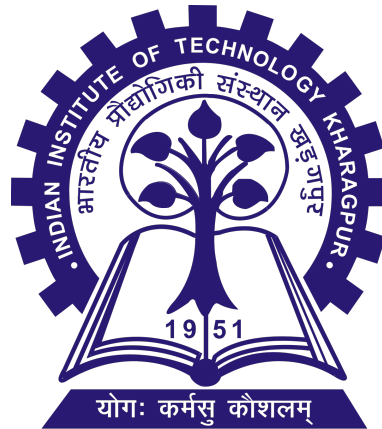


INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Kharagpur, West Midnapore, West bengal - 721 302



A Dissertation Report on

## Algorithms Visualization Web Application

Submitted in partial fulfillment of the requirements for the award of the degree of

*Master of Technology*

in

*Computer Science and Data Processing*

by

AuthorName Diganta Diasi

Roll 21MA60R06

Under the Guidance of

**Prof. Somesh Kumar**

Professor,

Dept. of Mathematics,

IIT Kharagpur.



Department of Mathematics

IIT kharagpur

Kharagpur - 721 302.

April 2023

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Kharagpur, West Midnapore, West bengal - 721 302.

Department of Mathematics



## CERTIFICATE

This is to certify that this thesis entitled “**Algorithms Visualization Web Application**” submitted to Indian Institute of Technology Kharagpur by Diganta Diasi for the degree of Master of Technology in Computer Science and Data processing is the bonafide record of original work done by the candidate under my supervision. The work was planned, organized and executed in the Department of Mathematics, IIT Kharagpur. We further certify that the entire work represents the independent work of Diganta Diasi and all the project work was undertaken by the candidate under my supervision and guidance. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Somesh Kumar  
19.04.23

Signature of the Supervisor

Prof. Somesh Kumar

Department of Mathematics,

IIT Kharagpur - 721302

Date: 20.04.2023

# Acknowledgements

Acknowledgment I would like to acknowledge and give my warmest thanks to my supervisor Prof. Somesh Kumar, Dept. of Mathematics, IIT Kharagpur who made the thesis work on “ **Algorithms Visualization Web Application**” possible. Her guidance and advice carried me through all the stages in semester-3 of writing my work analysis. I am eternally grateful for this. I would like to acknowledge that this project was totally done by me and not by someone else.

**Signature**

Diganta Diasi  
Mtech Student,  
Roll- 21MA60R06,  
Dept. of Mathematics,  
2021 - 2023  
IIT Kharagpur.

**Date:** 20.04.2023

# ABSTRACT

Algorithm analysis and design is a key challenge for students of computer and information science. The key causes of high dropout and failure rates in basic programming courses include fear of programming, a lack of motivation, and the abstract nature of programming ideas. Several experts have proposed several methods to motivate and aid students. Although it has been noticed that some of these tools benefit in the development of programming skills, the issue remains largely unresolved. The "Algorithms Visualizer" tool is described in this project.

Visualisation is a more effective approach of learning topics than traditional ways. Modern technology enables the development of e-learning tools, which also contribute significantly to the advancement of computer science education. The goal of this project is to develop a "Algorithm Visualizer" web-based e-learning tool that can be used to visualise all search algorithms, sorting algorithms, binary search tree algorithms, and shortest route algorithms. Implementing algorithms such as bubble sort, insertion sort, selection sort, fast sort, merge sort, heap sort, Dijkstra, BFS, and DFS highlight the project's conceptual applicability.

This project seeks to complete all of these requirements using HTML, CSS, JavaScript, and Bootstrap. The end result is a web application that allows any user to observe and learn how the algorithms function. The project's ease of use provides users with straightforward operational methods. The preliminary findings of utilising the web app indicate that the benefits of this e-Learning tool for students with a decent comprehension of algorithms are promising.

# Contents

<b>Acknowledgements</b>	<b>i</b>
<b>Abstract</b>	<b>ii</b>
<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Theoretical Background</b>	<b>2</b>
2.1 Algorithms . . . . .	2
2.2 Searching Algorithm . . . . .	3
2.2.1 Sequential search . . . . .	3
2.2.2 Interval Search . . . . .	3
2.3 Sorting Algorithm . . . . .	3
2.4 Binary Search Tree . . . . .	4
2.5 Graph Algorithm . . . . .	4
2.5.1 Breadth-First Search . . . . .	5
2.5.2 Depth-first search . . . . .	5
2.5.3 Dijkstra's algorithm . . . . .	5
2.6 Measurement of Algorithm . . . . .	7
<b>3 Need Of Visualization and Objective</b>	<b>8</b>
3.1 Visualization . . . . .	8
3.2 Objective . . . . .	9
<b>4 Web application as a Visualization tool</b>	<b>10</b>
4.1 HTML . . . . .	10
4.2 CSS . . . . .	11
4.3 JavaScript . . . . .	12
4.4 Bable . . . . .	17
4.5 GitHub and Git . . . . .	18

<b>5</b>	<b>Implementation of Project</b>	<b>19</b>
5.1	Building the application . . . . .	19
5.2	Structure of the project . . . . .	20
5.3	Code . . . . .	20
5.4	Hosting . . . . .	23
<b>6</b>	<b>Project Outcomes</b>	<b>24</b>
6.1	Welcome Page . . . . .	24
6.2	Searching Visualizer . . . . .	25
6.3	Sorting Algorithm . . . . .	26
6.4	Path-finding Algorithm : . . . . .	28
6.5	Binary Search Tree . . . . .	30
<b>7</b>	<b>Modeling Analysis</b>	<b>32</b>
<b>8</b>	<b>Conclusions</b>	<b>34</b>
	<b>References</b>	<b>35</b>

# List of Figures

4.1	HTML file with basic syntax and its output . . . . .	11
4.2	Styling the web page with CSS . . . . .	11
4.3	Working of function <code>addEventListener()</code> in the web page with HTML . . .	12
4.4	Basic DOM Manipulation . . . . .	14
4.5	Bable compiling the modern JavaScript to ES5 JavaScript code . . . . .	17
5.1	Project folder structures in VC code editor . . . . .	20
6.1	Home page of Algorithm Visualizer . . . . .	25
6.2	UI of Searching Visualizer . . . . .	25
6.3	Binary Search Visualise . . . . .	26
6.4	Sorting Visualizer Page . . . . .	27
6.5	Bubble Sort Visualization . . . . .	28
6.6	Path Finding Visualizer Web Page . . . . .	29
6.7	Path finding using Dijkstra's Algorithm . . . . .	30
6.8	BST Visualizer Page . . . . .	30
6.9	Insert elements in a BST . . . . .	31
6.10	Insert elements in a BST . . . . .	31
7.1	ER model of Algorithm Visualizer . . . . .	32

# List of Tables



# Chapter 1

## Introduction

When discussing complicated technical issues such as algorithms, it is critical for students to have a solid knowledge of the subject since it serves as the foundation for programming abilities and computational thinking. We had seen that traditional teaching approaches made it more difficult for pupils to grasp concepts and for professors to express their thoughts.

Algorithm visualisation approaches, according to researchers and educators, can help students understand algorithms more quickly and fully. This is due to their belief that images may convey more information than words. As a result, we created a learning technique that would assist both students and instructors by utilising visualisation and hands-on experience with various searching algorithms, sorting algorithms, and graph algorithms. By graphically portraying the many stages and animating the changes between those states, a good visualisation brings algorithms to life. The human visual system may be used to increase human intellect through visualisation. It may be utilised to better understand these critical conceptual processes as well as others.

