

Sorting Algorithm Visualizer

A Project Report

Submitted by

**ISHAAN KANGRIWALA,
PAARTH KAPASI,
DHRUV PATEL,
NIHAL SHETTY.**

Under the Guidance of
PROF. MOHINI REDDY

*in partial fulfillment for the award of the
degree of*
**BACHELOR OF TECHNOLOGY CSBS
COMPUTER ENGINEERING**

At



**MUKESH PATEL SCHOOL OF TECHNOLOGY
MANAGEMENT AND ENGINEERING**

OCTOBER, 2021.

DECLARATION

We, **Ishaan Kangriwala, Paarth Kapasi, Dhruv Patel and Nihal Shetty**, Roll Nos. **E019, E020, E035 and E050**, B.Tech CSBS (Computer Engineering), V semester understand that plagiarism is defined as anyone or combination of the following:

1. Un-credited verbatim copying of individual sentences, paragraphs, or illustration (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.
2. Un-credited improper paraphrasing of pages paragraphs (changing a few words phrases, or rearranging the original sentence order)
3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did wrote what. (Source: IEEE, The institute, Dec. 2004)
4. I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.
5. I affirm that no portion of my work can be considered as plagiarism, and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the seminar/ project report may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Signature of the Student:

Name: Ishaan Kangriwala

Paarth Kapasi

Dhruv Patel

Nihal Shetty

Roll No. E019

E020

E035

E050

Place: Mumbai

Date: 21/10/21

CERTIFICATE

This is to certify that the project entitled “**Sorting Algorithm Visualizer**” is the bonafide work carried out by **Ishaan Kangriwala, Paarth Kapasi, Dhruv Patel and Nihal Shetty**, of B.Tech CSBS (Computer Engineering), MPSTME (NMIMS), Mumbai, during the V semester of the academic year 2021-2022, in partial fulfillment of the requirements for the award of the Degree of Bachelors of Engineering as per the norms prescribed by NMIMS. The project work has been assessed and found to be satisfactory.

Prof. Mohini Reddy

Internal Mentor

Examiner 1

Examiner 2

Director

Table of contents

CHAPTER NO.	TITLE	PAGE NO.
	List of Figures	i
	List of Tables	ii
	Abstract	iii
1.	INTRODUCTION	1
	1.1 Project Overview	1
	1.2 Impact	2
	1.3 Hardware Specification	2
	1.4 Software Specification	3
2.	REVIEW OF LITERATURE	6
3.	ANALYSIS & DESIGN	9
4.	METHODS IMPLEMENTED	22
5.	CONCLUSION & FUTURE SCOPE	24
	REFERENCES	25

List of Figures

CHAPTER NO.	TITLE	PAGE NO.
3.	DESIGN AND IMPLEMENTATION	
	Fig3.1 System Architecture	9
	Fig3.2 Use Case Diagram	10
	Fig3.3 Sequence Diagram	13
	Fig3.4 Class Diagram	14
	Fig3.5 Activity Diagram	15
	Fig3.6 Login Page	19
	Fig3.7 Register/Sign up Page	19
	Fig3.8 Incorrect Login Details	20
	Fig3.9 Incorrect Email format	20
	Fig3.10 Homepage	20
	Fig3.11 UI while Sorting in progress	21
	Fig3.12 UI After Sorting	21
	Fig3.13 Stored Images Page	21

List of Tables

CHAPTER NO.	TITLE	PAGE NO.
2.	REVIEW OF LITERATURE	
	Table 2.1 Comparison with existing approaches	6

Abstract

One of the most common challenges faced by students as well as teachers these days is to explain the concept of certain algorithms mathematically. Multiple studies indicate that many students had either incomplete or faulty representations of the algorithmic concepts, which seem to be connected to their misconceptions about the programming variable concept. Fear of programming and the abstract nature of programming concepts are some of the reasons which make understanding algorithms more difficult. In this project we implement a sorting algorithm visualizer that helps us understand and conceptualize the algorithms with ease. This visualizer works with multiple algorithms like QuickSort, MergeSort, InsertionSort, BubbleSort and SelectionSort. It includes dynamic features, based on animation techniques, aiming at illustrating the behavior of basic algorithms and fostering students' experimentation and algorithmic knowledge by letting them interact with the said algorithms.

Keywords: *Algorithm, Visualization, Dynamic, Animation, Sorting, Experimentation, Misconceptions, Conceptualize*

Chapter 1

Introduction

1.1 Project Overview

Algorithmic thinking and problem-solving skills play a major role in studying computer science. Students usually need to become familiar with a lot of different algorithms and data structures while studying the course. The skill to devise an algorithm for a given problem statement is one of the most important and strenuous tasks in computer science education. However, various surveys show that the beginners in the field face a lot of difficulties in using abstract programming concepts like data structures and lack the necessary critical thinking skills required to envision an algorithm as a single entity, to understand its main parts and the relations among them.

We have always studied algorithms with a mathematical and empirical approach, but there is yet another way to study algorithms, called algorithm visualization and it can be defined as the use of pictorial representations to convey some useful information about the working of an algorithm. So, why visualize algorithms? Why visualize anything? For leveraging the human visual system to improve understanding or more simply, to use vision to think. It can be a visual illustration of an algorithm's operation, of its performance or of its execution speed. Visuals help learners grasp ideas and concepts easily by stimulating imagination and affecting their cognitive capabilities. Quite often, the students fail to understand the fundamental idea of a particular algorithm maybe because they are unable to visualize how they work. Various studies report that seventy-five percent of all the information processed by the brain is derived from visual formats. Furthermore, visual information is mapped better in students' minds, so, the most important thing to understand about these algorithms is to first visualize them.

1.1.1 Formatting Guidelines

The document is prepared using Microsoft Word 2021 and has used the font type 'Times New Roman'. The fixed font size that has been used to type this document is 12pt. with 1.5 line spacing. It has used the bold property to set the headings of the document. All pages except the cover page are numbered, the numbers appear on the lower right-hand corner of the page. Every image and data table are numbered and referred to in the main text.

Use case scenario is written according to Alistair Cockburn's template. Standard IEEE template is the template used to organize the appearance of the document and its flow.

