**PROJECT REPORT**

**ON**

**ALGORITHM VISUALIZER**

**Project-II**



Department of Computer Science and Engineering

**CHANDIGARH ENGINEERING COLLEGE JHANJERI, MOHALI**

**In partial fulfillment of the requirements for the award of the Degree of**

**Bachelor of Technology in Computer Science & Engineering**

**SUBMITTED BY: Under the Guidance of**

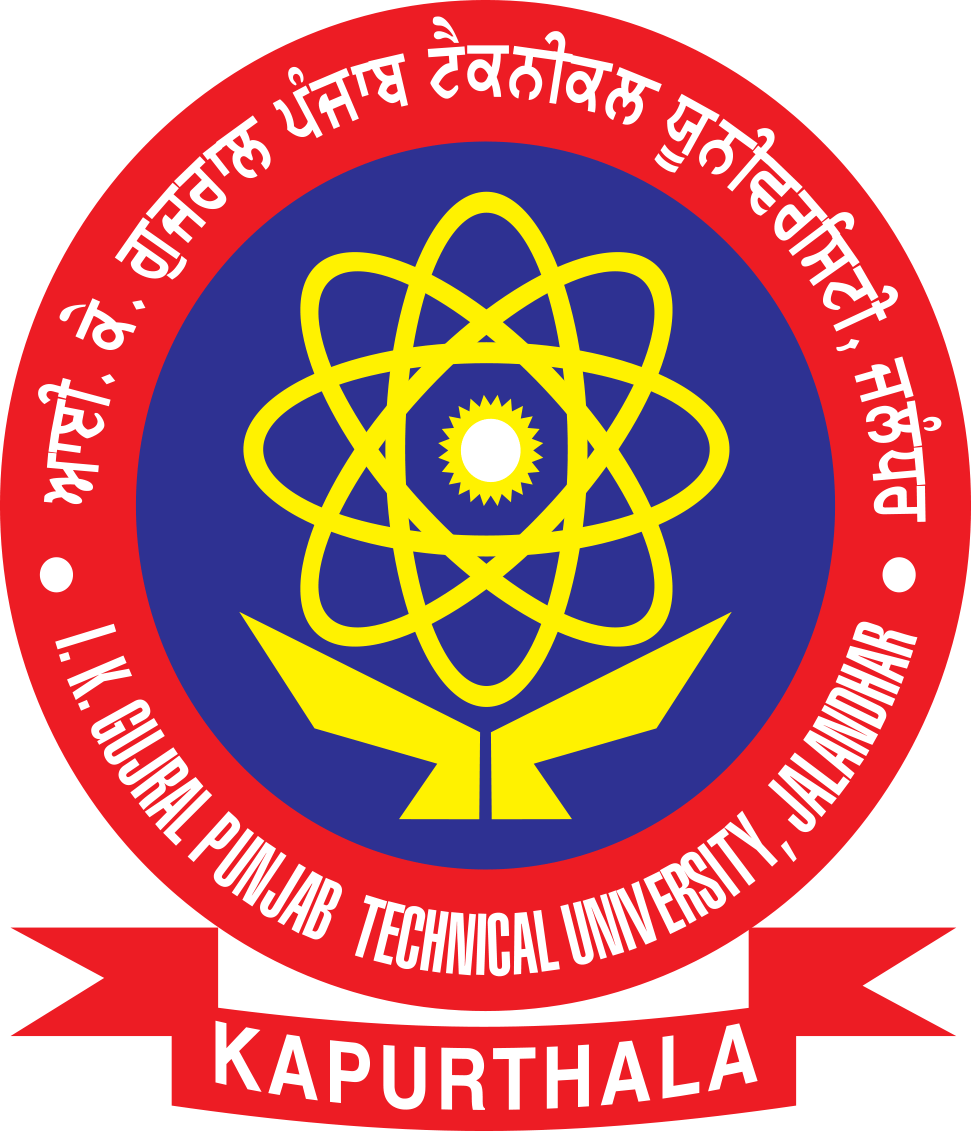
Name Name of Mentor

Sharanjeet Singh, Komal, Aryan Mehta Mrs. Pooja Verma

Rollo Designation of mentor

1915122, 1915079, 1915050

DECEMBER, 2022



**Affiliated to I.K Gujral Punjab Technical University, Jalandh****ar**

**(Batch: 2019-2023)**

**DECLARATION**

I, Sharanjeet Singh with my team members Komal and Aryan Mehta hereby declare that the report of the project entitled “ALGORITHM VISUALIZER” has not presented as a part of any other academic work to get my degree or certificate except Chandigarh Engineering College Jhanjeri, Mohali, affiliated to I.K. Gujral Punjab Technical University, Jalandhar, for the fulfillment of the requirements for the degree of B. Tech in Computer Science & Engineering.

(Student Signature with Date) (Mentor Signature with Date)

**SHARANJEET SINGH MRS. POOJA VERMA**

Univ. Roll No **1915122** DESIGNATION, CSE

Semester: SEVENTH

Signature of the Head of Department

(With Stamp)

**Dr. Sachin Yadav**

**DEAN(CSE)**

**DECLARATION**

I, Sharanjeet Singh with my team members Komal and Aryan Mehta hereby declare that the report of the project entitled “ALGORITHM VISUALIZER” has not presented as a part of any other academic work to get my degree or certificate except Chandigarh Engineering College Jhanjeri, Mohali, affiliated to I.K. Gujral Punjab Technical University, Jalandhar, for the fulfillment of the requirements for the degree of B. Tech in Computer Science & Engineering.

(Student Signature with Date) (Mentor Signature with Date)

**ARYAN MEHTA MRS. POOJA VERMA**

Univ. Roll No **1915050** DESIGNATION, CSE

Semester: SEVENTH

Signature of the Head of Department

(With Stamp)

**Dr. Sachin Yadav**

**DEAN(CSE)**

**DECLARATION**

I, Sharanjeet Singh with my team members Komal and Aryan Mehta hereby declare that the report of the project entitled “ALGORITHM VISUALIZER” has not presented as a part of any other academic work to get my degree or certificate except Chandigarh Engineering College Jhanjeri, Mohali, affiliated to I.K. Gujral Punjab Technical University, Jalandhar, for the fulfillment of the requirements for the degree of B. Tech in Computer Science & Engineering.

(Student Signature with Date) (Mentor Signature with Date)

**KOMAL MRS. POOJA VERMA**

Univ. Roll No **1915079** DESIGNATION, CSE

Semester: SEVENTH

Signature of the Head of Department

(With Stamp)

**Dr. Sachin Yadav**

**DEAN(CSE)**

**ACKNOWLEDGEMENT**

It gives me great pleasure to deliver this report on the Project-II ,I worked on for my B.Tech in Computer Science & Engineering final year, which was titled "ALGORITHM VISUALIZER“. I am grateful to my university for presenting me with such a wonderful and challenging opportunity. I also want to convey my sincere gratitude to all coordinators for their unfailing support and encouragement.

I am extremely thankful to **Dr. Sachin Yadav,** Dean of Computer Science & Engineering at Chandigarh Engineering College Jhanjeri, Mohali (Punjab) for valuable suggestions and heartiest co-opeartion.

I am also grateful to the management of the institute, Dr. Ashwani Sharma, Director Engineering, and Dr. Vinod Kumar, Director Academics, for giving me the chance to acquire the information. I am also appreciative of all of my faculty members, who have instructed me throughout my degree.

(Signature of Student)

**SHARANJEET SINGH**

**ACKNOWLEDGEMENT**

It gives me great pleasure to deliver this report on the Project-II ,I worked on for my B.Tech in Computer Science & Engineering final year, which was titled "ALGORITHM VISUALIZER“. I am grateful to my university for presenting me with such a wonderful and challenging opportunity. I also want to convey my sincere gratitude to all coordinators for their unfailing support and encouragement.

I am extremely thankful to **Dr. Sachin Yadav,** Dean of Computer Science & Engineering at Chandigarh Engineering College Jhanjeri, Mohali (Punjab) for valuable suggestions and heartiest co-opeartion.

I am also grateful to the management of the institute, Dr. Ashwani Sharma, Director Engineering, and Dr. Vinod Kumar, Director Academics, for giving me the chance to acquire the information. I am also appreciative of all of my faculty members, who have instructed me throughout my degree.

(Signature of Student)

**KOMAL**

**ACKNOWLEDGEMENT**

It gives me great pleasure to deliver this report on the Project-II ,I worked on for my B.Tech in Computer Science & Engineering final year, which was titled "ALGORITHM VISUALIZER“. I am grateful to my university for presenting me with such a wonderful and challenging opportunity. I also want to convey my sincere gratitude to all coordinators for their unfailing support and encouragement.

I am extremely thankful to **Dr. Sachin Yadav,** Dean of Computer Science & Engineering at Chandigarh Engineering College Jhanjeri, Mohali (Punjab) for valuable suggestions and heartiest co-opeartion.

I am also grateful to the management of the institute, Dr. Ashwani Sharma, Director Engineering, and Dr. Vinod Kumar, Director Academics, for giving me the chance to acquire the information. I am also appreciative of all of my faculty members, who have instructed me throughout my degree.

(Signature of Student)

**ARYAN MEHTA**

**TABLE OF CONTENTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **PARTICULARS** | | | | **PAGE NO** |
| Title Page | | | | I |
| Declaration by the Candidate | | | | II-IV |
| Acknowledgement | | | | V-VII |
| Table of Contents | | | | VIII – IX |
| Abstract | | | | X |
| List of Figures | | | | XII |
| **CHAPTER 1 INTRODUCTION**  **CHAPTER 2 REVIEW OF LITERATURE** | | | | **1-2**  **3-10** |
| 2.1 | | | Motivation | 3 |
| 2.2 | | | Introduction of Modern Way | 3-4 |
| 2.3 | | | Brown University Algorithm Simulator and Animator | 4 |
| 2.4 | | | Tango or XTango | 5 |
| 2.5 | | | Generalized Algorithm Illustration through Graphical Software | 5-6 |
| 2.6 | | | Dynamic Laboratory | 5-6 |
| 2.7 | | | SWAN | 6 |
| 2.8 | | | Java And Web-based Algorithm Animation | 7 |
| 2.9 | | | Flexible Learning with Artificial Intelligence Repository | 7-8 |
| 2.10 | | | POLKA | 8-9 |
| 2.11 | | | MOCHA | 9 |
| 2.12 | | | Hyper media Algorithm Visualization System | 9-10 |
| **CHAPTER 3 PROBLEM DEFINITION AND OBJECTIVES** | | | | **11-18** |
| 3.1 | Problem Definition | | | 11 |
| 3.2 | Purposed Solution | | | 11-13 |
| 3.3 | Aims and Objectives | | | 13-14 |
| 3.4 | Research and experiences | | | 14-18 |
| **CHAPTER 4 DESIGN AND IMPLEMENTATION** | | | | **19-27** |
| 4.1 | | The User Interface | | 19-21 |
| 4.2 | | System Architecture | | 21-22 |
| 4.3 | | Implementation | | 22-27 |
|  | | 4.3.1 The View | | 23-24 |
|  | | 4.3.2 The Model | | 24-26 |
|  | | 4.3.3 Sample Code | | 26-27 |
| **CHAPTER 5 RESULTS AND DISCUSSIONS** | | | | **28-29** |
| 5.1 | | Survey Results | | 29 |
| **CHAPTER 6 CONCLUSION AND FUTURE SCOPE** | | | | **30-34** |
| 6.1 | | | Conclusions | 30-33 |
| 6.2 | | | Future Scope | 33-34 |
| **REFERENCES** | | | | **35-39** |

**ABSTRACT**

Over the years we've observed that algorithms even tough being a complex subject are the foundation of computational thinking and programming skills of a student. So to ease up the hardships of students this idea of the project was formed. Our application Algorithm Visualizer is both interactive and alluring to students.