E-learning is currently being heavily encouraged among students of many fields. Modern technologies enable the creation of visualisation tools for subjects such as various sorting and graph algorithms, as well as their explanations. One of the most crucial criteria for the successful usage of any e-learning system is the deployment of such e-learning tools. Learning via visualisation has been found to boost learning abilities. Increase your personal learning process's autonomy. The programme seeks to make algorithms easier to comprehend by offering a visual picture of how they discover target nodes. A decent algorithm visualisation tool brings algorithms to life by animating transitions from one node to another.

Finding the shortest path is a frequently utilised use of graph theory in many practical applications such as maps, road networks, and robotic navigation. To demonstrate how the tool works, we employ Dijkstra's algorithm. This is due to the fact that it also works for weighted graphs. As a result, it takes longer to execute than BFS. This method ensures the shortest path feasible. In addition, adopting online learning techniques instead of face-to-face events can boost learning by increasing student performance, happiness, and learning flexibility.

# Chapter 2

## Theoretical Background

This section goes into detail into the theoretical roots of the thesis's various elements. Among the topics discussed are search algorithms, sorting algorithms, graph algorithms, BST, and visualisations, as well as their definitions, history, and scope.

### 2.1 Algorithms

- Algorithms are a set of procedures that are used to achieve a certain task. Its purpose is to make it simple to repeat the action while remaining efficient and precise. The term "algorithm" is derived from Abdullah Muhammad bin Musa al-Khwarizmi, a 9th-century Persian scholar, astronomer, and mathematician renowned as "The Father of Algebra."
- Algorithms may be found in almost every aspect of daily life. Algorithms originated in computation and are today employed in sophisticated research and activities such as artificial intelligence, gaming, computer processing, molecular biology, and other scientific study domains. They can range from as basic as turning on and off a light switch to complicated mathematical and computational procedures. Its primary applications include computing, data processing, and automated reasoning. When quality and efficiency are the most crucial aspects to consider when performing a task, algorithms become increasingly significant. The inverse of following an algorithmic process is performing the same work in a different method each time. This diminishes dependability, efficiency, and increases time, energy, cost, and a variety of other issues. As a consequence of the application of algorithms, these barriers are removed, resulting in uniformity, efficiency which is evident.

- Algorithms and features are linked and frequently used interchangeably, but they are not the same thing. A function is the real execution of an algorithm, which is a collection of ideas or abstract notions that define how to solve a problem. Algorithms capable of solving them. As a result, a function can implement a full algorithm or one or more phases of an algorithm.

## 2.2 Searching Algorithm

Algorithms are designed to find or retrieve an element from any data structure where it is present. Based on the type of search operation, these algorithms generally fall into two categories:

### 2.2.1 Sequential search

Sequential search is a very simple search algorithm. In this searching, the array or list is traversed sequentially and each element is checked. Linear Search is an example of sequential search.

### 2.2.2 Interval Search

These algorithms are specially developed for searching an element in a sorted data structures. This type of search algorithm is much more efficient than linear search because it repeatedly targets the center of the search structure and divides the search space in half. Binary Search is an example of interval search.

## 2.3 Sorting Algorithm

Sorting is the process of putting records in ascending or descending order based on data item common features. Name, price, size, weight, length, efficiency, and so on are examples of properties. The primary goal of record sorting is to improve the time efficiency of discovering and obtaining the things you require. Sorting is a frequent, yet often neglected, activity that we encounter and employ in our everyday lives. Sorting in everyday life includes language dictionaries, phone books, all sports scorecards, social media posts, and rubbish sorting. These products employ sorting to assist you access data like dictionary definitions, phone numbers by name, points for your favourite teams, recycling,

and more fast and effortlessly. As stated in the preceding section, using algorithms to complete tasks is desirable for efficiency, consistency, and quality of output. Sorting is similar, and a sorting algorithm is an algorithm for sorting. There are several sorting algorithms, each with its own method and reasoning. They have varying efficiency and are better suited for certain data distributions. Based on their sorting process, they may be split into different types. Some methods are given below.

- Comparison-based/non-comparison-based sorting: Comparison-based sorting occurs when elements are compared to each other to discover a sorted array; otherwise, non-comparison-based sorting occurs.
- In-place/out-place sorting: If the sorting of the array does not use any additional memory, then the sorting is in-place sorting, otherwise it is out-place sorting.
- Offline/Online Sorting: A sorting is an online sorting if it allows new data during the ongoing sorting process and returns results regardless. Otherwise it's an offline sort.
- Stable/Unstable sorting: If sorting does not change the order of place of an element with the same value, this sorting is stable sorting otherwise, it is unstable sorting.

## 2.4 Binary Search Tree

A binary search tree arranges elements according to a specific order. In a binary search tree, the left node's value must be less than its parent node's value, and the right node's value must be greater than its parent node's value. This rule applies recursively to the left and right subtrees of the root. Finding an element in a binary search tree is trivial because you always have a clue as to which subtree contains the desired element. Insert and delete operations are faster in BST compared to arrays and linked lists.

## 2.5 Graph Algorithm

In programming, a graph is a non-linear unique data structure composed of a finite collection of nodes or vertices and a set of edges linking those vertices. At this stage, adjacent nodes might be referred to as nodes connected by the same edge. A graph, in simplest terms, is a visual depiction of vertices and edges that share links or connections. There

are several well-known graphing algorithms, but only a few are in use. The rationale is straightforward. Because traditional graphing methods are intended to answer millions of problems using only a few lines of rationally written engineering. A perfect algorithm is only optimised to deliver such efficient outcomes to some extent. There are several types of graph algorithms.

### 2.5.1 Breadth-First Search

One of the most typical actions done while working with graphs is searching or traversing. Thus, in breadth-first search (BFS), the algorithm attempts to visit all neighbours at the specified depth before moving on to the next level of vertex traversal. Graphs, unlike trees, can have circular routes. Surprisingly, the first and last vertices always match. As a result, BFS must maintain track of all traces of vertices visited. Use a first-in, first-out queue data structure to implement such ordering.

**Applications:** The BFS algorithm has several applications. It is used to identify shortest pathways and least spanning trees, for example. It is also used as a search engine for social networks, assisting you in finding peer-to-peer networks on BitTorrent.

### 2.5.2 Depth-first search

Depth-first search (DFS) begins at a vertex and travels as far as possible along all branches before returning. DFS must maintain track of visited nodes, and it does it using a stack data structure.

**Applications:** DFS is used to find cycles by finding the path between two vertices. Topological sorting can also be easily performed using the DFS algorithm. DFS is also used for puzzles with solutions.

### 2.5.3 Dijkstra's algorithm

The Dijkstra algorithm is a shortest-path-first algorithm using a single source. It aids in determining the shortest path from source to destination, such as Between two grid points that can depict road networks, robot motions, maps, and so forth. Algorithms come in a variety of flavours. The technique was originally created to discover the shortest path between a source node and a destination node, however contemporary variations of the current approach use a single node as the source and all other nodes as the destination. Consider it a destination and find the shortest path between the source and all input

nodes. The raster target for producing the shortest route tree. I also prefer less expensive methods. Dijkstra's shortest path algorithm stores nodes sorted by distance from the source in a data structure, often the lowest priority queue, although an array can also be utilised. It is one of the quickest shortest-path-first methods when only one source is available to discover the shortest path and any directed network with positive and unbounded weights. This approach is also applicable to undirected graphs.

**Working of Dijkstra's algorithm :** The starting node is called the initial node, and the algorithm assigns initial distance values to all other nodes and updates them incrementally.

- First, mark all nodes as unvisited and put them in a set.
- Assign a tentative distance value to the starting node as zero and all others as infinity. Sets the current node to show as the starting node.
- For the first node, compare the tentative distance of the unvisited neighboring node and the newly found value with the originally assigned value and replace it with the smaller value.
- If all unvisited nodes are considered current nodes, set the current node in the grid as visited node and pull from the set of unvisited nodes .
- When a destination node is found and marked visited (while finding a route between two specific points) or the minimum interim distance between source and all destinations in the unvisited set is infinite In a situation where there is no relationship (this happens when there is no relationship between the source node and the destination node in the unvisited set),stops. Otherwise pick unvisited node with the smallest tentative distance, consider it the current node, and go to step 3.

