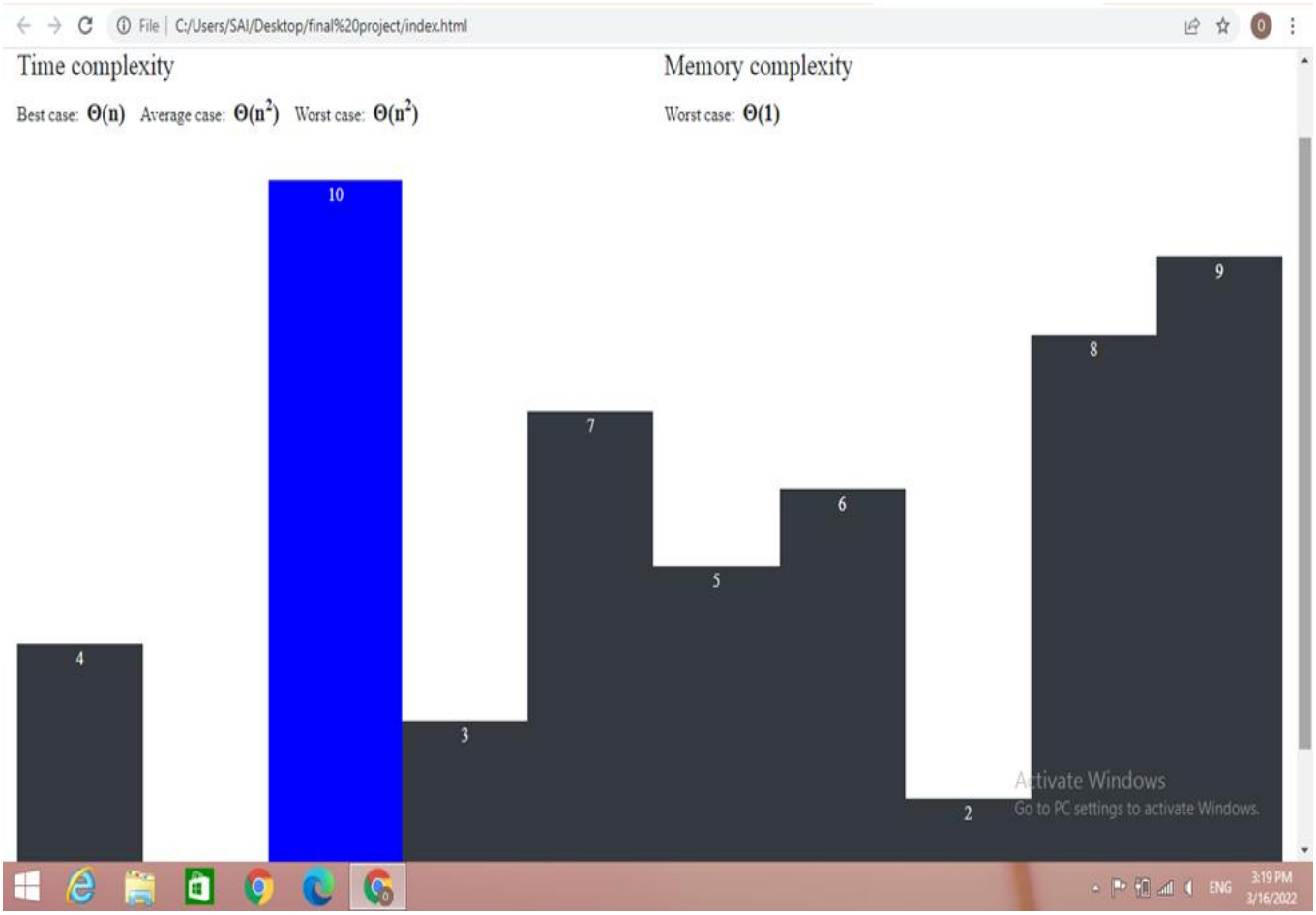
```
    await changeBackgroundColor(pivot, window.baseColor);
    return i;
}
// Bring the median of left, right, middle to right, and take it to be pivot
async function getPivot(items, left, right) {
    let mid = Math.floor((left + right) / 2);
    if (items[mid] < items[left]) {
        await prepareSwappingStep(items[mid], items[left]);
        await swap(items[mid], items[left]);
        swapArrItem(items, mid, left);
        await endSwappingStep(items[mid], items[left]);
    }
    if (items[right] < items[left]) {
        await prepareSwappingStep(items[left], items[right]);
        await swap(items[left], items[right]);
        swapArrItem(items, left, right);
        await endSwappingStep(items[left], items[right]);
    }
    if (items[mid] < items[right]) {
        await prepareSwappingStep(items[mid], items[right]);
        await swap(items[mid], items[right]);
        swapArrItem(items, mid, right);
        await endSwappingStep(items[mid], items[right]);
    }

    return items[right];
}
```