It gives the students hands on experience of the algorithms' implementation. It feeds into their imagination to help them get a better understanding while also helping teachers to help make their students understand better. Through this project every student can learn at their own pace with our three speeds of learning: slow, average and fast.

This interface is designed to make one feel fully engaged and concentrated. The concept of Time Complexity has also been introduced to the user through an interactive game. We have made use ofReact.js as framework and JavaScript as primary language for our project. The purpose of this project is to make learning less of a burden and more of an incredible experience which leaves students with the want to learn more.

Algorithm visualization illustrates how algorithms work in a graphical way. It mainly aims to simplify and deepen the understanding of algorithms operation. Within the paper we discuss the possibility of enriching the standard methods of teaching algorithms, with the algorithm visualizations.

As a step in this direction, we introduce the algorithm visualization platform, present our practical experiences and describe possible future directions, based on our experiences and exploration performed by means of a simple questionnaire

**List of Figures**

|  |  |
| --- | --- |
| 1. Main Window of Application | 21 |
| 1. View of the Application | 23 |
| 1. View of the Insertion Sort | 25 |
| 1. Sample code for HTML | 26 |
| 1. Sample code for JavaScript | 27 |

**Chapter-1**

**INTRODUCTION**

When we talk about complex subject topics like Algorithms, it becomes extremely necessary for students to have a strong grip over the topic as it would form the foundation of their computational thinking and programming skills. We had observed that through conventional methods of teaching it becomes a little difficult for students to understand the concept and also for teachers to explain their thoughts. Motivated by the age-old saying, “a picture speaks more than thousand words”, many researchers and educators assume that students would learn an algorithm faster and more thoroughly using algorithm visualization techniques

So, we developed a method of learning through visualization and hand-on experience over different searching and sorting algorithms which is bound to help the students and teachers. Good visualizations bring algorithms to life by graphically representing their various states and animating the transitions between those states, especially dynamic algorithm visualization which shows a continuous, movie-like presentation of an algorithm’s operations.

Visualization allows the human visual system to extend human intellect; we can use it to better understand these important conceptual processes, other things, too. Also, we are well aware of the fact that the more we do things ourselves and engage the more we tend to learn about a particular topic. Thus, engaging in various game like activities can surely help the users get a hold on the topics.

Algorithms and data structures as an essential part of knowledge in a framework of computer science have their stable position in computer science curricula, since every computer scientist and every professional programmer should have the basic knowledge from the area. With the increasing number of students in Central European’s higher education systems in last decades (more concrete numbers and impacts for the case of Slovak one can be found in), introduction of appropriate methods into the process of their education is also required. Our scope here is the higher education in the ﬁeld of computer science. So within the paper, we discuss the extension of standard methods of teaching algorithms, using the whiteboard or slides, with the algorithm visualizations.

According to they can be used to attract students’ attention during the lecture, explain concepts in visual terms, encourage a practical learning process, and facilitate better communication between students and instructors. Interactive algorithm visualizations allow students to experiment and explore the ideas with respect to their individual needs. Extensive studies on algorithm visualization eﬀectiveness are available nowadays, and results are quite encouraging. A systematic meta-study of 24 experimental studies can be found in. Results of empirical study aimed at the determination of factors inﬂuencing the eﬀectiveness of algorithm visualization are published in. Another example is the study with the objective to determine learning advantage of the interactive prediction facility provided by the courseware containing algorithm animations and data structure visualizations. Based on above mentioned reasons, results of studies carried, as well as our own experiences and explorations, we consider algorithm visualization important and perspective area of further research and application of its results in nowadays computer science education.

Except the algorithm visualization, the term software visualization is also often used within the papers published in last years. It usually covers both visualization of algorithms and visualization of data structures, but sometimes also another aspects of software (like its development process) are considered, too. Algorithm visualization, as part of software visualization, could be described as "graphical representation of an algorithm or program that dynamically changes as the algorithm runs". An overview of visualization taxonomies, together with an analysis of factors increasing the eﬀectiveness of software visualization, is summarized in. Even if the beginnings of algorithm visualization date back into 1940’s, the greatest development in the area we could observe within the last 20-30 years. Modern approaches to software visualization were brought in the 1980’s by the introduction of system BALSA (Brown & Sedgewick, Brown University, USA). Some of contemporary solutions include systems like TRAKLA23, ANIMAL4, JAWAA5or Algorithms In Action6. Concise overview of development in the area of software visualization we provided in, so it is not our intention to analyze this topic within the paper.

**Chapter-2**

**REVIEW OF LITERATURE**

A number of AV development programs are already well known in the CS education community. Collaborative visualization has been used in computer science education since the 1980s.

AV software has been used:

* help teachers demonstrate the effectiveness of algorithm in a lecture;
* assist learners as they learn and learn basic algorithms in computer science;
* to help instructors track down bugs in students linked-list programs during office hours
* help learners learn about the basic functionality of non-computer-generated data science laboratory.
  1. **Motivation**

The motivation behind this project is to study how the operations on data structure are performed. So that students can learn various algorithms through animation. To get a clear knowledge about various data structures and their operations on it. It will makes Data structures learning more interesting. The main goal of this project is to implement a system for various sorting algorithm , prime number , binary search game , - for investigation and visualization the best and worst case for every implemented algorithmic rule.

**2.2 Introduction of Modern Way**

Modern ways of seeing software were introduced in the 1980’s with the introduction of the BALSA program (Brown & Sedgewick, Brown University, USA). Since then, hundreds of algorithm images have been used and freely distributed to teachers, and points (or large) of paper have been written on them. It is widely thought that algorithm recognition can provide a more powerful alternative than standing document presentations.

Since the introduction of Java in the mid-1990s, almost all of the algorithm displays and algorithm detection tools have been used in Java. About half of the available views are provided as applets

directly on web pages. However, only a handful of Java applications should be downloaded and opened locally. These numbers are somewhat biased. There is a tendency for us to look for applets, as this turns out to be easy to do. Visuals found directly on web pages will often receive more attention from potential users, as they do not have to go through the extra step of downloading and installing the visual or display program.

Some of contemporary solutions include systems like TRAKLA23, ANIMAL4, JAWAA5 or Algorithms In Action :

**2.3 Brown University Algorithm Simulator and Animator**

Brown University Algorithm Simulator and Animator (BALSA) BALSA [Brown et al., 1985] was one of the first algorithm simulations designed to help students understand computer algorithms. The program has served as an example of the animation of many algorithms that were later developed [Wiggins, 1998]. It was developed in the early 1980s at Brown University to achieve a number of goals. Students could use it to look at the performance of algorithms and thus gain a better understanding of their performance. Students do not have to write code but ask for someone else's code. The teachers would use it to prepare the material for the students. The algorithm designers will use the resources provided by the system to obtain a dynamic image display of their operating systems to fully understand the algorithms. Cartoons using state of-the-art resources provided by BALSA will design and implement programs that will be featured in production. The program was written in C and the PASCAL programs were animated. BALSA has provided users with a few services. Users can control the display features of the system and thus be able to create, delete, resize, move, zoom in, and filter algorithm views. Users are given different views of the algorithm at the same time. The system allowed several algorithms to be used and displayed simultaneously. Users are also able to interact with the animation of the algorithm. For example they can start, stop, slow down, and run the algorithm backwards. After the algorithm worked once, the entire history of the algorithm was saved so that readers could refer to it and restart it. Users can save their window settings and use it to restore algorithm views later. The original version of the system introduced black and white animation.

**2.4 Tango or XTango**

Tango and XTango The animation system of the Xtango algorithm was developed under the direction of Drs. John Stasko at Georgia Tech as a follower of the animation system of the Tango

algorithm. XTango "supports color enhancement, real-time, 2 & 1/2 dimensional, smooth graphics of algorithms and programs" [Wiggins, 1998]. The system uses the transformation paradigm to achieve smooth animation [Stasko, 1992]. XTango is used in the X11 window system. It is distributed through sample animation programs (e.g., matrix replication, Fast Fourier Transform, Bubble Filtering, Two Stacks, AVL Trees) [Wilson et al., 1996]. The system is designed for easy use. It is intended to be useful for those who are not computer-assisted in the use of motion pictures [Wiggins, 1998]. The package can be used in two ways: Users can embed data structures and library animation packages in C-format or any other programming language that can generate a tracking file. The embedded dial system is then integrated into the Xtango and X11 window libraries. Another way to use animation package is to write a command text file read by a downloadable animated system translator

along with the Xtango package. A text file can be created with a text editor or as a result of printed statements in a user simulation program.

* 1. **Generalized Algorithm Illustration through Graphical Software**

Generalized Algorithm Illustration through Graphical Software (GAIGS) GAIGS Algorithm Visualization was established at Lawrence University from 1988-1990. The system does not actually create an algorithm but generates summaries, while the algorithm signs, of data structures in "interesting events" [Naps et al., 1994]. Users can then view these summaries at their own pace. GAIGS introduces a different simultaneous Vol-8 Issue-3 2022 IJARIIE-ISSN(O)-2395-4396 16908 ijariie.com 2039 view of the same data structure. It also allows users to go back to the previous state and replay the sequence of the algorithm abbreviations previously introduced. However, it provides users with a standard version of the program code. Users cannot detect how a particular line of code has an effect on system performance [Wiggins, 1998]. GAIGS provides an intuitive understanding of algorithms for testing without editing them. The computer science department of more than 60 institutions has used the previous version of this program in key subjects such as Introduction to

Computer Science, Software Design Principles, Systems Analysis and Design, as well as Data Structures courses and analytical algorithm.