**Applications :** This algorithm has proven useful when there is only one source and multiple destinations. Dijkstra's algorithm helps create road networks and reduces road construction costs. Applicable to network routing protocols. Open the shortest route first in the middle and middle series. It can be applied to improve the movement of robot systems.

## 2.6 Measurement of Algorithm

Generally, Algorithms are measured for their efficiency in terms of the time it takes to perform a task as the input increases and the disk space or memory they use. There are various terms used to describe the efficiency of an algorithm which are Big O ( $O()$ ), Little O ( $o()$ ), Omega ( $\Omega()$ ) and Theta ( $\Theta()$ ). Big O ( $O()$ ) represents an upper bound that contains an exact bound on complexity. Little O ( $o()$ ) indicates an upper bound but excludes an exact upper bound.  $\Omega()$  represents the lower bound of complexity and  $\Theta()$  represents the exact limit of complexity. Best case is considered  $O(1)$ . This means that the algorithm can process larger inputs without losing time or space. For example, the JavaScript function `console.log()` has the complexity of  $O(1)$ . Its performance does not depend on the size of the input. However, complex computations such as sorting algorithms have different complexity. In terms of time complexity, the fastest sorting algorithm is quicksort, and in terms of space, insertion sort is the most efficient.

# Chapter 3

## Need Of Visualization and Objective

### 3.1 Visualization

- The visual portrayal of ideas, stories, facts, and so on is known as visualisation. Simply said, utilising visual cues to describe information is also a sort of visualisation. It is not necessary for visualised information or things to be actual or physical objects, but they might be abstract ideas such as a new thought, imagination, sentiment, or belief. Numbers, letters, art, statuary such as drawings, symbols, paintings, computer graphics, murals, and cinema, among other things, are forms of visualisation used to transmit information. information that is optional Anything visualised in an interesting and informative manner has increased communication potential, allowing fundamental concepts to be communicated much faster and simpler.
- Visualisation has been throughout human history from prehistory. As a result, visualisation is not a novel mode of communication conveyance. It has existed from the beginning of time. It has taken many various shapes and ways over time, but it still serves the same function. Today's world is dense with information and data, and it is becoming increasingly complex. Information must be supplied as quickly, consistently, and precisely as feasible. It developed contemporary visualisation in the form of data visualisation for research, studies, business, entertainment, publications, magazines, advertising, and other purposes. All strive to be more aesthetically attractive and informative than the competition. As a result, visualisation is critical and will remain so.



## 3.2 Objective

- The basic goal of the thesis is to develop a web application that will serve as a visualisation tool. A single-page online application designed with the most recent JavaScript technology that visualises the logic and processes of various searching, sorting, graph, and tree algorithms. The application's home page will have four options: searching, sorting, graph, and tree algorithms, from which the user may choose and navigate to the appropriate page.
- The UI of the search visualizer will contain options to select one of the searching algorithms which were implemented and various items or elements in the array, enter an element, search button to start and sound button which can be turn off/on.
- The sorting visualizer's user interface will have choices for selecting one of the implemented sorting algorithms and various items or components in the array, control buttons to start, move to previous or next phases, and an option for sorting speed and colour mode. A data array of the specified size is filled with unique values created at random. The data set is represented as vertical bars with the height of each value. When sorting begins, the UI will display the step-by-step organisation of data in ascending order depending on their height.
- Silimilarly The graph visualizer's user interface will include options to select one of the graph algorithms that were implemented, as well as various nodes in the form of a matrix, visualise buttons to start, and an option for clear grid and colour mode. Among all nodes, one will be the start node and one will be the end node, with the ability to create an obstacle node. When we press the start button after picking the algorithm, start node, finish node, and obstacle node, the stepwise process of this selected algorithm finding the path will be visualised in the UI. The final goal is to design the BST user interface. The page's navbar will now provide options for inserting and removing nodes in a BST.

# Chapter 4

## Web application as a Visualization tool

To further visualise the algorithm approach, we employed HTML, CSS, and JavaScript software tools. In JavaScript, DOM manipulation and timer routines have been modified. To emulate the detailed working mechanics of all the algorithms, event listeners, and separation concepts, JavaScript, CSS, and HTML are frequently utilised. They are, in fact, the building pieces of a sorting visualisation system. This section contains a list of all of the tools used in the project to create the visualisation tool as a web application. This section also contains a brief description of the tools, including their history, usage, and purpose.

### 4.1 HTML

HTML is an abbreviation for Hypertext Markup Language. A markup language is a computer language that uses tags as components to construct human-readable texts. HTML is one of the most widely used markup languages. It is used to create and describe online documents and web pages. HTML components or tags are used to structure a webpage. HTML may be created using any simple text editor. The HTML file's content is stored as a file with the extension ".html" or ".htm." The diagram below depicts the building of a simple HTML website. The file may be opened in a web browser, which will parse and display it on the webpage using an HTML interpreter. HTML has had five distinct versions since its inception. HTML5 is the most recent.

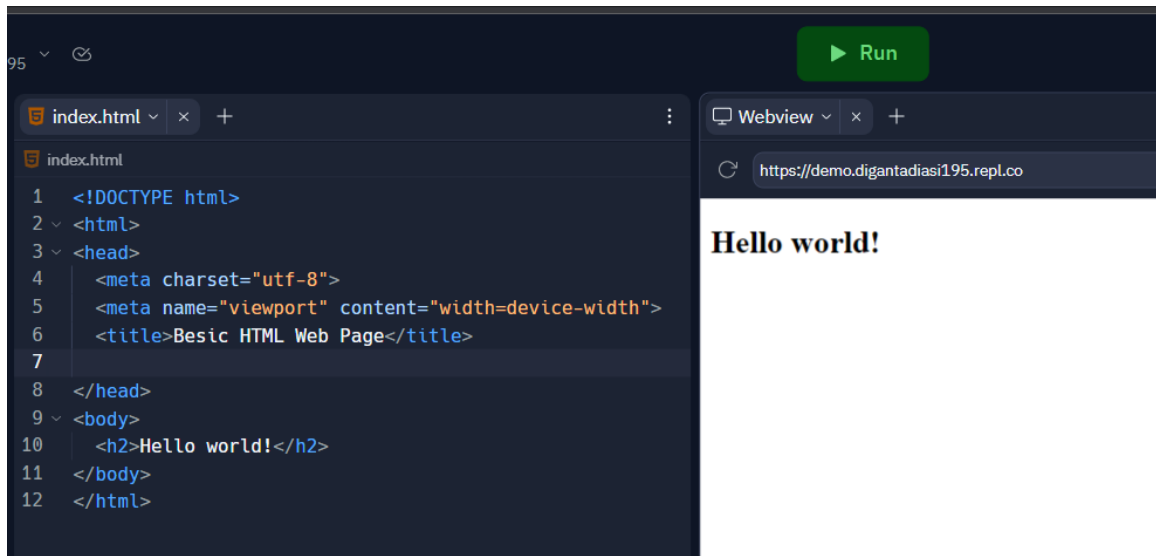


Figure 4.1: HTML file with basic syntax and its output

## 4.2 CSS

CSS is an abbreviation for Cascade Style Sheets. It is a programme that allows you to add presentation styles to your website, such as colours, layout, animations, and fonts. It allows web pages to adapt to different screen sizes on different devices. CSS is not dependent on HTML and may be used with any XML-based markup language. CSS employs a selector to target and customise HTML components. CSS properties can be put into an HTML page's tag or separated into a separate file with the ".css" extension.