4.1.6 SelectionSort.js:

```
async function doSelectionSort(array) {
    let size = array.length;
    let i, j;
    for (i = 0; i < size - 1; i++) {
        let minIndex = i;
        await changeBackgroundColor(array[minIndex], 'yellow');
        for (j = i + 1; j < size; j++) {
            if (isStopTriggered) return;
            await changeBackgroundColor(array[j], 'red');
            if (array[j] < array[minIndex]) {
                await changeBackgroundColor(array[minIndex], '#343a40');
                minIndex = j;
                await changeBackgroundColor(array[minIndex], 'yellow');
            } else {
                await changeBackgroundColor(array[j], '#343a40');
            }
        }
        if (minIndex != i) {
            await swap(array[minIndex], array[i]);
            swapArrItem(array, minIndex, i);
            await endSwappingStep(array[minIndex], array[i]);
        }
        await changeBackgroundColor(array[minIndex], '#343a40');
    }
}
```

4.2 Screenshots:

4.2.1 Bubble Sort:



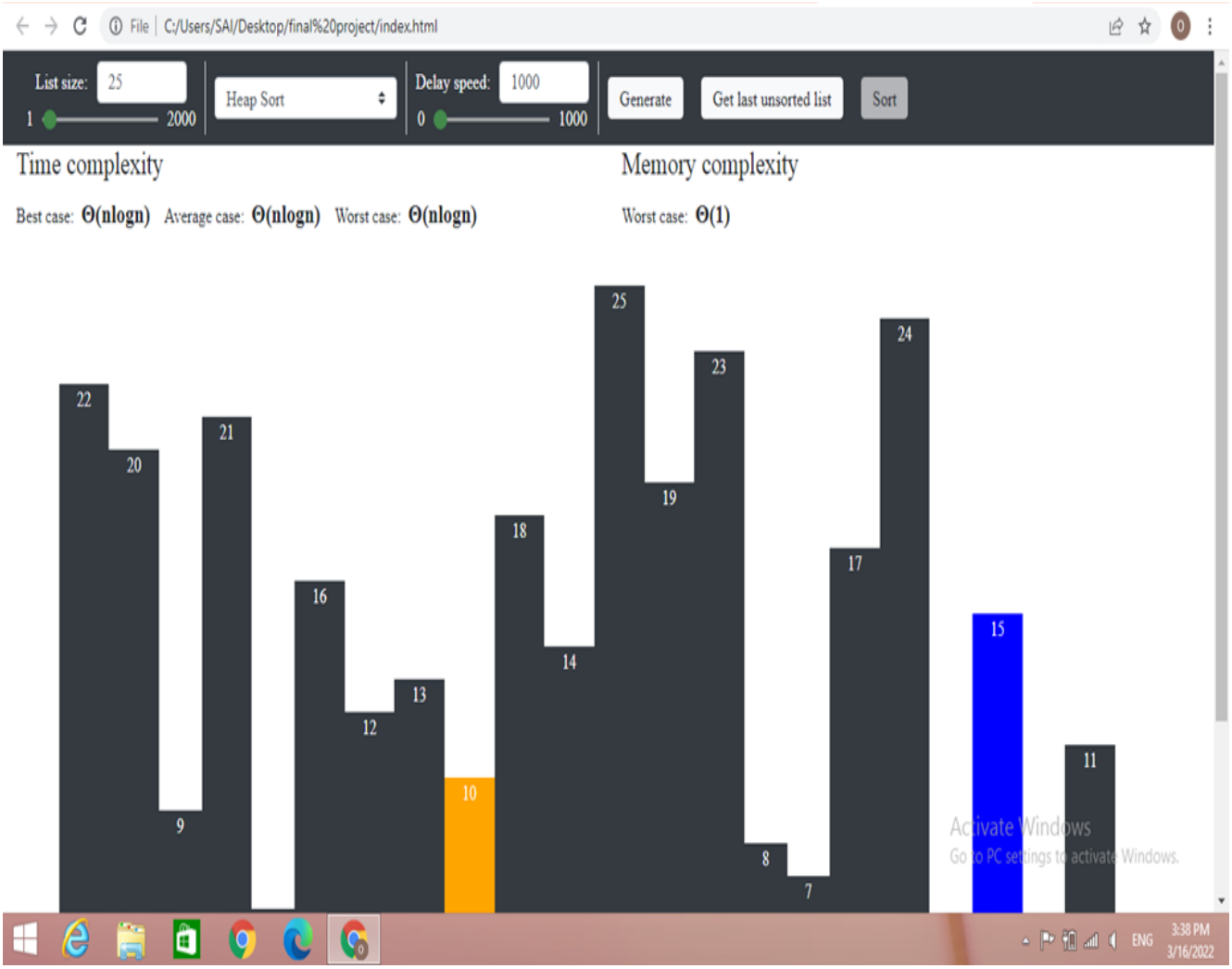
4.2.2 Insertion Sort:

4.2.3 Quick Sort:

4.2.4 Merge Sort:

4.2.5 Selection Sort:

4.2.6 Heap Sort:



5. System Testing

System testing, also referred to as system-level tests or system-integration testing, is the process in which a quality assurance (QA) team evaluates how the various components of an application interact together in the full, integrated system or application. System testing verifies that an application performs tasks as designed. This step, a kind of black box testing, focuses on the functionality of an application. System testing examines every component of an application to make sure that they work as a complete and unified whole. A QA team typically conducts system testing after it checks individual modules with functional or user-story testing and then each component through integration testing.

5.1 Unit Testing:

In unit testing, various modules have been tested individually. This has been done manually to test if the expected result is actually seen on the screen. The following are test cases with the help of which the web application has been tested.

5.1.1 List Size:

This module mainly focuses on taking the array or input size of the array elements. It is tested whether the given number of elements are being taken when generating action is performed.

5.1.2 Select Sorting Algorithm:

This module is like a drop down menu where it has many sorting algorithms to choose and to observe whether the sorting is happening as instructed and chosen here.

5.1.3 Delay Speed:

This module specifies the delay time to sort the elements. It helps in understanding the sorting process easily by delaying the time of its functionality.

5.1.4 Generate:

This module is an element generated module where it creates an array of

elements of given size. This module is tested whether it creates the exact number of elements that the size is specified.

5.1.5 Sort:

This is such a main module in the implementation where it takes all the inputs from array size,delay speed,generates and sorts the elements as per the chosen sorting algorithm and finally deploys the output. This module is tested whether it gives the final sorted array according to the chosen sorting algorithm.

5.1.6 Last Generated Array:

This module is tested whether it produces the last array elements that are generated or not.

5.2. Performance Testing:

Performance testing has been done to measure the responsiveness of the web application to the workload such as increasing users' requests. The performance of PlaySorts web application is very responsive and no faults occurred during accessing and performing operations on the website.

6. Conclusion and Future Enhancement

6.1 Conclusion:

Students work hard and choose rote learning to learn subjects. This is either not a good practice to learn the subject or they cannot remember for a long period of time and not understand. We thought of an idea to present a subject in a visualization mode. Picturization modes will be accepted by the brain more times faster than like reading books. The sorting visualization mode helps students to understand its functionality and its working principle in an easy way so that students can focus on enhancing their learning type and learn the subject rather than byharding the subject for grades or something. We implemented PlaySorts to present the sorting algorithms in a visualization modes so that the sorting algorithms functionality and their working principle will be understand in an easy even for Non-IT students.

6.2 Future Enhancement:

“PlaySorts” can be enhanced in the future by adding more features in it. Only sorting algorithms are included in the present implementation. For further enhancement we can add Data Structures, Searching algorithms and many more to understand their implementation and working principles. Thus, it makes easy to learn any programming language and its fundamentals.

7. REFERENCES