* 1. **Dynamic Laboratory**

Dynamic Laboratory (DynaLab) DynaLab was established in the early 1990s at Montana State University. The program is designed to "open to students a broader definition of algorithms, a world

of interesting problems, problems solved by algorithms, unresolved problems, unresolved problems and what practical solutions to the relevant problems from the set of solutions provided" and supports "interactive, visual and inspiring and demonstrative demonstrations. laboratory in computer science "[Boroni et al., 1996]. It had X-windows and MS-windows program animators as well as a complete animation program library. he can be move the program to the library and display it in the animator. The animation series consists of highlighting a portion of the system in use, displaying different converted values and displaying input in the system. Cartoons also store and display the total cost of making the program. To understand the confusing or complex sequence of events in the algorithm,

users can undo animation with a certain number of steps and restart it in the usual way forward. DynaLab was developed for use in the field of laboratory settings where students could perform various tests with a given algorithm (e.g., to detect the time complexity of the algorithm). Such a test setting can be easily accomplished because each student can have a copy of DynaLab and a specific algorithm with which to work. No more time is needed to set up these laboratory tests and that is why we can always focus on doing them.

**2.7** **SWAN**

SWAN [Shaffer et al., 1996] was created at Virginia Tech, and can be used to visualize data structures used in the C / C ++ system. All data structures in the system are considered graphs. The graph can be oriented or unrestricted and can be limited to special situations such as trees, columns and columns. The program has three key components: Swan Annotation Interface Library (SAIL), Swan Kernel, and Swan Viewer Interface (SVI). SAIL contains library functions that allow users to create different system views. SVI allows users to test the Swan annotation system. The Swan Kernel is the main module of the system.

**2.8** **Java And Web-based Algorithm Animation**

Java And Web-based Algorithm Animation (JAWAA) JAWAA [Pierson and Rodger, 1998] uses simple command language to create animations of data structures and display them through a web browser. The system is built on JAVA so it can be used on any machine. Animation to be created is written in simple script language. A text file can be easily transcribed into any text editor or can be

generated as an output. Texts have one command or image function per line. The JAWAA applet returns the command file and executes it. The system interprets commands by line and performs

specific functions for each command. JAWAA provides commands that allow users to create and move both archeological objects such as circles, rows, text, rectangles, etc., and data structure objects such as columns, stacks, rows, columns, trees, graphs etc. The interface of the system contains animation. canvas and panel that gives users controls such as start, stop, pause and step animation. To control the animation speed the scroll bar is also provided.

**2.9 Flexible Learning with Artificial Intelligence Repository**

Flexible Learning with Artificial Intelligence Repository (FLAIR) FLAIR [Ingargiola et al., 1994] is a repository of algorithms and teaching materials to be used in Artificial Intelligence courses. Students can learn from these resources, which are laboratory-based learning areas, by experimenting with them. For example, students can use the Search Module to streamline common search algorithms on a city map. Students can choose from a set of algorithms a specific algorithm for survival, various heuristics to implement the algorithm, and start and end cities. They can also adjust animation speed, create new city maps to launch the algorithm, skip through animation or pause. Along with the animation of the search algorithm students can resume detailed animation of basic data structures. To compare the performance of different search algorithms, students can have multiple windows open at the same time, each animating a different algorithm that works on the same problem at the same time. Teachers can use these resources to guide students to work on different algorithms, to analyze how different algorithms work in different contexts, and under what circumstances the given algorithm works better than other algorithms that solve Vol-8 Issue-3 2022 IJARIIE-ISSN(O)-2395-4396 16908 ijariie.com 2040 the same problem. The system operates on SUN SPAR operating stations. Recorded

on Common Lisp (CL) and Common List Object System. The graphical user interface of the system was developed using Garnet System (Garnet was developed at Carnegie Mellon University using X-Windows and has a Lisp based GUI). The program can be installed on any hardware / software that supports CL running Garnet.

**2.10** **POLKA**

A low-key animation focused on the corresponding system (POLKA) The animation system of the POLKA algorithm [Stasko et al., 1993] was created at Georgia Tech, under the direction of Drs.

John Stasko. The program not only allows students to watch the animation of a pre-created algorithm but also allows them to create their own animation. There are two versions of POLKA: a 2D version

built on the X window System and a 3D version of the Silicon Graphics GL system. Both versions of the system have two important features. They contain ancient texts that provide “true” images [Stasko et al., June 1993], a skill not found in some systems. These primitives show smooth, continuous movements and actions on an object and not just flashing objects or color changes. These high-quality photographic skills should preserve the context and promote comprehension. The first elements that show the various acts of animation make Polka particularly useful in accurately depicting the same

functionality that occurs in the same system. The 3D version offers many automatic and easy parameters so that designers do not have to worry about image details. Editors do not need to know 3D image techniques such as blurring, ray tracking, etc., in order to create 3D visuals. To record animation in Polka, you need to create a C ++ program and inherit the basic classes offered by the system. The authors say the program is easy to use and report that students who are unfamiliar with C ++ have been successful in using it.

Samba is the animation system application of the POLKA algorithm [Stasko et al., June 1993] described above. Designed for students to be able to write algorithm images as part of their classroom assignments. Samba is designed to be easy to use and read so that students can create their own pictures. Students who write animation will be tied to the algorithm and its functionality. Therefore, as they build the animation of their algorithm, students should discover the basic features and characteristics of the algorithm. Samba is a cartoon translator and generator that can work in bulk mode. It assumes as follows a sequence of ASCII commands, one command per line. There are

different types of commands given to the system. One set of commands generates animation images. The second set of commands fixes things that are already in production. There are other types of commands that can be used to create complex and complex images. For example there are commands that can be used to have multiple views (windows) of the algorithm. The command group can be grouped together to run simultaneously. Samba uses the X-Window System and Motif. The program was developed for Windows and Java in 1997 [Stasko, 1997].

* 1. **MOCHA**

Mocha [Baker et al., 1996] was developed to provide animation algorithm on the World Wide Web. It has a distributed model with a client server configuration that separates the software

components of the animation system of a common algorithm. Two important features of Mocha is that users with limited computer power can access the animation of complex algorithms, and it provides code protection to what end users do not have access to algorithm scripting. Animation is considered an event-driven program of communication processes. The algorithm contains annotations of interesting events called algorithm functions. There is an animated section that provides multimedia visualization of algorithm functions. The animation component is further divided into a GUI, which handles user interaction, as well as an animator, which displays algorithm performance and user requests into flexible multimedia scenes. Users interact with the system using the HTML interface, which is

transmitted to the user's machine and the interface z. The security of the animation code is only achieved by exporting the visual interface code to the user's machine. The algorithm is applied to a server running on the provider's machine. Multithreading on GUI usage and animator is used to provide responsive feedback to users. Also, an object / component based software structure has been used to ensure system flexibility. Mediators are used to distinguish similarities between client interactions with servers, which provides a high level of interaction.

* 1. **Hyper media Algorithm Visualization System**

Hyper media Algorithm Visualization System (HalVis) HalVis was developed at Auburn University in the late 1990s. The program is designed with the idea that, in order to make the

algorithm visualization more academically effective, in addition to gaining visual attention (made by multiple algorithm visions), you also need to get mental attention and engage the student's mind while viewing algorithm visibility. . The Vol-8 Issue-3 2022 IJARIIE-ISSN(O)-2395-4396 16908 ijariie.com 2041 program displays algorithms in the multimedia area. HalVis has five modules. The ‘Basic’ module contains information about basic functionality (e.g. data exchange, looping performance, recursion) that is common to almost all algorithms. This module cannot be directly accessed. Requested for hyperlinks from other modules. The module introduces contextual information into a student algorithm. The ‘Vision Perspective’ module defines a specific algorithm for students in terms of commonly used metaphors. For example, defining a Merge sort algorithm uses a simulation of dividing playing cards and combining them to create a sequential sequence. Animation, text, and interaction are used to explain important algorithm features to students. The 'Detailed View' module defines the algorithm in a more detailed way. Two presentations are used for

this. One presentation contains a textual description of the algorithm and a false code. The text description contains 'basic' module links. The second presentation consists of four windows: Animation Output window displays data updates as a result of algorithm creation using smooth

animations, Activity Message window describes important events and algorithm actions using text and comment feedback, Pseudocode window displays. algorithm steps for self-expression and animation, the Performance Variables window displays a "point board similar to the panorama of the variables involved in the algorithm" [Hansen et al., 2002]. Initially, only a limited number of items can be included. This allows students to focus on low-level algorithm behavior. The ‘Full View’ module captures large data sets as incorporating to make high-level behavior algorithm clear to readers. The animations embedded in this concept are similar to those found in previous editions.

**CHAPTER 3**

**PROBLEM DEFINITION AND OBJECTIVES**

The most proven method to learn any algorithm is visualizing them. Especially for subjects like

Data Structures, where algorithms form its basis, it is not practical to memorize them theoretically

without comprehending them practically. It becomes quite confusing and tedious to visualize them on our own since abstract thinking plays a crucial role in forming its conceptual model in our mind. Thus, a need for a tool for visualizing the data structure algorithms interactively so that the student can experiment and learn as need arises.

**3.1 Problem Definition**

The whole process of designing, analyzing, implementing, tuning, debugging and experimentally evaluating algorithms can be referred to as Algorithm Engineering. Algorithm Engineering views algorithmics also as an engineering discipline rather than a purely mathematical discipline. Implementing algorithms and engineering algorithmic codes is a key step for the transfer of algorithmic technology, which often requires a high-level of expertise, to different and broader

communities, and for its effective deployment in industry and real applications we need to understand

the fundamentals of those algorithms in a very correct manner.

* 1. **Purposed Solution**

Seeing Algorithms in action would be an effective way to understand the complex data

structures. The proposed system has various interactive animations for a variety of algorithms. Our

visualization tool is written in JavaScript using the canvas of HTML and CSS. The

visualizations are meant to be fairly self- explanatory. Our proposed system can be used to enhance the traditional classroom education and textbook for Data Structures and Basic Geometric Algorithms courses.

Our System is an integrated platform where students can enhance their coding skills, teachers can evaluate student’s work and the focus is on ‘Algorithm Visualization’ for a better understanding of the algorithm’s flow and operations. It contains lab integration into one platform which makes it more feasible for teachers and students to use. This platform consists of three user groups namely student, teacher, and developer. The client side and server- side are connected using REST API, which performs retrieve and CRUD operations on the server. The server contains server-side and client-side scripts and is connected to the MongoDB database. When student’s login into the platform, they will land upon a fully interactive dashboard. This dashboard will give them their performance metrics, leaderboard, personal profile, assignment, discussion forums, and subjects. Each subject will be organized into pre- and post-assessments, brief textual explanation, line-byline visualization, and coding playground. First, preassessment will be taken which will be beneficial for students to understand their prior knowledge related to the selected algorithm/topic. Then a brief textual explanation of the algorithm will be given so that students should have some prior knowledge before directly going for the visualization. To understand algorithms better, being familiar with the significance of each line, understanding how each line is responsible for manipulating data, what changes a line makes in the input data provided, which line is the crux of the algorithm, and from where and in which direction the data flows play a very crucial role in understanding the subject better. This is where the visualization comes into the picture. Visualization of algorithms will help students understand the algorithms in a precise and accurate manner. Further coding playground will help students to get an understanding of the code and implement the algorithm programmatically. The coding platform will have compiler/interpreter support of various languages. Post assessment will finally test their overall learning from the module. The scores of the assessments will be used to generate leaderboards standings, personal growth metrics, and provide feedback for improvement (if needed). In the assignment, students will have to submit the assignments before the due date given by teachers.