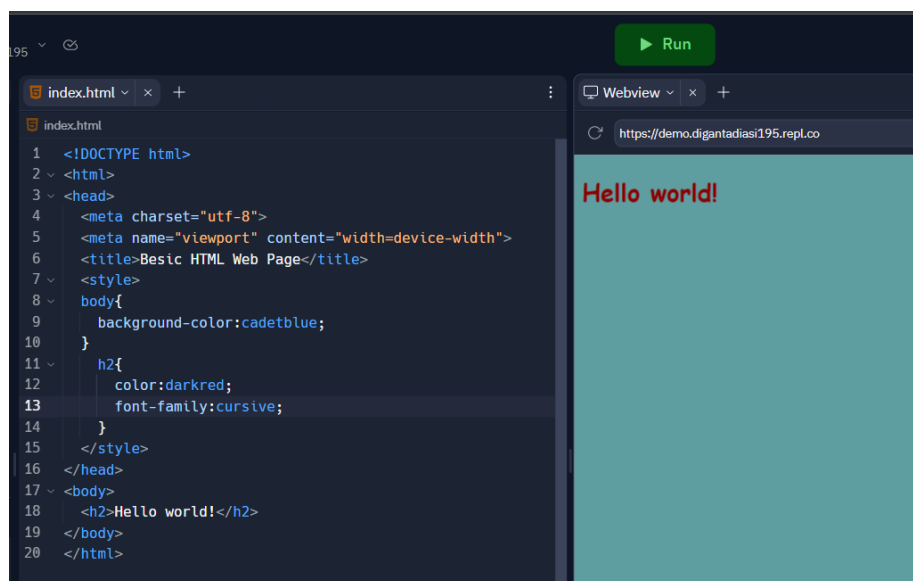


Figure 4.2: Styling the web page with CSS

## 4.3 JavaScript

JavaScript is one of the most popular programming languages today. The formal name for JavaScript, which is widely used, is ECMAScript. It is a programming language that may be interpreted or compiled at any point in time. Brendan Eich created it, and it was initially made available in 1995. It is presently standardised and maintained by ECMA International. ".js" is the file extension for JavaScript. The JavaScript programming language may be used to control the behaviour of numerous components. JavaScript Event Listeners: For any event that is just happened we can do two things: 1. Listen to the events 2. Reacting to Events

1. **Listen to the events** I used the `addEventListener()` function to listen for huge events that keep happening. The `addEventListener()` function is the sole responsibility of listening to functions and sending alerts or useful information to other parts of the application within milliseconds.

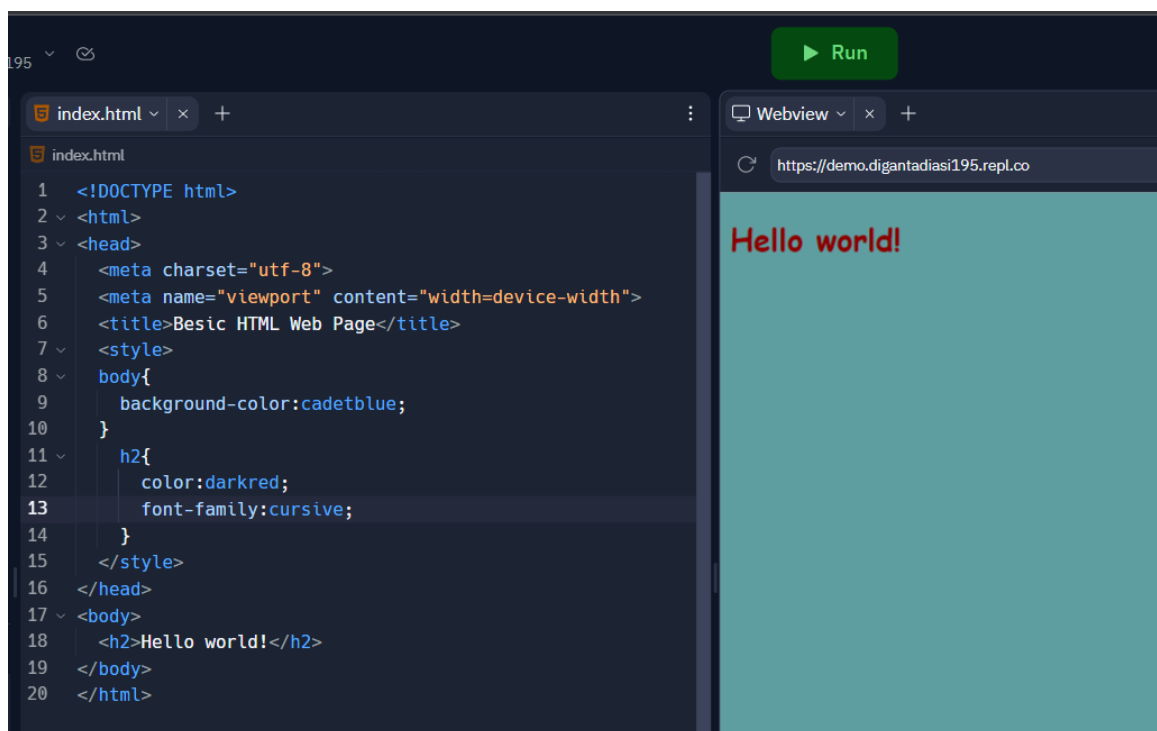


Figure 4.3: Working of function `addEventListener()` in the web page with HTML

Syntax: `source.addEventListener(eventName, eventHandler, false)`

**Description :** The object on which `addEventListener` should listen for events is the source. This can be a DOM element, such as a page, window, or any object intended expressly to catch events. Event Handler: A function that runs when

an event is detected. The Event Handler: A method which is executed when listening to an event. Example: `document.addEventListener("click", changeColor, false)` In the example above, the document is the object to which we added the `addEventListener` event listener. The first parameter of `addEventListener` is `click`. This is a mouse click event on the document object. `changeColor` is an event handler that changes the color of the document when Mouse click event occurs.

2. **Reacting to Events** Then, while listening for the event that occurred on the document, `addEventListener` runs the event handler. An event handler is a function that takes a desired action in response to a preset event. The example code below has event handlers and event listeners.

```
Used: document.addEventListener("click", colorChange, false);
function colorChange(event){
  Color is change!!
}
```

In the small code above, `addEventListener` is the event listener and `colorChange` is the event handler. An event handler function writes a block of statements that perform the desired action.

## Basic DOM manipulations :

The HTML document's member function, `querySelector()`, retrieves the initial element in the document that corresponds to a specific selector. In case of no match, the function returns a null value.

Syntax: `var element = document.querySelector(selectors);`

Like `querySelector()`, `querySelectorAll()` returns a constant `NodeList`. This `NodeList` contains a list of the document's elements that match the specified selector.

Syntax: `var elementList = document.querySelectorAll(selectors);`

When the above statement is executed, a static list of document elements is stored in the element list. If you're only interested in retrieving a specific element with a known ID, you can use the document's `getElementById()` mem-

ber function. Example: `var element = document.getElementById('myid')`.