1.2 Impact

Technology is an essential part of an evolving world. This means that computer programming is an important domain for our future. Computer Science graduates can help create this future by automating processes, collecting data, analyzing information, and sharing knowledge to continuously innovate and improve upon existing processes. But in order to do so, their fundamentals of programming have to be very strong. This also means that the students must be well versed with the key concepts of Data Structures and Algorithms. One of the major issues faced by the students is comprehending the complex codes and logic behind the algorithms by merely reading them or through diagrams. A new tool for teaching these algorithms such as Merge Sort, Quick Sort, Selection Sort etc., through visualization needs to be implemented. We aim to do this through our project i.e., Algorithm Visualizer. This will help students and teachers alike to understand these concepts and be prepared for more complex programming concepts.

1.3 Hardware Specifications

The Bank Loan Management System consists of two components namely Frontend and Backend. The Frontend part deals with the Graphical user interface and the Backend part deals with the connection to the database and storage and retrieval of data from the database. The hardware interface for the project should be a Personal Computer/Laptop with a Windows 10 Operating System. There is no specific hardware component requirements since our project is not a memory intensive or high storage consuming

in nature. The purpose of this PC/Laptop is to provide information of the data entered by the user.

The minimum system requirements needed for the software to run are as follows:

- Processor: 1 gigahertz (GHz) or faster processor
- Hard Disk Space (HDD/SSD): 500 MB
- RAM: 1GB and above
- Display: Minimum 800 x 600 Resolution
- Graphics: 512MB with DirectX 9.0 and later versions

Display Unit:

A Display unit is necessary for the input and output operations. It is used to display greetings to welcome the user, help him/her with the interface, display information about the software and to accept the input and display the output that has been obtained based on it. It can also display video/audio media as a form of entertainment for the user.

Printer (Optional):

The Printer can print the transcripts of the user interaction and final output that was obtained in a standard format.

1.4 Software Used

1.4.1 Visual Studio Code

Visual Studio Code is an Integrated Development Environment (IDE) 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. 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 several

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. The following were implemented using this platform –

- 1) Backend Development: Backend Development involves everything required for the following:
 - i) Login and Registration of the User: We implemented a login/registration portal for the user.
 - ii) Algorithms: We implemented the various sorting algorithms using JavaScript on VS Code.
 - iii) Visualizations: The array elements are visualized based on the sorting algorithm selected using a program written on JavaScript on VS Code.
 - iv) Upload and Fetch Data from the Database and passing it into the templates for visual purpose like creating graphs, tables, etc.
- 2) Frontend Development: Frontend Development involves the code for creating the User Interface. The languages that we used for creating it are HTML and CSS. We created the register and login pages, home page, etc. with the help of the above-mentioned languages. The primary goal of frontend development is to make the website responsive and user-friendly with the help of a simple yet elegant UI.

1.4.2 XAMPP Server

XAMPP is a free and open-source cross-platform web server solution stack package. XAMPP is an abbreviation for cross-platform, Apache, MySQL, PHP and Perl, and it allows you to build your site offline, on a local web server on your computer. We have used MySQL as our Database; hence we have used XAMPP for MySQL Database management.

In our project we have implemented 1 Database:

- Register: In this database we have 2 tables- users and tbl_images. The users table stores the email, name, username, ID and md5 encrypted password of the user which will be used for authentication. The tbl_images table is used for storing the media files and contains the ID and BLOB data type for storing images.

1.4.3 PHPMYAdmin

PHPMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the Web. phpMyAdmin supports a wide range of operations on MySQL and MariaDB. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc.) can be performed via the user interface, while you still have the ability to directly execute any SQL statement.

Chapter 2

Review of Literature

For the literature review of this project, we reviewed papers which have studied the problems faced by students and any person in general who are trying to study/understand the said algorithms. The solutions listed and conclusions of these papers were then analyzed and studied about in detail. Based on their approach and their drawbacks, we then proposed our own solution to the problem of understanding algorithms without any aid in the further sections.

In the first paper that we reviewed (Secondary education students' difficulties in algorithmic problems with arrays: An analysis using the SOLO taxonomy), It was found that grasping those problems which require to use an array in their solution algorithm were most challenging for the students.

In the second paper (Design and evaluation of a web-based dynamic algorithm visualization environment for novices), the proposed system's visualization is in a list format which itself can be confusing for the learners and therefore, adds little to no benefit in terms of ease of understanding the algorithms for the student. This is quite a significant drawback.

The third paper (Visualizing Sequence of Algorithms for Searching and Sorting) fails to document the images of the proposed system UI which makes it difficult for us as reviewers to gauge the effectiveness of their proposed solution. Also, without the user's opinions on their system, there is no substantial proof to the efficacy of their system.

The last paper which we reviewed (ViSA: Visualization of Sorting Algorithms), does not provide the user with an option to store the results which, could be used in their study notes, assignments, and projects as a visual representation or as a study tool for the future.

Thus, based on the drawbacks and the implementation gaps in the reviewed papers' solutions, we seek to implement a system which considers all these points and provides an optimal solution for the end user.

Table 2.1 Comparison with existing approaches

Paper	Description	Features & Drawbacks
<p>Secondary education students' difficulties in algorithmic problems with arrays: An analysis using the SOLO taxonomy</p>	<ul style="list-style-type: none"> • This paper presents the results of an empirical study on secondary education students' misconceptions and mental representations of the array data structure. It infers that many students under the study had incomplete or faulty representations of the array concepts, which led to difficulties in understanding algorithmic problems • The issue seemed to be connected to their misconceptions about the programming variable concept. 	<ul style="list-style-type: none"> • It was found that students face difficulty in judging the type of problems that require to use an array in their solution algorithm • The results presented in this paper could be of value for the design and the successful implementation of new learning environments that support students' algorithmic thinking, as well as the development of appropriate mental models about basic programming entities.
<p>Design and evaluation of a web-based dynamic algorithm visualization environment for novices</p>	<ul style="list-style-type: none"> • This paper presents DAVE, a web-based dynamic algorithm visualization environment designed to support students' learning about basic algorithms. • DAVE facilitates students' experimentation with array algorithms. 	<ul style="list-style-type: none"> • The results obtained from an evaluation study, provide evidence of the usability of the system and its potential to support students' development of efficient mental models regarding basic array algorithms. In addition, students were positive about the environment, its usability and its usefulness to support their learning. • The main drawback of the proposed system in this paper is that the visualization is in a list format which itself can be confusing.