In the discussion forum, students can post their difficulties. Their academic-related doubts will be solved by teachers or other classmates and their system-related doubts will be solved by developers. When teacher’s login into the platform, over the dashboard they will see a leaderboard, assignment allocation, assignment evaluation, student’s report, and discussion

forum. Inside the leaderboard, they can see the progress of students in a class. In a student’s report, they can see overall class performance and modules completed by each student. In the assignment allocation section, teachers can assign various assignments or organize a quiz for students, set the due date, and send notifications. Assignment evaluation can be done manually or automatically.

They can also post their doubts related to the system in the discussion forum. Another main task of teachers is to add students to the system and provide the credentials. The developer group holds supreme power over the application. They maintain the whole system. They can create a visualization of new algorithms, add or edit assignments and can also assign roles to the users. They can register the teacher on the platform and provide the credentials. For security, the application will have copy/paste disabled. During the test, switching tabs will cause the test to end. The test will be proctored by webcam and it will have an inbuilt timer for the test.

* 1. **Aims And Objectives**

When we learn various algorithms whether it be from data structures or any basic geometry, most of the tools explain them without providing any details about how they are actually represented and their behavior in a real program. This results in the students knowing only how the data structures

work theoretically and may not be able to use them practically for solving a programming or any other task which in turn further enlarges the gap between students’ theoretical and practical ability, adversely affecting the students’ problem-solving skill. The main objective of this visualization platform is to provide a means for learning the concepts requiring abstract thinking in a more effective as well as enjoyable way which will engage the learners and drive away any boredom. The purpose is to build a user friendly interface where the students can experiment and learn data structures. A system providing visualizations for some basic geometric algorithms and widely used data structures such as array, stack, queue, tree, heap, graph, etc. along with the animation of common operations associated with the data structures, such as inserting an element into and deleting an element from array, stack, and queue. The main aim is to mitigate the gap between students’ theoretical and practical ability related to data structures via this platform to provide a systematic approach for learning and conceptualizing implementation of data structures.

JavaScript will be used for creating animated implementations of algorithms which will be given a database connectivity.

Thus, our main objective is to help the analysis of concepts of basic algorithms to those eager to learn.

The main objective of this project is to help beginners to be able to visualize the basic algorithms and get a better understanding of the underlying operations. And obviously it is

needless to say that anyone who is willing to contribute is in voted to use their creativity in making the visualizations even better and attractive. One can add fresh Algorithms and visualization of their choice too.

This project is for educational purpose. Extract the required feature and classify among others. Users able to establish a relation between a property and a table simply by clicking on a row and dragging it over the target table. Tables are rendered in HTML inside of a div. An SVG HTML element is placed behind the tables’ div. This is where we’ll draw the relations. When table positions change, our lines need to be redrawn.

Another objective of the project is to create a web application as a visualization tool. A single-page web application built using modern JavaScript technology that will visualize the flow and logic of various sorting algorithms. The UI will contain options to select one of the sorting algorithms which were implemented and several items or elements in the data array, control buttons to start, pause, navigate to previous or next steps along with an option for sorting speed and color mode. The data array of the selected size will be filled in with randomly generated unique values. The data set is represented as a vertical bar with the height of their respective values. After the sorting is started, the stepwise arrangement of data in ascending order based on their value/height will be visualized in the UI.

* 1. **Research and experiences**

Our motivation behind the effort of gathering information from students by means of simple questionnaire was twofold: Firstly, we wanted to know how well the tool is accepted by a group of it’s potential users. Whether our design decisions were right and whether the visualization of algorithms is considered helpful at all. Secondly, we were curious, what new visualizations and

new features in general are expected by the users of the system. Their opinions could serve us as a motivation and inspiration for decisions within the next development. Respondents in our case were 53 students from four different groups having enrolled the Data structures and algorithms subject, taught in the second year of the Informatics bachelor study program. Questionnaire itself consisted of five questions. First two of them were used to get the feedback on the tool in its actual state and

its usefulness in a process of teaching algorithms and data structures. Questions from three to five were oriented to the future development of the tool. Within questions three and four, students by

their answers were able to express their opinions on the list of algorithms that could be implemented, as well as (more generally) on the areas of topics from the subject, that should be covered by the tool in the future. While these two questions were connected to the development of new (or enhancing existing) plugin modules, last question concerns the main module and its (potentially new) features.

The language of the questionnaire was Slovak (same as the language in which the subject is taught in participating study groups) and the questions had the following meaning:

1. Does using of the Algorithm Visualizer tool help to understand operation of algorithms?
   1. Yes
   2. No
   3. Don’t know
2. Which of currently available visualizations was most helpful for you?
   1. BubbleSort
   2. HeapSort
   3. InsertSort
   4. RadixSort
   5. SelectSort
3. Which next algorithms should be visualized? Please specify:
4. Which areas, covered by the subject, require visualization more than others?
5. Elementary data structures
6. Algorithm design techniques
7. Sorting
8. Advanced data structures for ADT Set implementation
9. Data structures and algorithms for external storage
10. What new features of the tool are welcomed?
11. Step back in running visualization
12. Testing mode
13. On-line availability
14. Other features:

Summary of responses in tabular form and their brief analysis follow within the rest of the section.

Responses to question 1.

1. Yes -50
2. No - 0
3. Don’t know responses -3

The opinion of almost all of students (50 from 53, Table 1) about helping to understand operation of algorithms was positive. Even if we expected it could be useful, we didn’t know it will be so clear. The result supports our belief, that it is purposeful to continue with development of the tool and its use within the teaching process.

Responses to question 2.

1. Bubble Sort -6
2. Heap Sort -15
3. Insert Sort -4
4. Radix Sort -35
5. Select Sort -4

The Radix Sort visualization seems to be the most useful with its 35 votes from students. The second place took visualization of HeapSort algorithm. As it could be seen from the sum of numbers in Table 2, some of respondents selected more then one algorithm here.

Responses to question 3.

1. Quick Sort -13
2. Trees -5
3. Graphs-2
4. Shell Sort -2

Most wanted visualization to develop is the one of Quick Sort algorithm Table 3, followed by visualization of operations on different kinds of trees. It should be noted here, that quite a big part

of students did not exploit the possibility to express their opinion and leaved this question without an answer. Some of answers probably would not be satisfied so easy, including ’as much as it is possible’, or even ’all algorithms’ (probably mentioned in the subject).

Responses to question 4.

1. Elem.DS -5
2. Alg. techniques -13
3. Sorting -36
4. Adv.DS -23
5. External storage -4

Areas requiring visualization more than others, according the answers in Table 4 are sorting (36) and advanced data structures (23). Some of sorting algorithms were implemented till now, so this result confirms our decision to start with development of plugin modules for the platform from that area. Advanced data structures in case of the subject include different kinds of tree structures or hash tables.

Responses to question 5.

1. Step back -35
2. Testing mode -8
3. On-line -25
4. Other features -2

According to results summarized in Table 5, step back in running visualization was selected as the most useful feature to implement in the future, followed by the on-line availability of the tool.

Quite small number of suggestions was provided here within the ’Other features’ option (including graphically better visualizations, optional changing algorithm properties, and better specification of input set). 188 Slavomír Šimonák

**CHAPTER 4**

**DESIGN AND IMPLEMENTATION**

Here in the proposed system, the user can select whichever model or algorithm he/she wants to study. On its selection according to the algorithm, a graph or its visual representation will be generated. On starting the animation, a systematic and detailed animation will be shown so as to how the algorithm works for a better understanding. The animation speed can be controlled according to the user’s pace. After learning, the user can also test their knowledge by trying to predict the working before playing the animation.

* 1. **The User Interface**

The design and structure of the user interface components has remained unchanged even if the underlying back-end code was refactored midway through the construction. Each component has its own feature: The canvas has twelve features; 10 control buttons, and a volume toggle button. The canvas area is where the four sorting algorithms are visualized, and that area will be the location where the sorting algorithms' output is edited in. The first row of four blue-bordered buttons at the bottom of the canvas are the selectable algorithms: Selection Sort, Bubble Sort, Insertion Sort, and Merge/Insertion Sort. This type of visualization is offered to users to select an algorithm of their choice, and to observe how that algorithm functions. Before launching the animation, the user will need to select an algorithm. The sorting algorithm must be selected before the input data type is specified. To choose between sorting input data that is already in order (as shown in Figure 1 on the previous page) or to reverse and randomize the order, the three gray-bordered buttons on the left of the bottom row are available (shown in Figures 2 and 3 in the following paragraphs). Sorted order is the default. The sorting algorithm is picked once the input and sorting method have been selected. Following, the “Start” button in the next row of buttons is clicked to perform the sort from the beginning to the end. The user can click the yellow-orange-bordered “Step” button to watch the algorithm step-by-step. Once the process is already underway, you can simply stop it by pressing the “Stop” button. In order to reduce misunderstanding in userfriendliness, I've tried to make the interface basic and placed buttons

related to one another for easy access. The algorithm buttons are on separate tiers and have a blue coloration associated with them. Additional buttons are arranged in a grouped fashion, with a gray background. The "oddballs" are the animation controls. Despite being grouped together, each controller has a distinct color to denote the kind of animation it does. The colors are modeled on a traffic light with green being the go signal, red being the stop signal, and yellow being the signal to slow down (or in this case, yellow-orange means pace yourself). Additionally, each button also provides visual feedback to the user by changing color as the cursor hovers over it. The volume toggle button is the final function that is available on the Web page. The button appears to the left of a speaker picture on the lower left-hand side of the web page. While conducting some study on sound effects for animations, I realized the tool could be more participatory by both hearing and seeing the animation, rather than merely observing. To determine the sound each bar makes, use the following rule: each bar has a sound allocated to it based on its height. To hear four octaves on a piano, from low to high, the bars need to be in order. With the sound enabled, each bar in the sound animation plays a different tone from left to right when it appears. You'll only be able to hear the full four octaves in order if the bars are out of order. The bar's presently being played color will change to green, and then, when the sound animation is finished, it will change back to blue. However, because the animation is hungry with memory, the animation may stop momentarily and then resume. This means that the sound animation is turned off by default, but the user has the option to toggle it. You'll get the best results if you use the sound animation with Selection Sort.