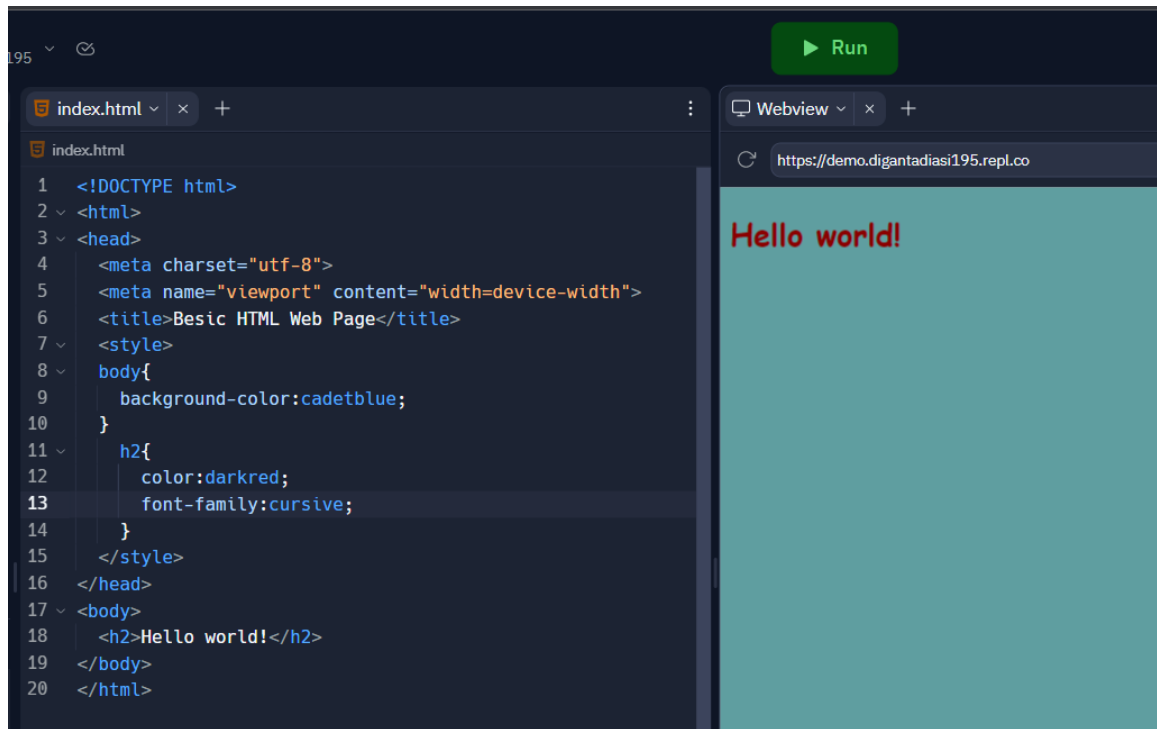


Figure 4.4: Basic DOM Manipulation

The `getElementsByTagName()` function returns an array type of object containing the elements on the page of a particular type. For example `<p>` paragraph `</p>`, `<a>` anchor element `</a>` etc. The type of element is supplied as a parameter to the function. For good understanding consider the below statement,

i.e. `const elementArr = document.getElementsByTagName('p')`.

If the tag name is not recognized, `createElement()` can be used to create the HTML element mentioned by the tag name or unknown HTML element.

Syntax: `const element = document.createElement(tagName[ , options]);`

The statement describes the use of the function `createElement()` in creating new elements in web development. The function takes a `tagName` parameter, which is a string that specifies the type of element to create. However, it's important to note that using qualified names like `"html:a"`

is not recommended. Additionally, the function converts the `tagName` to lowercase before creating the element. In some web browsers, using `createElement(null)` is equivalent to `createElement("null")`.

The second optional parameter for `createElement()` is an object that can have a single property `name` with the value of a custom element tag name defined using `customElements.define()`.

The `createTextNode()` function, on the other hand, creates a new text node and can be used to escape HTML characters.

Syntax: `let textdoc = document.createTextNode(data);`

All browsers support the `window` object, which typically represents the browser's window. The `window` object includes variables, functions, and global objects like `JavaScript` as its members. The `window` object's global variables are its typical properties, while a global function is a member of the `window` object. Additionally, the HTML Document Object Model is a `window`'s document object.

The below statement 1 and statement 2 are same.

Example: `window.document.getElementById('header');` ————1

`document.getElementById('header');` ————2

The `"innerHTML"` property sets/returns the HTML content of an element,

Example: `HTMLElementObject.innerHTML = text`

The `classList` property returns the element's class name as a `DOMTokenListObject`. You can use this to add, toggle and remove the CSS class of an element. Class list properties are read-only, but can be modified using the

`add()` and `remove()` methods.

Syntax: `const arr = element.classList`

You can use the `disabled` property to set or reset the dropdown list. When an item is disabled, you can no longer use it to click. By default, disabled items are grayed out in the browser. To get the `disabled` property you have to write the statement like `selectObject.disabled`

To set or deactivate the selected object, you have to write a statement like `selectObject.disabled = true|false`

You can disable the selected object by setting the `disabled` property to a value of `true`. Similarly, you can enable a property by setting the value `false` to the `disabled` property.

**JavaScript Timers :** To run a function at a specific time you can use a timer function. For example, if you want to delay code execution, here that means you don't want it to run when an event is triggered or when the page loads. In JavaScript, we have two types of timer functions,

#### 1. `setTimeout()` function

We can use the `setTimeout()` function to execute a task or operation immediately after a specified time interval elapses.

Syntax : `setTimeout(function, milliseconds)`

This function accepts two parameters, function and time in milliseconds. A function is just a piece of code that you want to execute, and the time in milliseconds represent how long it takes to execute code.

#### 2. `setInterval()` function

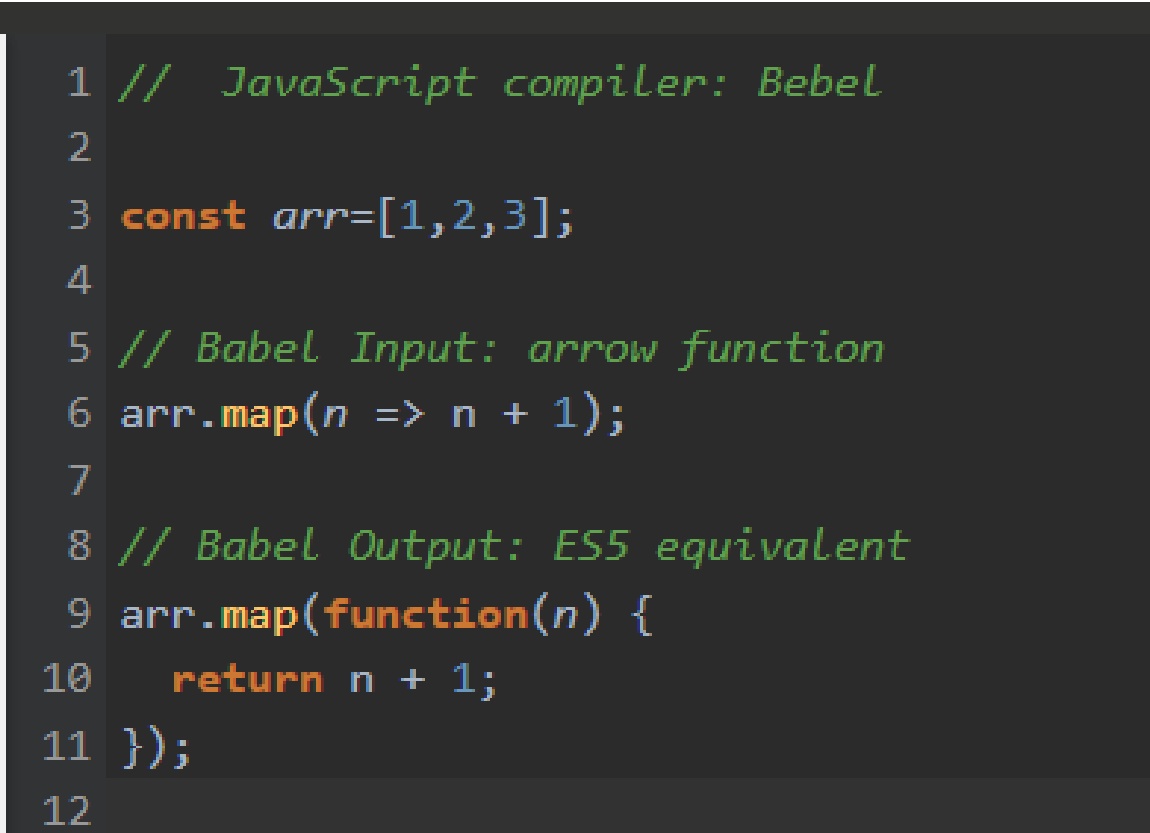
To run code periodically, we can use `setInterval()`. The `setInterval` method requires two parameters: a function and milliseconds.