Visualizing Sequence of Algorithms for Searching and Sorting	<ul style="list-style-type: none"> • In this paper sequence of execution of algorithms are explained visually in an interactive manner. • It helps to realize the fundamental concept of algorithms such as searching and sorting method in a simple manner. 	<ul style="list-style-type: none"> • Comparisons of the time complexity of the steps have been carried out. • The system proposed in the paper is interactive for better user experience. • The papers fail to document images of the UI of the proposed system making it difficult for the reviewers to gauge the effectiveness of the system • The papers make the claim that the application improves the perspective of algorithms but fails to incorporate the users' feedbacks to support the same.
ViSA: Visualization of Sorting Algorithms	<ul style="list-style-type: none"> • This paper describes ViSA, a tool for visualization of sorting algorithms. ViSA 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 implication and educational benefits are presented and discussed. 	<ul style="list-style-type: none"> • Visualization of sorting algorithms (ViSA) offers a full range of functionalities such as data set entry and animation control as well as the explanation and detailed algorithm analysis. • Using ViSA, the learning curve and time spent learning and understanding should be significantly reduced. • Although the paper and the system proposed have been executed well, there is no option to store the results.

Chapter 3

Analysis and Design

3.1 System Flow

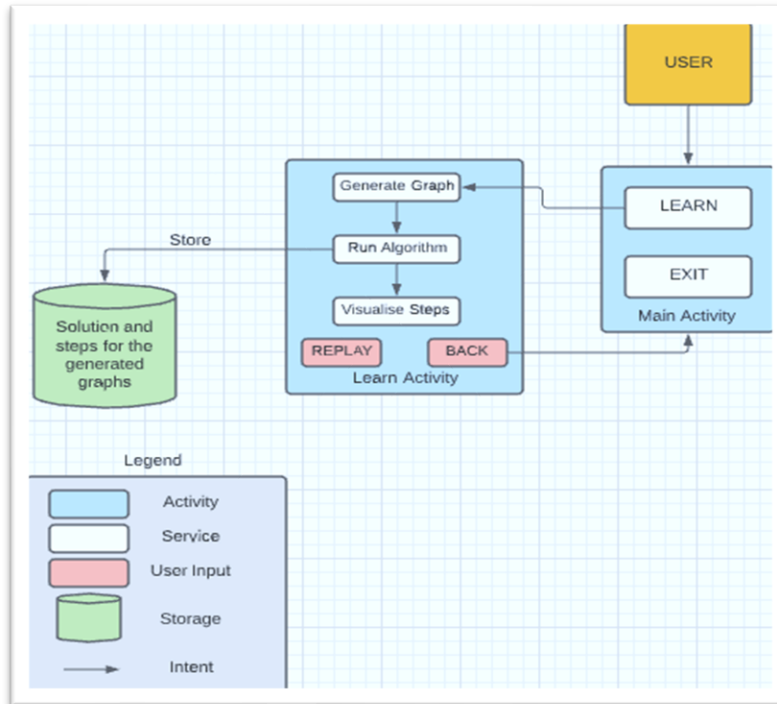


Figure 3.1 System Architecture

This image describes the flow of the system from the user's perspective as well the working of the backend. The user can select to learn algorithms by viewing the sorting of arrays in real-time by following the steps below:

1. The user first selects the size of the array.
2. Next the speed of sorting needs to be set.
3. The user can then select an algorithm from the given choices.
4. The website then runs the selected algorithm and generates a graph for the same.
5. Then the user has an option to store the output, which can be retrieved later.
6. Finally, the user can explore the other algorithms or chose to logout of the system.

3.2 UML Diagrams

3.2.1 Use Case Diagram

Use case diagrams belong to the category of behavioral diagram of UML diagrams. Use case diagrams aim to present a graphical overview of the functionality provided by the system. It consists of a set of actions (referred to as use cases) that the concerned system can perform, one or more actors, and dependencies among them.

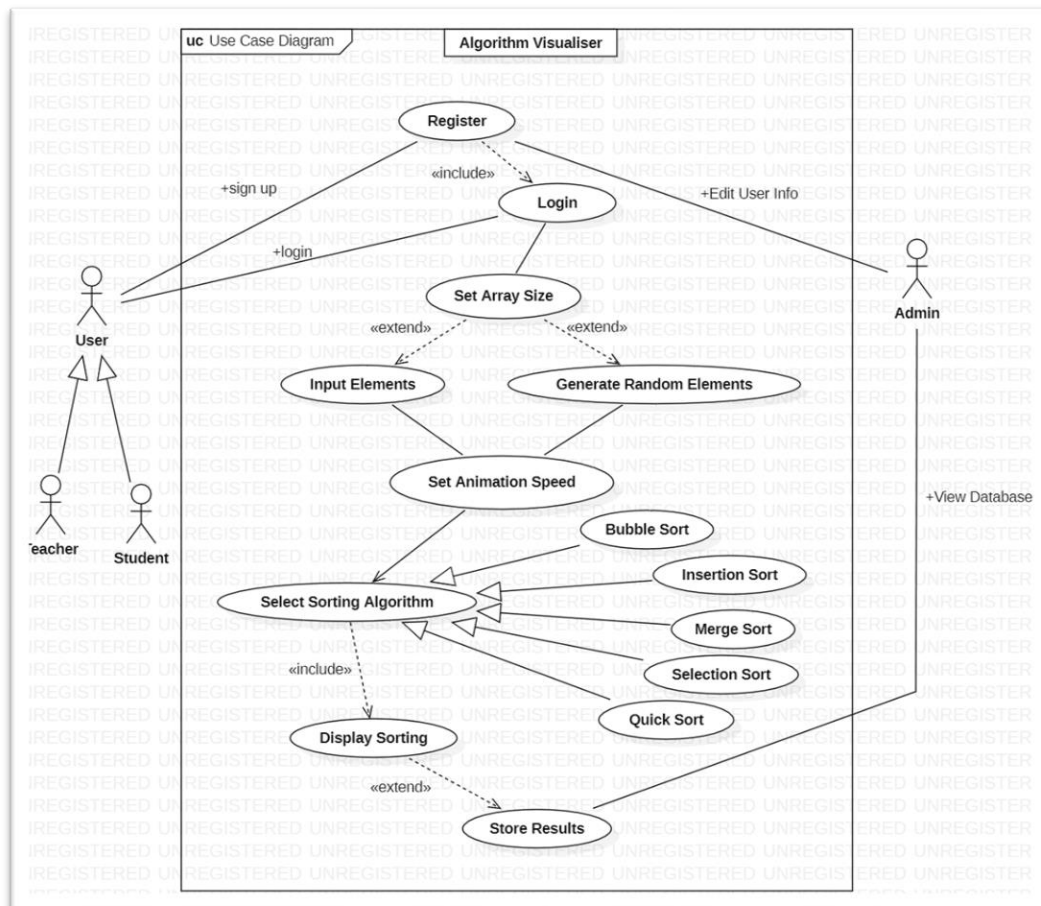


Figure 3.2 Use Case Diagram

Actors:

An actor in use case modeling specifies a role played by a user or any other system that interacts with the subject. An Actor models a type of role played by an entity that interacts with the subject but is not with the subject in the square box.

In our case, the actors are:

- Admin/System
- User (Involves two types):
 - Student
 - Teacher

Use cases:

Use cases are the functionalities involved in the project.

The use cases involved in our project are:

- Admin:
 - Store Results: Allows the admin to access and retrieve stored results by users.
 - Register: Allows the admin to verify user credentials.
- User:
 - Register: The user can register as a teacher or a student using this function.
 - Login: Allows the user to login.
 - Set Array Size: Using this function the user can vary the size of the input array.
 - Input Elements: Custom array could be passed as an input using this functionality.
 - Generate Random Elements: Random numbers are generated and passed as input with this function.
 - Set Animation Speed: This animation speed could be varied using the slider.
 - Select Sorting Algorithm: Any of the given algorithms can be selected to sort the input array.
 - Selection Sort
 - Insertion Sort

- Merge Sort
- Bubble Sort
- Quick Sort
- Store Results: The sorted array can be stored in the database using this function.
- Logout: The user can exit the application and return to the login page.

3.2.2 Sequence Diagram

Sequence diagrams are a graphical way to illustrate a scenario:

- They are called sequence diagrams because they show the sequence of message passing between objects.
- Another big advantage of these diagrams is that they show when the objects are created and when they are destructed. They also show whether messages are synchronous or asynchronous

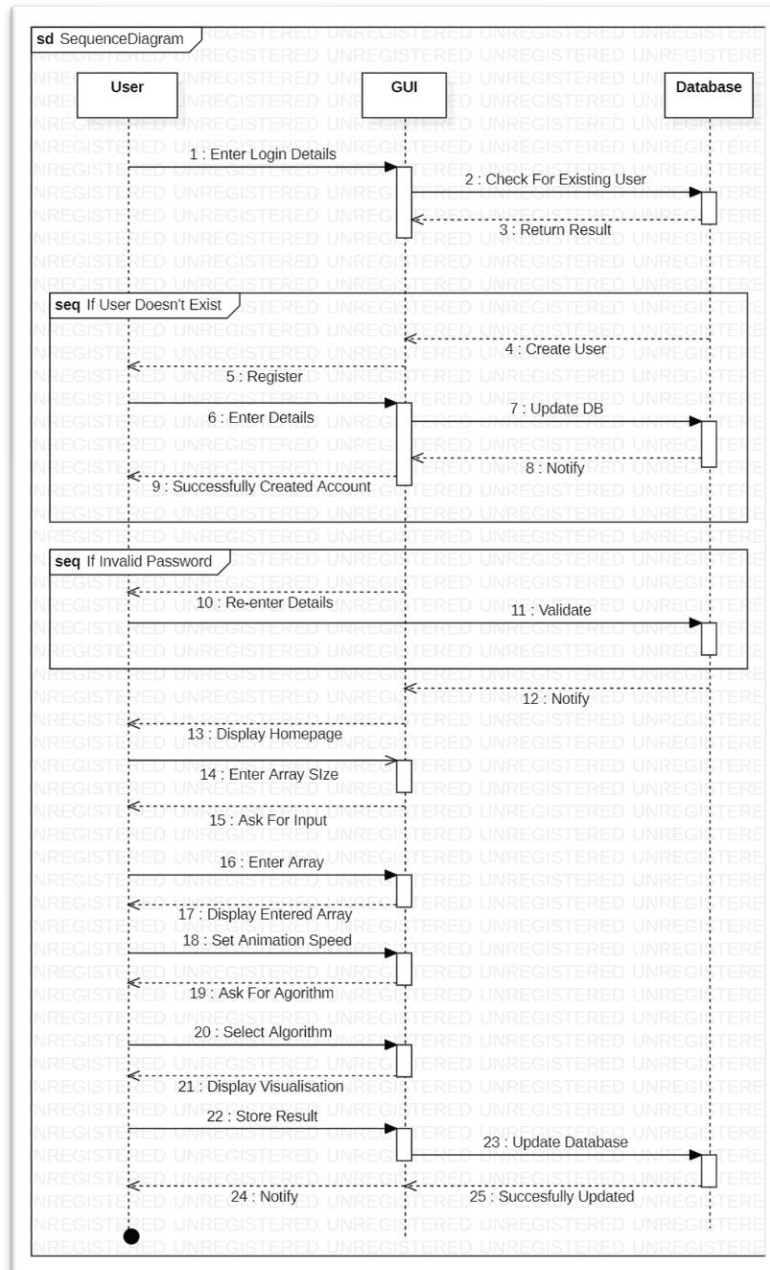


Figure 3.3 Sequence Diagram

Objects:

- Entity objects: Teacher, Student.
- Boundary objects: User Interface, Database.
- Control objects: None.

3.2.3 Class Diagram

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. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints.