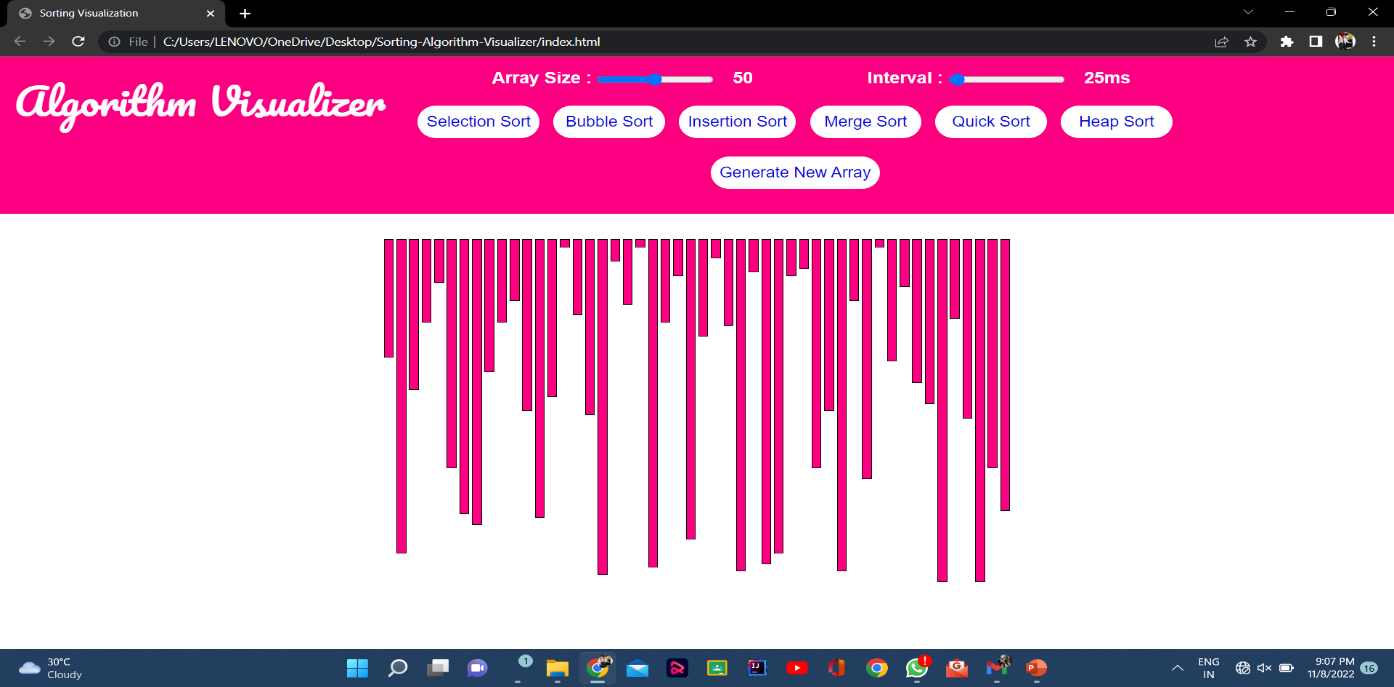
****

Figure 4.1: Main Window of Application

Finally, there is one feature that you will discover only after making a selection and attempting to sort: you may modify the sorting algorithm on the fly. In other words, you can pause an animation

while using a sorting algorithm like Selection Sort, then choose another algorithm, like Bubble sort, while the animation is paused. To continue sorting the bars, start the animation by clicking "Start" or "Stop." How the movement of data changes based on what algorithms are already affecting it is remarkable to behold. Also, sorting semi-sorted data using the provided options provides a unique view on how the algorithms function.

* 1. **System Architecture**

HTML5, CSS, and JavaScript make up the back-end code. There are three varieties of code in one .html file and all three can be executed from this file alone. Including different types of web languages in a single page is one of the shortcomings of HTML 5. Since, therefore, there were three different types, each had been segregated, producing three different files (plus the miscellaneous sound and image files). Readability and keeping relevant code together are benefits of excellent programming practices. However, in the end, I opted not to break the code into two separate sections because of these two reasons. By just having to worry about one project file

instead of three, the project may be more easily transported and sent. And because the changes to the coding languages are identified unambiguously in the project file, they do not reduce readability. An RIA can have more than one programming language in a single file (Rich Internet Application). As you can see, the three coding languages are the only important components. However, since JavaScript runs immediately in the browser, it is unnecessary to employ a server on the back-end (like PHP). HTML5 and CSS are employed in web development. As illustrated with a single, bidirectional arrow, the HTML5 and JavaScript communicate to run the relevant algorithms and update the interface. The code for HTML5 and CSS did not change significantly throughout the project. The parts of HTML5 that were updated were the function calls for each button, since they were altered from a functional programming mindset to an object-oriented one. We've abstracted away all of the back-end code behind all of the different algorithms and animation selectors.

* 1. **Implementation**

The use of HTML5 (Hypertext Markup Language 5), JavaScript, and CSS combine to form this project's implementation (Cascading Style Sheets). There is only one project file which is an HTML file and contains the code. The only additional piece of code added to the main HTML file

is the .m4a sound files to support the sound animation functionality (which are saved as .m4a files). As of now, I only did extensive testing using the Mozilla Firefox browser, and it's the browser of choice in this context. However, tests done quickly revealed the possibility of Google Chrome and Safari integration. This software uses both object-oriented and functional programming paradigms in how it organizes the code. Before the final phase of development, the design was almost completely functional, where only three objects were used: one to control the canvas that displayed the animation, another to represent a piece of data, or “bar” object (blue rectangle with dynamically changing height and position), and a final one to represent the positions that each bar moved to, or “pos” objects. Although this incorporated several function calls, some instance variables and Boolean values were utilized to keep track of the algorithm picked and when to animation, but this led to a greatly integrated mass of code that was difficult to maintain. Several big refactorings later, the code has now taken on the form of a Model-View

Controller Architecture. Although, because of its functional character, it possesses a multitude of individualized functions that alter the instance variables and Boolean values, which means it has a multitude of functions that directly alter the View and Controller. The major module in the HTML code between the tags is known as the global scope. Everything within the framework is able to access the aforementioned variables and methods.

* + 1. **The View**

There are three items on the view: the sort Area, the bar, and the position. These objects operate within the container defined by the tags in the .html file. This function's space is sometimes referred to as the "main" function, the first function invoked in a program.

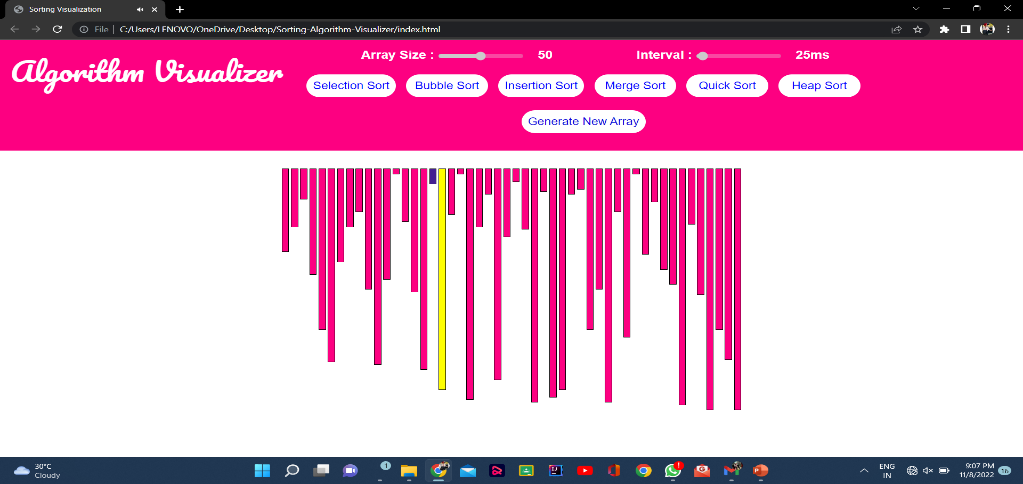


Figure 4.2 : View of the Application

It is the sort Area that keeps the bars up to date using a timer, while at the same time generating the bar graph. As a result, whenever "Step" is invoked, the bar values are updated depending on the steps array (discussed later). In the sort Area, after every second iteration of the timer, the rectangles will be redrawn with varied heights that represent the new values. The bars change sixty times per second, so when the “Step” button is selected, the change is instantaneous. In the

sort Area, the bar object represents each piece of data. The statement encompasses all of the aspects of color, value, location, height, and sound. While having a distinct array named bars for the current bars in the bar graph helps preserve attributes such as the total number of bars (total value) independent of other characteristics, it is simple to update any or all of the attributes by iterating over the array as necessary. To update the bars to a fresh data configuration, the In Order, Reverse, and Random buttons iterate fast across the array. The bar object is related to the pos object (which is short for position). The region on the canvas that is updated when the Sort Area event fires is an x-y pixel coordinate grid. This item was made to make arranging the bars a little bit simpler (1-32). Thus, if I wanted to move a bar, I would supply merely the number of the bar's location. In order for the bar to move, it will first determine the exact coordinates and then go to that location. Another way to say this is to say that, position one is defined by the two-dimensional coordinate pair (9, 135), which is the bottom left-hand corner of the bar. As long as each bar has a rectangular object that is associated with it, as well as a top left-hand corner point that defines the rectangle's height, the bar must be relocated to its right location.

* + 1. **The Model**

The model is made up of one item, known as the sorter. This object houses the algorithm's code divided into methods. Start method centralizes on an integer constant and uses it to order the algorithm's possible algorithms. This object is directly controlled by the four sorting algorithms shown on the user interface as "Selection Sort, Bubble Sort, Insertion Sort, and Merge/Insertion Sort." The sort algorithm method is invoked as the user selects a sorting technique and clicks on one of the sort algorithm buttons. Then, when the algorithm sorts the data, a trace is created. The steps array, which contains all the movements in the animation, is a twodimensional integer array that is available to methods on the web user interface. A typical back-end code interface is implemented using the steps array. The “Start,” “Stop,” and “Step” buttons function as controllers

for which sub-array will be displayed on the canvas, after the computational back-end has completed the tracing.

If this loads before the user has selected "Start" or "Step", then this represents a user action. When you click the “Step” button, the next step on the canvas loads and animates. An animated

sequence of a single bar moving across the others is produced because the timer continually redraws the bars. There's no “Start” button, only a “Step” button that is set to go off on a timer. Using a two-dimensional array gives you the option to view the sorting algorithm's stages within the View. The process of adding an algorithm is similar to writing down the trace of the new algorithm, which is then saved in the same location. To complete the algorithm's walkthrough, the View will cycle through the data and update the bars in the bar graph to show how the algorithm calculated the steps it took. It's important to note that if the algorithm generated a change in the position of a piece of data, the steps are merely recorded.