Syntex: `setInterval(function, milliseconds)`



## 4.4 Babel

Babel is a popular JavaScript compiler that allows developers to write modern JavaScript code using the latest syntax and features, and then convert it into a backward-compatible version of JavaScript that can be executed in older environments. Babel is commonly used in modern JavaScript development workflows to ensure that code written using the latest language features can be used in older browsers or environments that may not support those features. Babel supports various ECMAScript (ES) versions, including ES6, ES7, ES8, ES9, ES10, ES11, ES12, and beyond. It also provides support for other JavaScript syntax, such as JSX, which is commonly used with popular libraries and frameworks like React. Babel has a large and active community, and it is widely used in many JavaScript projects to write modern, forward-compatible code that can be used in a wide range of environments.



```
1 // JavaScript compiler: Babel
2
3 const arr=[1,2,3];
4
5 // Babel Input: arrow function
6 arr.map(n => n + 1);
7
8 // Babel Output: ES5 equivalent
9 arr.map(function(n) {
10     return n + 1;
11 });
12
```

Figure 4.5: Babel compiling the modern JavaScript to ES5 JavaScript code

## 4.5 GitHub and Git

GitHub and Git are related but distinct tools used in software development for version control and collaboration. Git is a distributed version control system (VCS) that helps developers track changes to their source code. It allows multiple developers to work on the same project simultaneously, and provides a history of all changes made to the codebase over time. GitHub is a web-based hosting service for Git repositories. It provides a user-friendly interface for managing Git repositories, making it easier to collaborate on code with other developers. GitHub offers features such as code hosting, issue tracking, pull requests, and team collaboration tools. It also provides integrations with other development tools, such as continuous integration and deployment (CI/CD) services, project management tools, and code review tools.

# Chapter 5

## Implementation of Project

This section describes the project implementation, coding practices, and project vision and project outcomes in detail.

### 5.1 Building the application

The project's goal was to provide a visualisation platform for visualising various algorithms in the form of web application type algorithms. Although there are various possibilities for developing web apps, the technologies indicated in the preceding section were utilised to complete that project. A good project structure is required to achieve scalability. As a result, the project began by laying a solid foundation that incorporates all of the tools required to manage the project properly. This solution was painstakingly created, use tools such as Babel to translate contemporary JavaScript code into browser-compatible versions, and Webpack to bundle the code into a single file for optimised speed. Git is used to maintain version control, while ESLint is used to verify code quality for syntactical coherence. A well-organized organisational system has been constructed, with different folders assigned for different tasks. The folder structure is given below. Once the foundation is in place, a GitHub repository is formed and the files are pushed to it. This base project may be used as a template for developing many React apps, reducing the number of repeated procedures often associated with starting new projects.

Following the creation of the template, the project implementation proceeded, utilising the Hybrid development technique and the requirements gathered during the Requirement analysis phase. The features were broken down into smaller tasks, and their progress was recorded using the Trello programme. This provided an up-to-date perspective of the application's progress, which aided in maintaining a constant and continuous devel-

opment pace. Although this strategy is time-consuming for small teams, it was effective in organising and steering the project on a broader scale, including several team members across various elements of the project.

## 5.2 Structure of the project

In every software development project, the structure and organisation of files and folders are critical for scalability and maintainability. This feature, which is sometimes disregarded, should be prioritised throughout all phases of software development. A well-planned project structure makes it easier and faster for new team members to become acquainted with the project. Maintaining logical order in the project files becomes increasingly important as the programme expands in order to manage components, utility methods, styles, and other features consistently. A modular and scalable project structure also assists in the identification and debugging of application faults. Different methodologies are used in various application development standards, and there is no one best option. The project structure chosen is also determined by the project kind and needs.

In this thesis project, a screenshot (Figure 5.1) is provided along with a table that showcases the role of each folder in the project structure. The folders are named in a way that describes their purpose and contents.

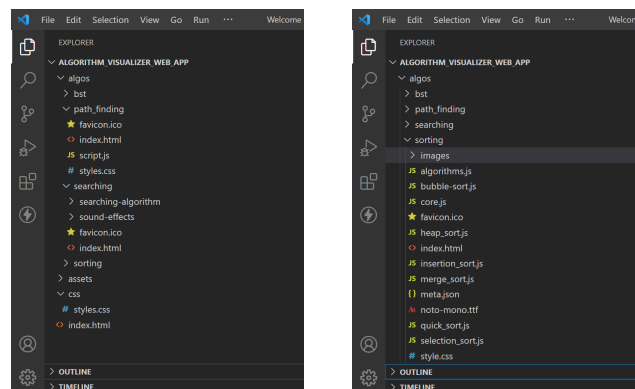


Figure 5.1: Project folder structures in VC code editor

## 5.3 Code

To implement this project I have create many file which is shown in previous section among then main file is "Index.html" . I have attached all the remaining file with this

main file and I have upload in github which is I have describe in next section.

The main HTML file code is attached below:

```
<!DOCTYPE html>

<html>

<head>

<title>Algorithm Visualizer</title>

<link rel="stylesheet" href="css/styles.css" />

<link rel="icon" href="assets/favicon.ico">

</head>

<body>

<div id="main">

<div id="header">

<a href="/index.html"> <h1><div id="title">Welcome to Visual Platform</div></h1>

</a>

</div>

<h2 id="demo">Algorithm Visualizer</h2>

<div id="container">

<div class="algo_container" >

< ahref = "algos/searching/index.html" class = "categorylink" >

< divclass = "image_container" >

< imgsrc = "assets/search.png" class = "image" / >

< /div >

< div > SearchingVisualizer < /div >

< /a >

< /div >

< divclass = "algo_container" >

< ahref = "algos/sorting/index.html" class = "categorylink" >

< divclass = "image_container" >

< imgsrc = "assets/sorting.png" class = "image" / >

< /div >

< div > SortingVisualizer < /div >

< /a >

< /div >

< divclass = "algo_container" >
```

```

< ahref = "algos/path_finding/index.html" class = "categorylink" >
< divclass = "image_container" >
< imgsrc = "assets/path_finding.png" class = "image" / >
< /div >
< div > PathFindingVisualizer < /div >
< /a >
< /div >
< divclass = "algo_container" >
< ahref = "algos/bst/index.html" class = "categorylink" >
< divclass = "image_container" >
< imgsrc = "assets/bst.png" class = "image" / >
< /div >
< div > BinarySearchTree < /div >
< /a >
< /div >

</div>
<footer> <div class="footer_text" >
< p >
Madewith < spanclass = "love" > love < /span > by
< ahref = "https : //www.linkedin.com/in/digantaitkgp" > DigantaDiasi < /a ><
br >
< /p >
< /div >
< /footer >
< /div >
< /body >
< /html >

```

## 5.4 Hosting

Hosting is the last step in web application development, and there are several popular providers such as Amazon Web Services, Google Cloud Platform, and Azure that offer web hosting solutions. These platforms offer many additional features beyond hosting and are affordable. Netlify, a web hosting service, was used for this thesis project because it is a user-friendly, fast, and uncomplicated hosting option that provides free web hosting. It also has a command-line interface for deploying applications and offers custom domain names with the "netlify.app" suffix.