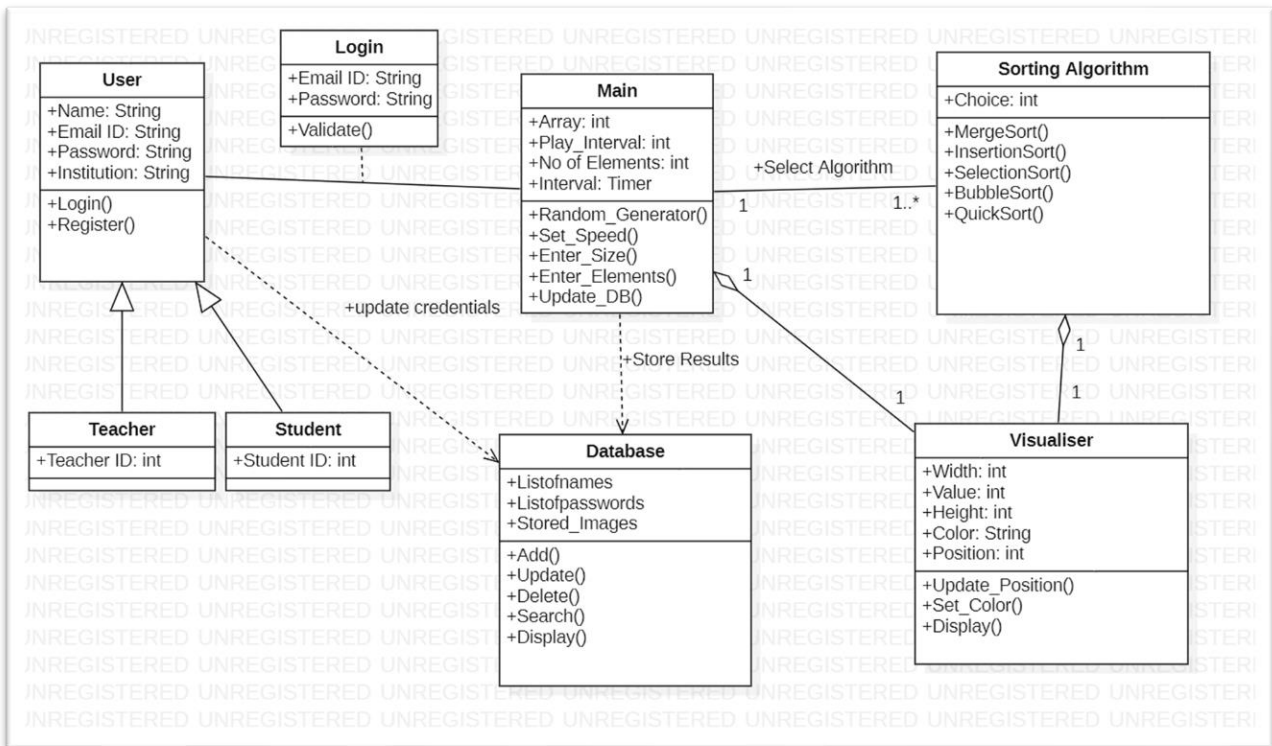


Figure 3.4 Class Diagram

Classes:

- Entity Classes: User
- Boundary Classes: Database
- Control Classes: Main

3.2.4 Activity Diagram

An activity diagram is a behavioral diagram i.e., it depicts the behavior of a system. It portrays the control flow from a start point to a finish point showing the various decision paths that exist while the activity is being executed.

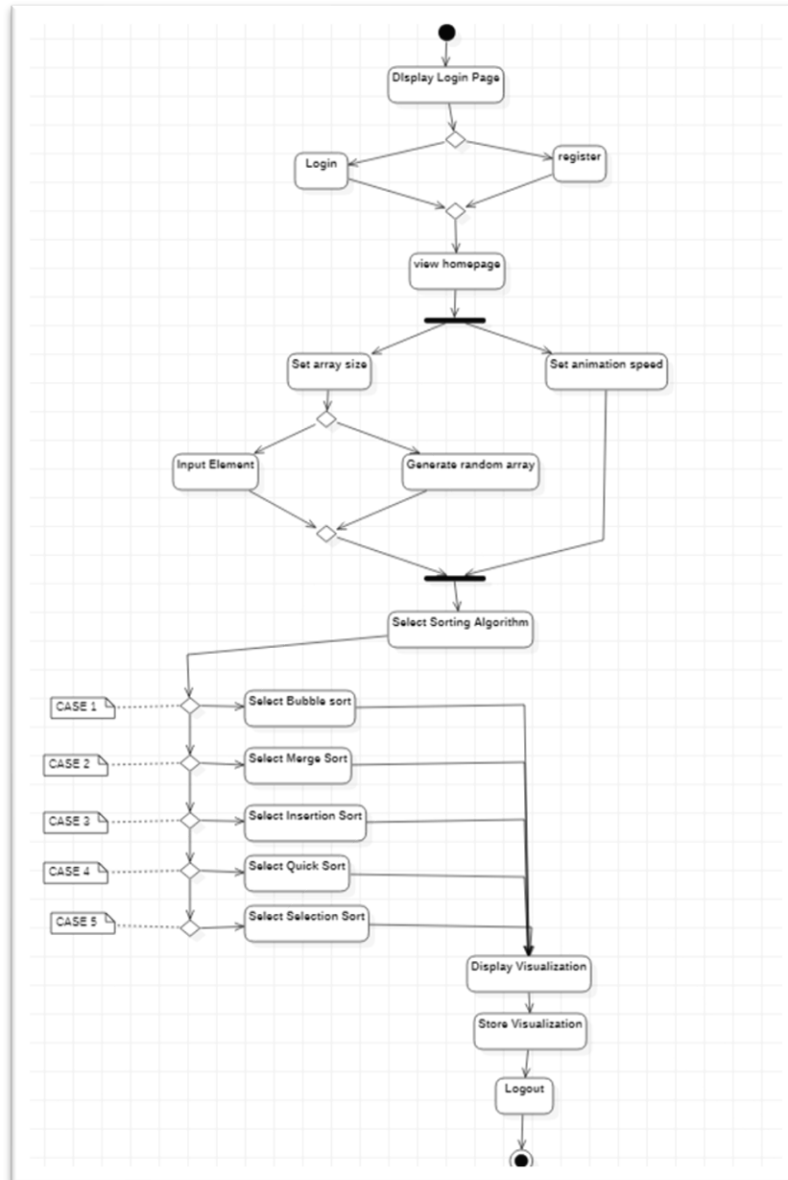


Figure 3.5 Activity Diagram

3.3 Algorithms

For our Algorithm Visualizer project, we have shortlisted five sorting algorithms to be implemented and visualized, which include the following:

3.3.1 Selection Sort

Selection sort is a simple comparison-based sorting algorithm. The idea behind this algorithm is pretty simple. We divide the array into two sub parts: sorted and unsorted wherein the left part is a sorted subarray and the right part is an unsorted subarray. Initially, the sorted subarray is empty and the unsorted array is the complete original array. We perform the following steps until the unsorted subarray becomes empty:

- a) Pick the minimum element from the unsorted subarray.
- b) Swap it with the leftmost element of the unsorted subarray.
- c) Now the leftmost element of unsorted subarray becomes a part (rightmost) of sorted subarray and will not be a part of unsorted subarray.