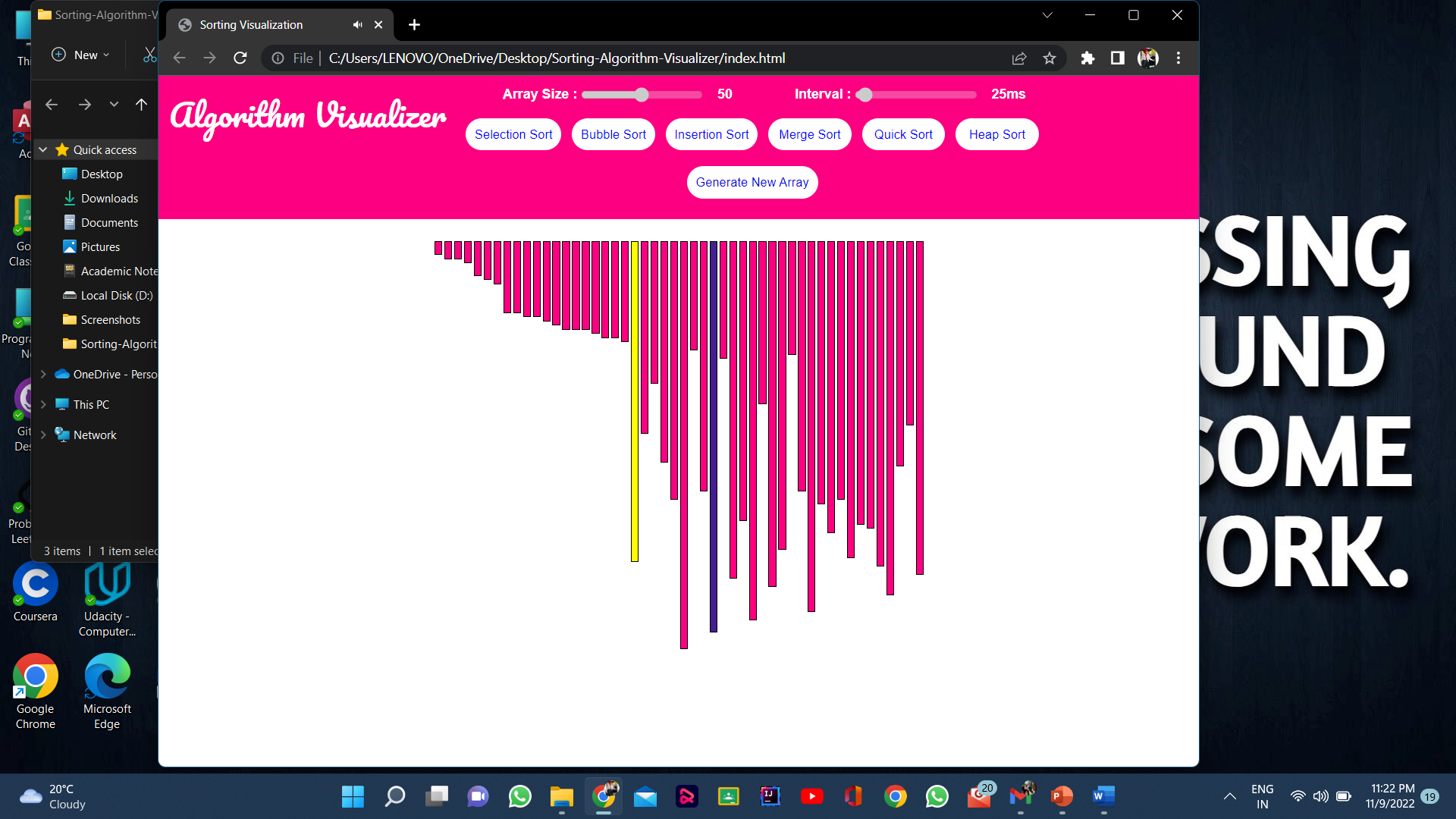


Figure 4.3 : View of the Insertion Sort

Let's give an example: When sorting the pieces of data using Selection Sort, each piece of data is moved to its final and accurate location after one step, whilst the others require numerous steps to get at their final positions. While this sorting method appears to do the most effort compared to other sorting methods, it finishes sorting the most slowly. As a result, the

visualization doesn't provide the correct visual impression of the data comparisons, which is one of the most important aspects of sorting algorithms. Two-dimensional arrays do demand more memory than a one-dimensional array.

The size of the array is based on the number of steps that are required to sort the data. We may assess the algorithm's space needs by examining how long it takes. In Computer Science, using Big-Oh analysis is the standard way for determining how long something will take. The notation consists of a capital letter O, which represents the worst-case performance of the algorithm in question, followed by a constraint in parentheses that describes the worst-case performance of the algorithm.

* + 1. **Sample Code**

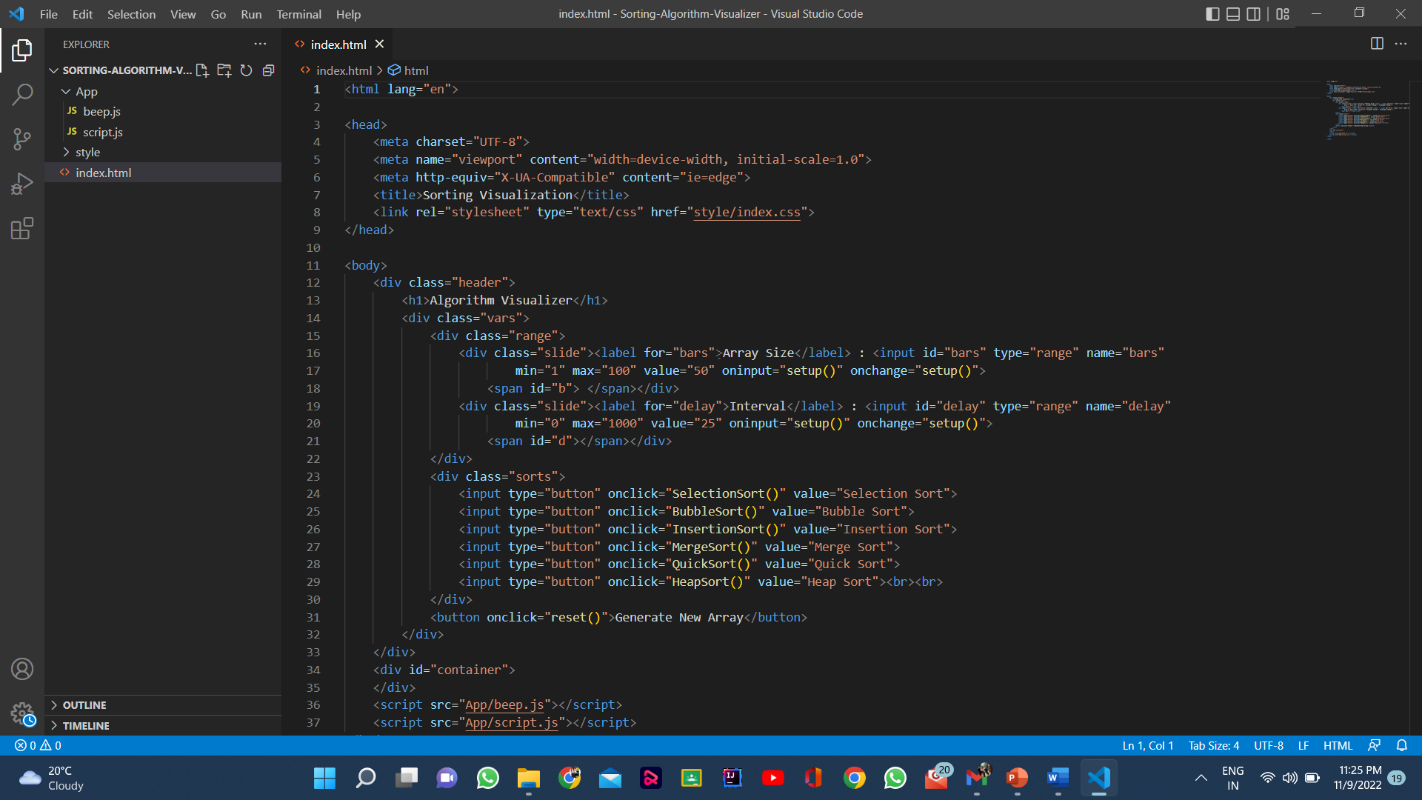
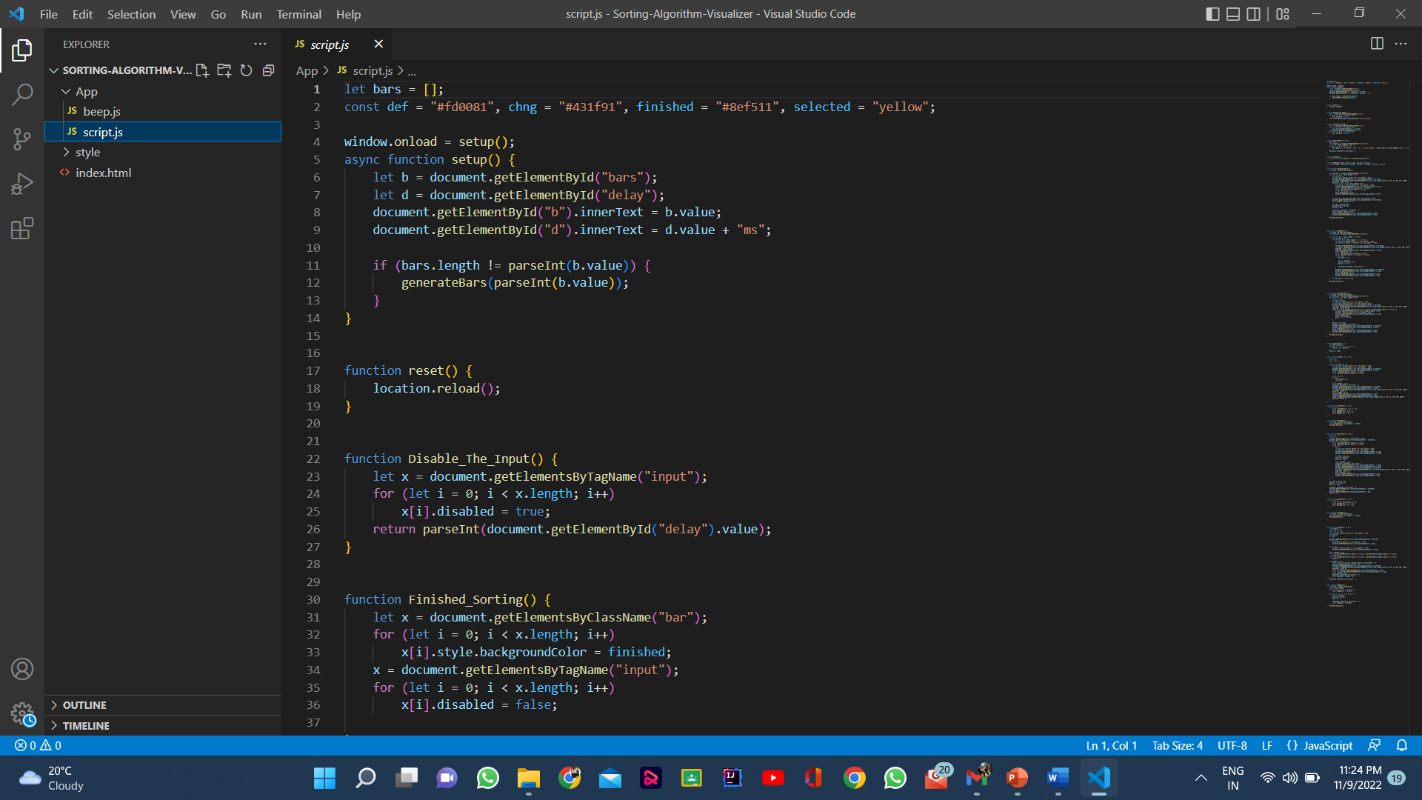