The web application is currently being hosted on Netlify and can be accessed via the URL <https://joyful-axolotl-96df38.netlify.app>

Also we can host a website on GitHub. GitHub provides a free service called GitHub Pages that allows you to host static websites directly from your GitHub repositories. I have uploaded all the coding files in github and also hosted it through github. Any one can see the code implementation of the web application project go through the link <https://github.com/digantadiasi195/algovisual>

# Chapter 6

## Project Outcomes

At the end of the thesis, and with numerous iteration a web application visualizing all the algorithms was successfully created using JavaScript functionalities. In this web application there are five web pages and which are given below:

1. Welcome page
2. Searching Visualizer page
3. Sorting Visualizer page
4. Graph Visualizer page
5. BST Visualizer page

Now we will be describes the features of all the web pages

### 6.1 Welcome Page

This is the home page of my application and which has four option which are Searching Visualizer, Sorting Visualizer, Path Finding Visualizer and Binary Search Tree. User can select any of the page and can go to the respective page. User can traverse from home page to any of the visualizer page or this particular page to home page. The Welcome page of the application is as in Figure 6.1 below.





Figure 6.1: Home page of Algorithm Visualizer

## 6.2 Searching Visualizer

After clicking searching visualizer option of home page , user can go to this page. Then user will see the below UI

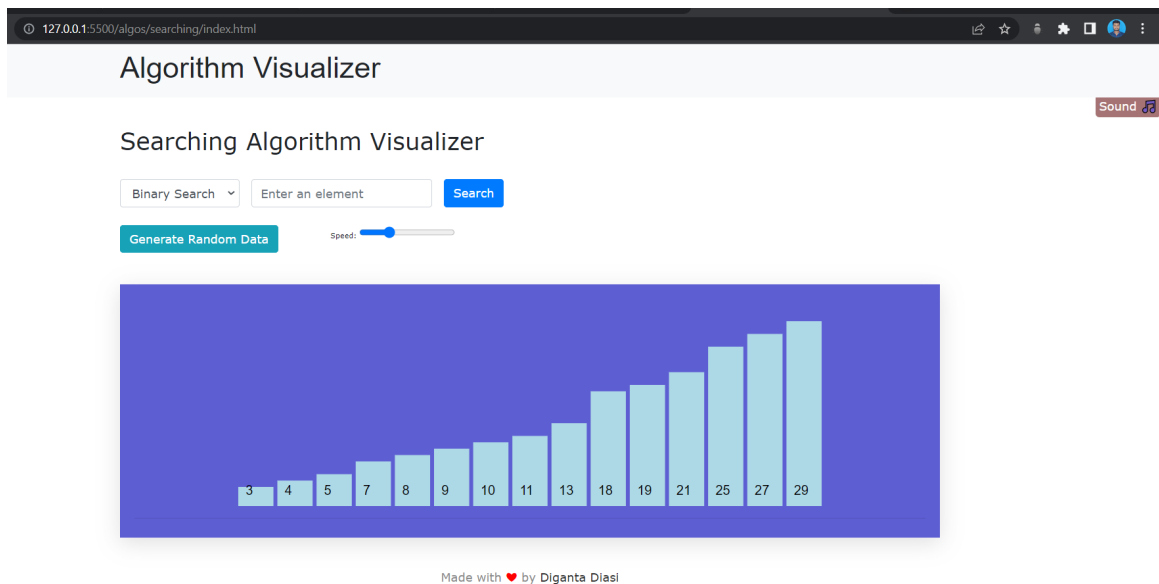


Figure 6.2: UI of Searching Visualizer

This page contain many option which are given below

- **Generate random array :** Each click on the tab will trigger the generation of a fresh random array, with array elements displayed as bars. The height of

each bar will be proportional to the numerical value it represents, creating a visual representation of the array.

- **Enter an Element and search :** In order to comprehend linear search or binary search, we need to input an element. Next, we need to click the search button. If the element is present in the array, the search will reveal the position where the element exists. Otherwise, a message will indicate that the element does not exist.
- **Sound :** Sound option present in the navbar of this page and User can use it.
- **Algorithms :** In this page there are two algorithm which are Linear Search and Binary Search. Using binary search if we search 26 in an sorted array , then for that search step by step procedure shown through the figure 6.3 below

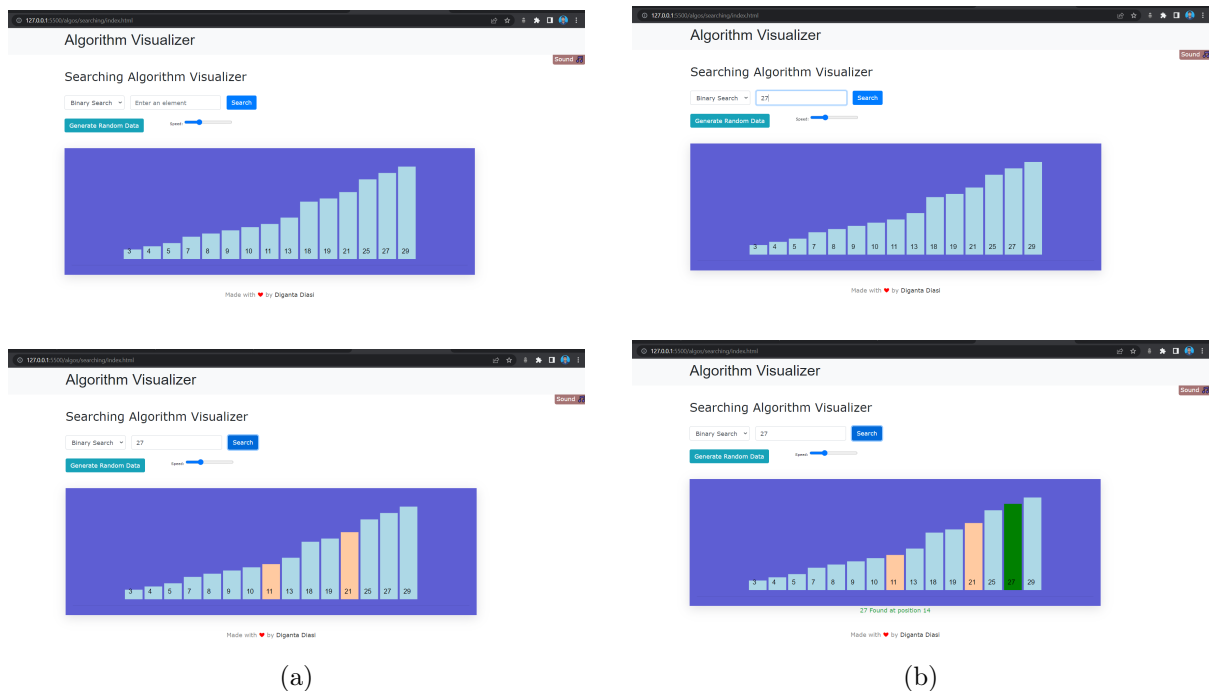


Figure 6.3: Binary Search Visualise

## 6.3 Sorting Algorithm

The navbar of sorting algorithm consists of the following options:-

- **Generate random array :** A new random array will be generated every time we click on this tab. The array elements will be presented as bars, with the height of each bar corresponding to its numerical value. During the sorting process, bars of

different colors will be used to distinguish between sorted, unsorted, and currently sorting values from the input array.