3.3.2 Insertion Sort

Insertion sort is the sorting mechanism where the sorted array is built having one item at a time. The array elements are compared with each other sequentially and then arranged simultaneously in some particular order. The analogy can be understood from the style we arrange a deck of cards. This sort works on the principle of inserting an element at a particular position, hence the name Insertion Sort.

Insertion Sort works as follows:

- a) The first step involves the comparison of the element in question with its adjacent element.
- b) And if at every comparison reveals that the element in question can be inserted at a particular position, then space is created for it by shifting the other elements one position to the right and inserting the element at the suitable position.
- c) The above procedure is repeated until all the element in the array is at their apt position.

3.2.3 Merge Sort

Merge sort is one of the most efficient sorting algorithms. It works on the principle of Divide and Conquer. Merge sort repeatedly breaks down a list into several sub-lists until each sub-list consists of a single element and merging those sub-lists in a manner that results into a sorted list.

3.2.4 Bubble Sort

Bubble sort, also referred to as comparison sort, is a simple sorting algorithm that repeatedly goes through the list, compares adjacent elements and swaps them if they are in the wrong order. This is the simplest algorithm and inefficient at the same time. Yet, it is very much necessary to learn about it as it represents the basic foundations of sorting.

Following are the steps involved in bubble sort (for sorting a given array in ascending order):

- a) Starting with the first element (index = 0), compare the current element with the next element of the array.
- b) If the current element is greater than the next element of the array, swap them.
- c) If the current element is less than the next element, move to the next element. Repeat Step 1.

3.2.5 Quick Sort

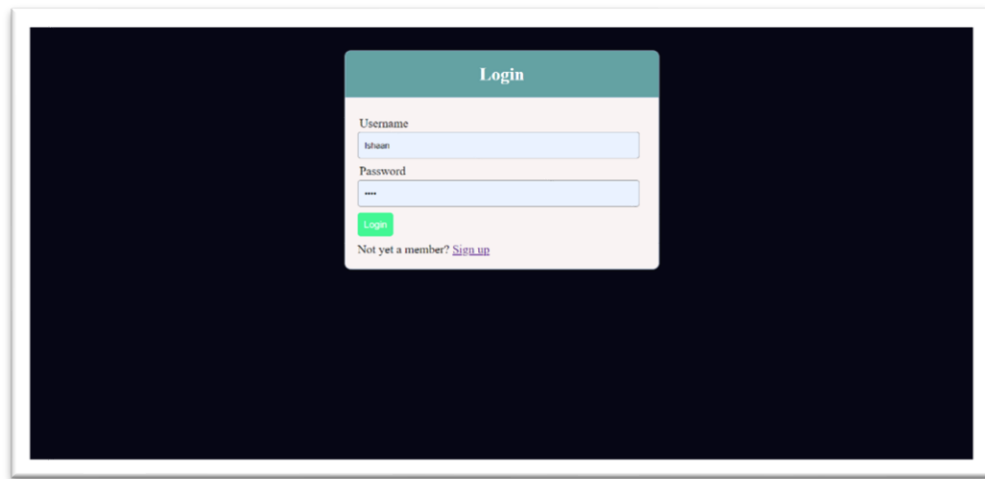
Quick Sort is one of the most efficient sorting algorithms and is based on the splitting of an array (partition) into smaller ones and swapping (exchange) based on the comparison with 'pivot' element selected. Like Merge sort, Quick sort also falls into the category of divide and conquer approach of problem-solving methodology.

Following are the steps involved in quick sort algorithm:

- a) After selecting an element as pivot, which is the last index of the array in our case, we divide the array for the first time.

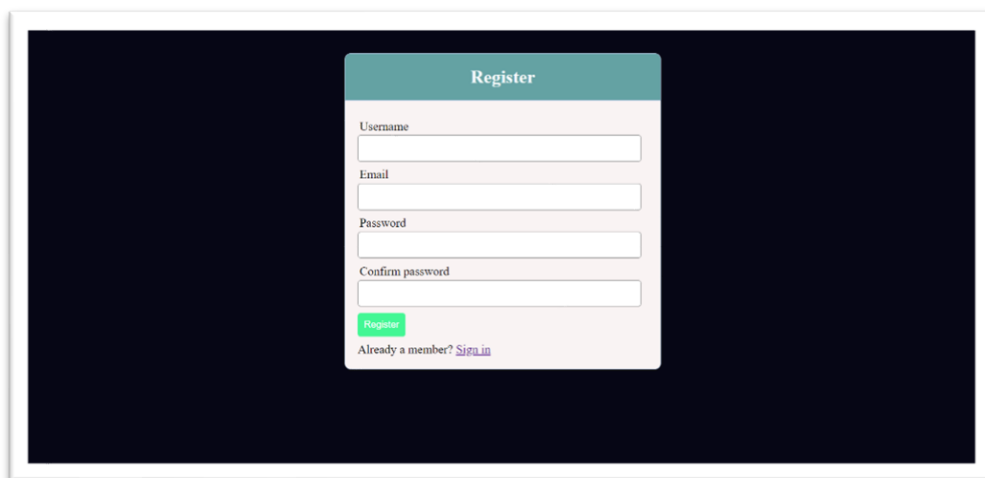
- b) In quick sort, we call this partitioning. It is not simple breaking down of array into 2 subarrays, but in case of partitioning, the array elements are so positioned that all the elements smaller than the pivot will be on the left side of the pivot and all the elements greater than the pivot will be on the right side of it.
- c) And the pivot element will be at its final sorted position.
- d) The elements to the left and right, may not be sorted.
- e) Then we pick subarrays, elements on the left of pivot and elements on the right of pivot, and we perform partitioning on them by choosing a pivot in the subarrays.