Figure 4.4: Sample code for HTML

Figure 4.5: Sample Code for JavaScript

**CHAPTER 5**

**RESULTS AND DISCUSSIONS**

The best way to go about using the tool is to first select the ordering of the data and then select which algorithm to visualize. When any one of the algorithm buttons are selected, it will sort the data as it appears on the interface. The ordering takes precedence, as selecting the ordering after the algorithm updates the interface momentarily, while the code has already run the initialization with the previous data set. After conducting the surveys, this sparked some confusion as the algorithm buttons are listed above the ordering buttons in the interface. One student commented on having difficulty trying to start sorting, thinking that it may be the cause of pressing the buttons in the wrong order, which in turn did not run the animation. The responses of all the students can be found in Appendix D. Overall, there was not a significant advantage in using my animation tool to help learning about sorting algorithms. By looking at the student responses for question 3, which asked if their understanding of a particular algorithm changed after using the tool, 5 of the 13 students (38%) said yes in some way. The other 7 did not find it very helpful, even though most appreciated the idea of the tool. One student, however, gave a false positive to the tool being helpful (whom I did not include in the 5 that said it was helpful). A drawback to the animation is that it only shows the movements without the comparisons that lead to the movements of the data. This student saw how Selection Sort completes quickly compared to the other algorithms, as there are O(n) swaps that take place, which is beneficial in avoiding unnecessary data moves the computer needs to make. In contrast, the process of comparing the data results in a O(n2) runtime (the slowest overall). Another student noted this discrepancy in response to question 5 that asked for comments and feedback, noting that Merge Sort is the best of the four sorts. Merge Sort has an average runtime of O(n log2 n), which is the best average runtime out there. One way to resolve this would be to integrate visualizing the comparisons as well as the movements. This way, the bars would change color when an algorithm is comparing data, taking up more time in the animation. Selection Sort and Bubble sort use the most comparisons, so their time to complete would slow down and be more appropriate compared to the other algorithms.

* 1. **Survey Results**
* Through a survey conducted by us we inferred that 60% of the students responded better to understanding concepts through visualization rather than their own imagination or the regular teaching methods.
* It's been proved time and again through different experiments and research on masses that any kind of visual aid such as an image, a video or even an animation clip tends to be remembered more by humans.
* Not only will the visualization help but due to features of mazes and patterns in our application, the students can relate the working of the algorithms to real life examples (likes obstructions in the form of walls).
* Often we see teachers struggling to make students understand concepts such as algorithms without it getting monotonous, that's where our project comes into play as a great teaching aid. Because of our user-friendly and engaging interface the problem of distraction or losing interest tends to decrease, making it very efficient.
* Our project can easily be incorporated alongside our education system by promoting different ways of learning rather than the old age blackboard method as we just need to access a website hosted on the internet to use the application.
* And with uncertain times like nowadays, we cannot only afford to be dependent only on our teachers and one to one offline teaching to understand different concepts. E-learning is the new age learning technique and our project is a step towards reinforcing this method of learning.

**CHAPTER 6**

**CONCLUSION AND FUTURE SCOPE**

In a nutshell, we identifies some issues by experiencing them ourselves in the present learning strategies in use and we tried to help better the scenario for aspiring students in this domain through or progressive web application. When we ourselves were learning the subject of algorithms in our curriculum, we found it a bit difficult to relate and understand the practical implementation of the algorithms owing to the difficulty in communication of the concepts from the teachers to the students. We found that there were no proper means that the teachers could adopt to portray their ideas in a better and easy manner in front of the students. So, we built an application which could help in the following ways:-

* It has been found that it becomes easier for humans to retain the concepts when learnt through visuals than just textual or speech explanations.
* Application is extremely user friendly so people of any age can engage and start learning new things right away.
* The application would also include various fun filled activities like visualization through mazes and patterns.
* This application will also include a parameter of time complexity which will be displayed after the particular sorting algorithm has completed its execution for better comparison.
* Almost all the famous and important algorithms will be present in the application for visualization with both path-finding and sorting algorithms present in same application, thus making it a one stop destination for the students of this domain.
  1. **Conclusion**

According to our ﬁndings, algorithm visualization can be seen as a valuable supporting tool, used in addition to standard ways of education in the ﬁeld of computer science. Within the paper we provided an overview of the algorithm visualization platform as well as our practical experiences with the system. We believe (and the results of questionnaire support our belief) it helps to improve the quality of education in the ﬁeld and contribute to the solution for some of the problems in higher

education mentioned at the beginning of the paper. There are still open issues with using algorithm visualizations. Algorithm visualizations can help understanding the principles, but do not replace the need to implement algorithms by students is a chosen programming language.

Another drawback of using algorithm visualizations within our subject is the lack of the tool oﬀering required visualizations in a single package with the uniﬁed interface. This platform can also be considered as a step in this direction. Generally, more systematic evaluation of algorithm visualization tools is required, as there is rather informal evidence available that applications of algorithm visualizations are useful. We summarized results of the questionnaire ﬁlled in by students in order to support our decisions on further development of the platform, too. Our intentions here include development of new plugin modules from the area of sorting algorithms and more complex data structures. Some of proposed core-related features are on the list too (like graphically better visualizations, optional changing of algorithm properties), but some of them will probably not be implemented in a near future (like undo/step back in running visualization), as they would require more fundamental changes. Except the extensions mentioned within the questionnaire, we also consider some other interesting features: dynamic changes in algorithm pseudocode reﬂected in visualization, diﬀerent visual views on running algorithm or simultaneous comparison of diﬀerent algorithm visualizations.

Based on our findings, algorithm recognition can be seen as a useful supportive tool, used in addition to conventional educational methods in the field of computer science. Algorithms are an exciting way to be used in viewing. To visualize the algorithm, we simply do not enter the data into the chart; no primary database. Instead, there are sound rules about behavior. This may be the reason why algorithm recognition is rare, as designers try novel forms to better communicate. This is a good reason to read them. But algorithms are also a reminder that visualization is more than just a tool for finding patterns in data. Visualization enhances one's visual system to enhance one's intelligence: we can use it to better understand these important invisible processes, and perhaps other things.

We started our project by studying a number of the well-known algorithm visualizations that are developed over a few years. According to our findings, algorithmic visualization are often seen

as a valuable supporting tool, utilized in addition to straightforward ways of education within the field of computer science. With the execution of this project, we have got with success attain our objective of our project is to engraft Graph Path Finding with Visualization and differentiate their performance. As is the case with most other teaching areas, there has been a major gap between the idea and practical understanding of algorithms realization. This is often true for shortest paths

algorithms and in specially for Dijkstra algorithm. The main goal of the project is to use it from research educators and students for teaching and studying the existing known graph algorithms.

The main plan of the system is to provide an associate educational environment for both instructors and students to facilitate the learning process in economical way. This web-based animation tool for viewing the following sorting algorithms functions in great part because of all the time and effort that I invested into it. In spite of its memory overhead, the feedback given to it was mostly good from the students that worked with it. This is consistent with my prior research, which revealed that there was no substantial difference in learning the content. What I do agree with totally is the attitude that holds there is a great need to investigate and produce animated presentations to enhance education in the classroom. Overall, I am not concerned that a large rework to a different language will be required soon because JavaScript is still one of the most popular web languages. We all know about my laundry list of upcoming projects, but there is one elephant in the room that still has to be addressed: resolving the memory difficulties. Following this, we would implement Merge/Insertion Sort, which takes into account the Merge Sort. Then, I would start up Quick Sort so as to finish the job because the code is ready to be integrated. Finally, I would make the online tool available to the public, with the feature I want most, which is to make it available to the public. This might be tough as well. The application that created the animation tool knows that it's available locally, but because of concurrency, it can serve numerous requests to the web site by separate users. As I try to figure out how to make the code as efficient as possible, I'd need to spend some time thinking about how to make it work with numerous people using it. This would be excellent, as it would enable a form of comparison study.

From the results that have been obtained throughout this project, the Java codes of the sorting algorithm user interface were successfully designed, synthesized and analyzed. The sorting algorithm for the Java program has proven that each of the types for sorting algorithm is working

precisely. Not only is that, based on the table of the result for run time given by each of the sorting algorithm in line with the theory. This result verified that the half of the objectives for this project have been achieved. This is because this project does not have any animation or visualization for sorting algorithm. It can only measure the performance of the run time based on different algorithms. As a conclusion, the development of system for algorithms visualization using JavaScript has been successfully made by wanted targets and particulars. Aside from that, this study can be enhanced to make further developed and dependable. There are some suggestions in

order to improve this study. First of all, the design of this sorting algorithm visualization should be transform to sorting algorithm animation in order to use as a good teaching tool and able to gain a good understanding of the sorting algorithm for computer science student. Besides that, this program should be done using Netbeans software. This is because the Netbeans has their own user interface design. Netbeans can be used when prototyping and designing applications on top of the NetBeans Platform. It has drag-and-drop capabilities and point-and-click features to make the ideal environment for user interface design or visualization.

* 1. **Future Scope**

A visualization tool for visualizing some basic geometric algorithms along with data structure algorithms and operations associated with them has been presented. This tool provides an easy way to play and learn data structure concepts with its user-friendly and self-explanatory interface. In this system, only some commonly used and basic algorithms are implemented like arrays, queues, stacks, linked lists, linear and binary search tree, various sorting methods etc. Its scope can be extended by implementing more complex algorithms in the software. It can also be categorized for a more systematic interface. Developing and implementing a mechanism for the software package to recognize the user- defined observable data structures, and leave the implementation to the user is yet another way to extend its current scope, allowing users to use their own observable data structures, thus adding more flexibility to the software.

With the shift of remote and digital literacy, a combined platform serving the effective literacy requirements of students is required. Algorithm Visualizer is a combined platform that's a comprehensive result for educators and students to educate and learn online effectively. It

substantially focuses on “algorithm visualization", which allows a better understanding of its inflow and operation

The developed platform offers a complete perspective of visualizations for sorting and pathfinding algorithms so far. Also, the web-based platform can run on small devices while providing the feature to visualize at one’s own pace. As a future scope, more Algorithms for Trees, Graphs, Linked List and many more can be added. To empower learning of the learner concept of community learning through forums, discussion and user-based feedback can be added. The objective of the platform is to reduce the fear level in the minds of learners especially students regarding Algorithms and Data Structures.