3.8 UI DESIGN



The image shows a login page with a dark blue background. A light blue rectangular box is centered on the page, containing the login form. The box has a teal header with the word "Login" in white. Below the header, there are two input fields: "Username" with the text "lham" and "Password" with four asterisks. A green "Login" button is positioned below the password field. At the bottom of the box, there is a link that says "Not yet a member? [Sign up](#)".

Figure 3.6 Login Page



The image shows a register/sign up page with a dark blue background. A light blue rectangular box is centered on the page, containing the registration form. The box has a teal header with the word "Register" in white. Below the header, there are four input fields: "Username", "Email", "Password", and "Confirm password". A green "Register" button is positioned below the "Confirm password" field. At the bottom of the box, there is a link that says "Already a member? [Sign in](#)".

Figure 3.7 Register/Sign Up Page

Login

Wrong username/password combination

Username
Ishaan

Password

Login

Not yet a member? [Sign up](#)

Figure 3.8 Incorrect Login Details

Register

Username
Paarth

Email
Pk123

Please include an '@' in the email address. 'Pk123' is missing an '@'.

Confirm password

Register

Already a member? [Sign in](#)

Figure 3.9 Incorrect Email format

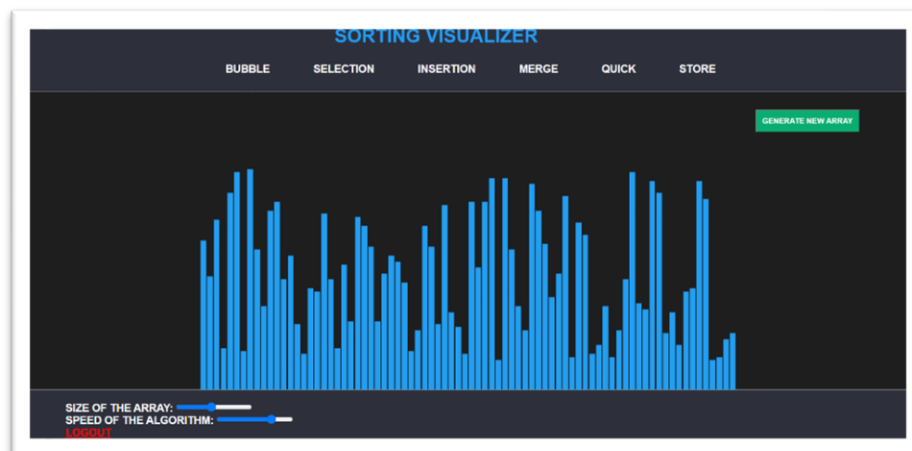


Figure 3.10 Homepage

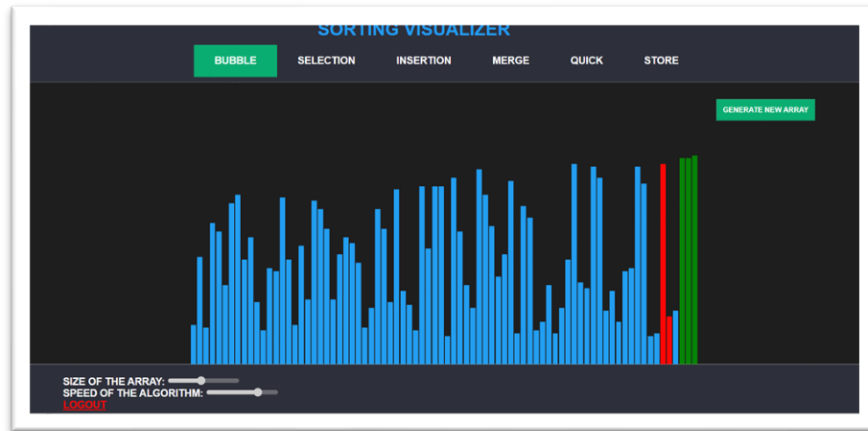


Figure 3.11 UI while Sorting in progress

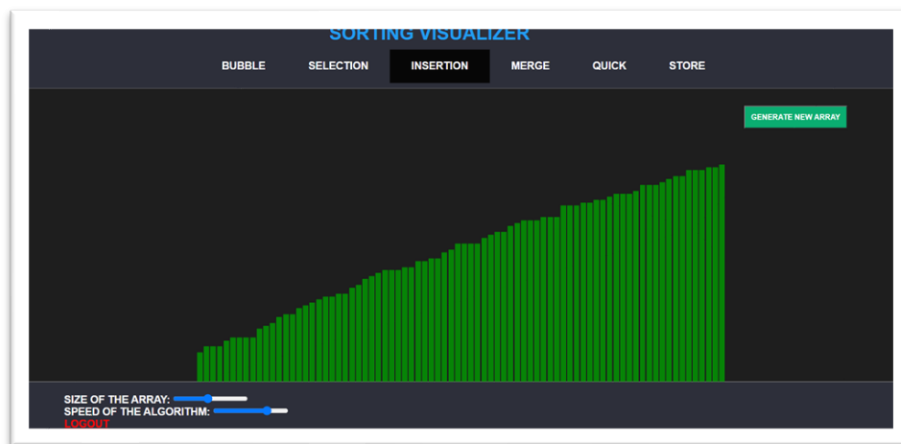


Figure 3.12 UI After Sorting

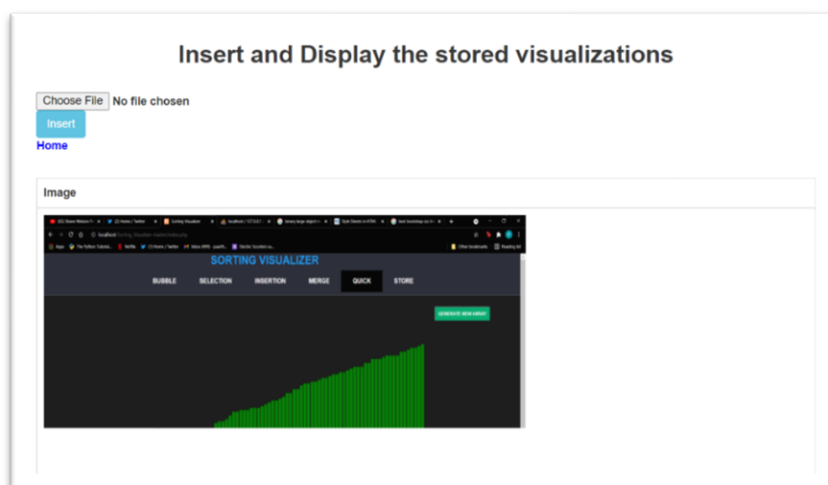


Figure 3.13 Stored Images Page

Chapter 4

Methods Implemented

- **div_update ():** This function is used to update the color, height and width of the individual array elements (divs) during the visualization procedure. The three main processes- iteration through elements, swapping and update sorted part of array, are colored yellow, red and green respectively. It makes use of this formula to do so:

```
cont. style= " margin:0% " + margin_size + "%; width:" + (100/array_size-  
(2*margin_size)) + "%; height:" + height + "%; background-color:" + color + ";;
```