**REFERENCES**

1. “E-learning Tool for Visualization of Shortest Paths Algorithms” by Daniela Borissova and Ivan Mustakerov, ResearchGate, July 2015.
2. “Algorithm Visualization: The State” of the Field by Clifford A. Shaffer, Matthew L. Cooper, Alexander Joel D. Alon, Monika Akbar, Michael Stewart, Sean Ponce and Stephen H. Edwardsacm Transactions on Computing Education, Vol. 10, No. 3, Article 9, Pub. date: August 2010.
3. “Visualizing sorting algorithms” by Brian Faria, Rhode Island College, 2017.
4. “ViSA: Visualization of Sorting Algorithms” by Tihomir Orehovački, ResearchGate, May 2012.
5. “ Finding the shortest path in a graph and its visualization using C# and WPF” by Radoslav Mavrevski,Metodi Traykov, Ivan Trenchev, International Journal of Computers, ISSN: 2367-8895, Volume 5, 2020.
6. “Visualization of Abstract Algorithmic Ideas” by Ludˇek Kuˇcera, Proceedings of the 10th International Conference on Computer Supported Education (CSEDU 2018).
7. “Willow: A Tool for Interactive Programming Visualization to Help in the Data Structures and Algorithms Teaching-Learning Process” by Pedro Moraes and Leopoldo Teixeira, SBES 2019, September 23ś27, 2019, Salvador, Brazil
8. K. Mehlhorn, P. Sanders, Algorithms and Data Structures (Springer-Verlag, Berlin Heidelberg, 2008)
9. J. Genči, Possibilities to Solve Some of the Slovak Higher Education Problems Using Information Technologies, In proceedings of: 10th IEEE International Conference on Emerging eLearning Technologies and Applications, ICETA2012, Stará Lesná, The High Tatras, Slovakia, November 8-9, 2012
10. S. Khuri, Designing Eﬀective Algorithm Visualizations, In proceedings of: First International Program Visualization Workshop, ITiCSE, Porvoo, Finland, July 7 - 8, 2000
11. C.D. Hundhausen, S.A. Douglas, J.T. Stasko, A Meta-Study of Algorithm Visualization Eﬀectiveness, J. Visual Lang. Comput. 13, 259–290, 2002
12. V. Lazaridis, N. Samaras, A. Sifaleras, An empirical study on factors inﬂuencing the eﬀectiveness of algorithm visualization, Comput. Appl. Eng. Educ. 21, 410–420, 2013
13. D.J. Jarc, M.B. Feldman, R.S. Heller, Assessing the beneﬁts of interactive prediction using Web-based algorithm animation courseware, Proceedings of SIGCSE 2000 (ACM Press, New York, 2000)
14. S. Diehl, Software visualization: Visualizing the Structure, Behaviour, and Evolution of Software (Springer, NewYork, 2007) 187
15. M.E. Tudoreanu, R. Wu, A. Hamilton-Taylor, E. Kraemer, Empirical Evidence that Algorithm Animation PromotesUnderstanding of Distributed Algorithms, In proceedings of: IEEE Symposium on Human Centric Computing Lan-guages and Environments, HCC02, Arlington, Virginia, September 2002
16. B.A. Price, R.M. Baecker, I.S. Small, A Principled Taxonomy of Software Visualization, J. Visual Lang. Comput. 4(3),211–266, 1993.
17. Ž. Šuchová, Visualization of Algorithms and Data Structures, Bachelor thesis, DCI FEEI TU of Košice, Bachelorthesis, 2010 (in Slovak)
18. S. Diehl (Ed.), Software Visualization, Lecture Notes in Computer Science 2269, 2002
19. M.H. Brown, R. Sedgewick, A system for algorithm animation, Proceedings of the 11th annual conference onComputer graphics and interactive techniques, SIGGRAPH’84 (ACM New York, NY, USA, 1984)
20. G. Rößling, B. Freisleben, ANIMAL: A system for supporting multiple roles in algorithm animation, J. Visual Lang.Comput. 13(3), 341–354, 2002
21. S. Šimoňák, Algorithm Visualization Using the VizAlgo Platform, Acta Electrotechnica et Informatica 13(2), 54–64,2013
22. A. Sajko, Algorithm Visualization, Bachelor thesis, DCI FEEI TU of Košice, 2012 (in Slovak)
23. A. Gogeshvili, AVL Tree visualization, 2007, available: http://www.qmatica.com/DataStructures/Trees/AVL/AVLTree.html
24. J. Morris, Hash Table Animation, 1998, available: http://people.cs.pitt.edu/~kirk/cs1501/animations/Hashing.html
25. CORMEN, T. H.; LEISERSON, C. E.; RIVEST, D. L.; STEIN, C. Introduction to algorithms. Second Edition. 2001. ISBN 0-262-03293-7.
26. KNUTH, D. The Art of Computer Programming: Fundamental Algorithms. Third Edition. 2004. ISBN 0-201-89683-4.
27. SIPSER, M. Introduction to the Theory of Computation. Boston, MA: PWS Publishing Company, 1997. ISBN 0-534-94728-X.
28. BĚLOHLÁVEK, R. Algoritmická matematika 1 : část 2. Available also from: http://belohlavek.inf.upol.cz/vyuka/algoritmicka-matematika-1-2.pdfi.
29. KNUTH, D. The Art of Computer Programming: Sorting and Searching. Second Edition. 2004. ISBN 0-201-89685-0.
30. SEDGEWIK, R. Algorithms in C : Fundamentals, data structures, sorting, searching. Third Edition. 2007. ISBN 0-201-31452-5.
31. GeeksforGeeks. Available from: hhttps://www.geeksforgeeks.org/i.
32. BĚLOHLÁVEK, R. Algoritmická matematika 1 : část 1. Available also from: http://belohlavek.inf.upol.cz/vyuka/algoritmicka-matematika-1-1.pdfi
33. Stackoverflow. Available from: https://stackoverflow.com/i
34. Java documentation. Available from: hhttps://docs.oracle.com/javase/8/i.
35. Ahmad Affandi Supli, Norshuhada Shiratuddin and Syamsul Bahrin Zaibon, “Critical Analysis on Algorithm Visualization Study”, International Journal of Computer Applications (0975 – 8887), Volume 150 – No.11, Sep 2016.
36. Christopher D.Hundhausen, Sarah A.Douglas And John T.Stasko, “ A Meta-Study of Algorithm Visualization Effectiveness”,Journal of Visual Languages & Computing,Volume 13, Issue 3, pp. 259-290, Jun 2002.
37. C. Shaffer. A Practical Introduction to Data Structures and Algorithm Analysis. Prentice Hall, second edition, 2001.
38. P. Saraiya. Effective features of algorithm visualizations. Master’s thesis, Department of Computer Science, Virginia Tech, July 2002.
39. K. Mehlhorn, P. Sanders, Algorithms and Data Structures (Springer-Verlag, Berlin Heidelberg, 2008)
40. J. Genči, Possibilities to Solve Some of the Slovak Higher Education Problems Using Information Technologies, In proceedings of: 10th IEEE International Conference on Emerging eLearning Technologies and Applications, ICETA2012, Stará Lesná, The High Tatras, Slovakia, November 8-9, 2012
41. S. Khuri, Designing Effective Algorithm Visualizations, In proceedings of: First International Program Visualization Workshop, ITiCSE, Porvoo, Finland, July 7 - 8, 2000
42. C.D. Hundhausen, S.A. Douglas, J.T. Stasko, A Meta-Study of Algorithm Visualization Effectiveness, J. Visual Lang.Comput. 13, 259–290, 2002
43. V. Lazaridis, N. Samaras, A. Sifaleras, An empirical study on factors influencing the effectiveness of algorithm visualization, Comput. Appl. Eng. Educ. 21, 410–420, 2013
44. D.J. Jarc, M.B. Feldman, R.S. Heller, Assessing the benefits of interactive prediction using Web-based algorithm animation courseware, Proceedings of SIGCSE 2000 (ACM Press, New York, 2000
45. S. Diehl, Software visualization: Visualizing the Structure, Behaviour, and Evolution of Software (Springer, New York, 2007) 187
46. M.E. Tudoreanu, R. Wu, A. Hamilton-Taylor, E. Kraemer, Empirical Evidence that Algorithm Animation PromotesUnderstanding of Distributed Algorithms, In proceedings of: IEEE Symposium on Human Centric Computing Lan-guages and Environments, HCC02, Arlington, Virginia, September 2002
47. B.A. Price, R.M. Baecker, I.S. Small, A Principled Taxonomy of Software Visualization, J. Visual Lang. Comput. 4(3),211–266, 1993
48. Ž. Šuchová, Visualization of Algorithms and Data Structures, Bachelor thesis, DCI FEEI TU of Košice, Bachelorthesis, 2010 (in Slovak)
49. S. Šimoňák, Algorithm Visualization Using the VizAlgo Platform, Acta Electrotechnica et Informatica 13(2), 54–64,2013
50. A. Sajko, Algorithm Visualization, Bachelor thesis, DCI FEEI TU of Košice, 2012 (in Slovak)
51. https://www.researchgate.net/publication/3923388\_A\_tool\_for\_data\_structure\_visualization\_and\_user-defined\_algorithm\_animation
52. https://www.researchgate.net/publication/242356207\_A\_Tool\_for\_Data\_Structure\_Visualization\_and\_User-defined\_Algorithm\_Animatione-ISSN:2582-5208International Research Journal of Modernization in Engineering Technology and Science
53. https://www.deepdyve.com/lp/association-for-computing-machinery/algorithm-visualization-areport-on-the-state-of-the-field-WnWaZNoy0v
54. https://www.irjmets.com/uploadedfiles/paper/volume2/issue\_4\_april\_2020/788/1628083002.pdf
55. https://ieeexplore.ieee.org/document/9378249/authors#authors
56. https://www.sciencedirect.com/science/article/pii/S1110866518302603
57. https://visualgo.net/en
58. https://dl.acm.org/doi/10.1145/1900008.1900105
59. https://www.sciencedirect.com/science/article/pii/S1110866518302603
60. https://www.researchgate.net/publication/278027474\_AlgoVis\_A\_Tool\_for\_Visualizing\_and\_Simulatio n\_of\_Data\_Structure\_Algorithm
61. https://www.researchgate.net/publication/2551844\_Using\_StudentBuilt\_Algorithm\_Animations\_as\_Learning\_Aids
62. Eric Elliot, “Programming JavaScript Applications: Robust Web Architecture with Node, HTML5, and Moderns JS Libraries”.

.