- **generate_array ():** This function is used to assign random values to the array elements based on the size chosen by the user using the formula:

```
div_sizes[i]= Math.floor(Math.random() * 0.5*(inp_as.max - inp_as.min)) + 10;
```

This formula basically scales the randomly generated value to a fixed range and then rounds it off to the lower bound before assigning to the element.

- **update_array_size ():** This function invokes the generate array () function to assign the respective array elements their values. It is invoked by the users pressing of the “generate array” button on the webpage.
- **vis_speed ():** This function is used to assign a speed to the visualizer’s animation process using a switch-case with five cases: 1, 2, 3, 4 and 5 the values of which are obtained from the input slider which we have created on the home page and will assign values 1, 10, 200, 1000 and 10000 respectively.

- **quick_partition ():** This function is used to implement the divide and conquer algorithm which is an essential part of quick sort. The function is called recursively within the main sorting function until the array is sorted.
- **merge_partition ():** This function is used to implement the divide and conquer algorithm which is an essential part of merge sort. The function is called recursively within the main sorting function until the array is sorted.
- **enable_buttons ():** This function is used to unlock the sorting algorithms, array size, array speed and the generate array buttons once the execution of the visualizer is completed. If the user presses on these buttons when the program is being executed, the inputs of the different parameters can be altered which will harm the visualization process which is why locking them is necessary. After the execution is completed, the buttons are unlocked for the next execution.
- **disable_buttons ():** These functions are used to lock the sorting algorithms, array size, array speed and the generate array buttons during the execution of the visualizer. If the user presses on these buttons when the program is being executed, the inputs of the different parameters can be altered which will harm the visualization process which is why locking them is necessary. After the execution is completed, the buttons will be unlocked for the next execution.
- **runalgo ():** This function will initially call the disable_buttons () function to lock the specified buttons and execute the algorithm for which the button was selected by the user. The name of this button is matched to the defined algorithms via a switch case.

Chapter 5

Conclusion And Future Work

As Vrachnos, Euripides & Jimoyiannis, Athanassios suggested in their paper, students usually face difficulties in understanding the concept of algorithms which involve arrays. Our project i.e., An Algorithm visualizer is a proven way to assist these students in understanding the concept of algorithms. Over the course of this project, we learned to implement algorithms visually which required a detailed understanding and examination of the defined algorithms. In our project we have developed a web-based sorting visualizer and for the purpose of implementation, we have selected and implemented five sorting algorithms. The sorting algorithm visualizer has an interactive user interface which allows the users to set the array size or the sorting speed according to their convenience. The app also allows users to store the sorting process in our database for future reference. Using the algorithm visualizer, the learning curve should be reduced significantly. This visualizer aims to reduce the student's fear of coding algorithms and it does so by initially removing the need to code altogether. Once the students are familiar with the algorithms, they can begin to code with greater understanding.

In the future, we could implement this project using ReactJS as it opens up more opportunities to develop complex algorithm visualizers like Prims, Dijkstra's, Pathfinder, etc. We would also like to work upon our UI, so as to make it more interactive as well as appealing to the users. In the future, we would also like to build on this and create an educational software dedicated towards learning algorithms and data structures where in one could learn, experiment and test their knowledge of various algorithms.

References:

- 3 Euripides Vrachnos, Athanassios Jimoyiannis, Design and Evaluation of a Web-based Dynamic Algorithm Visualization Environment for Novices, *Procedia Computer Science*, Volume 27, 2014, Pages 229-239, ISSN 1877-0509, <https://doi.org/10.1016/j.procs.2014.02.026>.
- 4 Reif and T. Orehovacki, "ViSA: Visualization of sorting algorithms," 2012 Proceedings of the 35th International Convention MIPRO, 2012, pp. 1146-1151.
- 5 Ashwani Kumar Singh and Danish Jamal and Pranjal Aggarwal, Algorithm Visualizer, EasyChair Preprint no. 5385, EasyChair, 2021.
- 6 B. R., R. V. and T. S., "Visualizing Sequence of Algorithms for Searching and Sorting," 2009 International Conference on Advances in Recent Technologies in Communication and Computing, 2009, pp. 647-649, doi: 10.1109/ARTCom.2009.20.
- 7 Faria, Brian, "Visualizing Sorting Algorithms" (2017). Honors Projects Overview. 127. https://digitalcommons.ric.edu/honors_projects/127.
- 8 Saraiya, Purvi & Shaffer, Clifford & Mccrickard, D. & North, Chris. (2003). Effective Features of Algorithm Visualizations. Proceedings of the SIGCSE Technical Symposium on Computer Science Education. 36. 10.1145/1028174.971432.
- 9 Hundhausen, Christopher & Douglas, Sarah & Stasko, John. (2002). A Meta-Study of Algorithm Visualization Effectiveness. *Journal of Visual Languages & Computing*. 13. 259-290. 10.1006/jvlc.2002.0237.
- 10 <https://bost.ocks.org/mike/algorithms/>
- 11 <https://visualgo.net/en>
- 12 <https://www.interviewbit.com/tutorial/selection-sort/>
- 13 <https://www.interviewbit.com/tutorial/merge-sort/>
- 14 <https://www.interviewbit.com/tutorial/quick-sort/>
- 15 <https://www.interviewbit.com/tutorial/insertion-sort/>
- 16 <https://www.interviewbit.com/tutorial/bubble-sort/>
- 17 <https://www.gcu.edu/blog/engineering-technology/computer-programming-importance>
- 18 <https://www.phpmyadmin.net/>