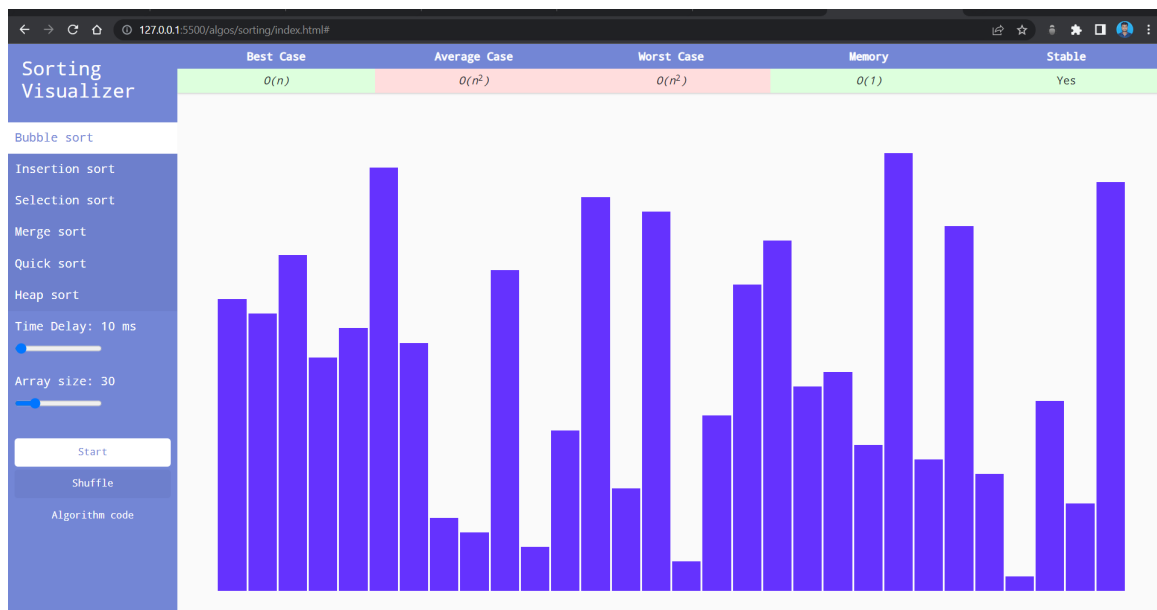


Figure 6.4: Sorting Visualizer Page

- Change array size and sorting speed** Users will have access to a slider that allows them to adjust the size of the array, which in turn affects the speed of sorting. The speed of sorting is directly proportional to the size of the array, meaning that a larger array size will result in faster sorting. This feature has been implemented to enable users to learn at their own pace without feeling rushed.
- Algorithms** Sorting visualizer page contain six sorting algorithm which are Bubble Sort, Insertion Sort, Selection Sort, Merge Sort, Quick Sort and Heap Sort. The user can select any algorithm from the list of available algorithms above. Algorithms are well-chosen based on their popularity and importance to the existing curriculum.
- Design :** To ensure clear visualization, we utilized various colors to distinguish between sorted and unsorted bars, as well as bars undergoing the comparison and sorting process. Once sorting is complete, the colors of the bars will be changed to the same color, distinct from the original colors of the array, and the array elements will be arranged in ascending order. user can select any algorithm which are present in left side of this page . If user selecting any algorithm , then user will see the time complexity, space complexity and stability of the algorithm at the navbar. Then user can fixed array by generating array and there is a option to start visualization. After clicking start button user will see the visualization of

that particular algorithm. The picture below(Figure 6.5) shows the step of bubble sort and user can see the other algorithms visualization by clicking that particular algorithm.

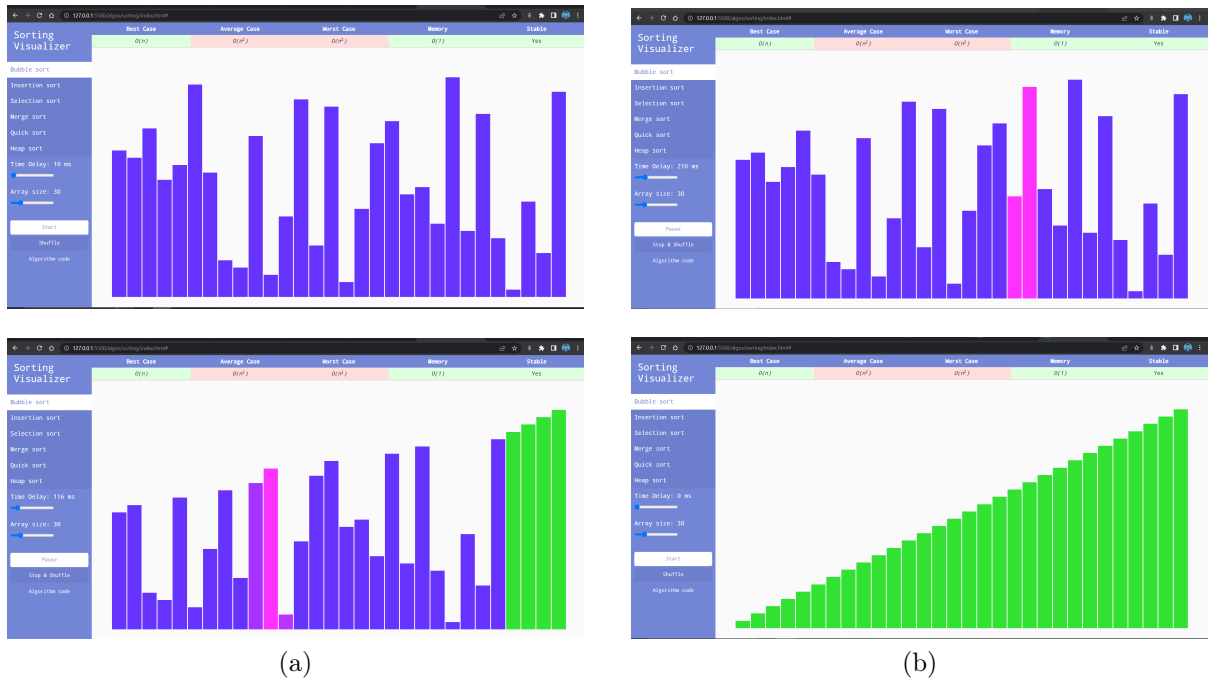


Figure 6.5: Bubble Sort Visualization

## 6.4 Path-finding Algorithm :

The navbar of path-finding algorithm consists of the following options:-

- **Algorithms :** This pages contains three algorithm which are Dijkstra's Algorithm, Depth First Search and Breath First Search. Dijkstra's Algorithm is most popular to shortest path.
- **Maze and Pattern :** To enhance comprehension of algorithms, mazes and patterns are incorporated. These visual aids simulate real-world scenarios by introducing obstacles or walls between the starting and goal nodes. Moreover, users can determine the efficiency of algorithms by analyzing their time complexity. These amusing options serve as a suitable means of grasping complex topics, especially for those seeking a more playful approach.



Figure 6.6: Path Finding Visualizer Web Page

- **Design and Architecture** The navbar features algorithms selected based on their popularity and level of difficulty. However, students often struggle to comprehend these algorithms in theory alone. By providing visualizations of these algorithms, students can gain a better understanding. The visualization allows users to distinguish between the functionalities of different algorithms based on their time complexity.

Each node will be represented using a grid structure. The computer will generate the starting and ending nodes which will be displayed initially. The user will have the ability to modify the positions of the start and end nodes according to their preference. The structure of mazes and their patterns can also be adjusted based on the user's desires, including the pattern of adding new walls.

In the navbar there are option to select the graph algorithm to find path and also the navbar contain various color which represent state of the visualization. Green color represent start node, Red color represent the end node and the black represent obstacle and yellow color represent visited node. User can fixed the start node, end node and for better visualization, user can add wall between two node. After clicking the algorithm , finally user will see the working of the algorithm. The picture below(Figue 6.7) shows how the Dijkstra's algorithm finds shortest path between two node.

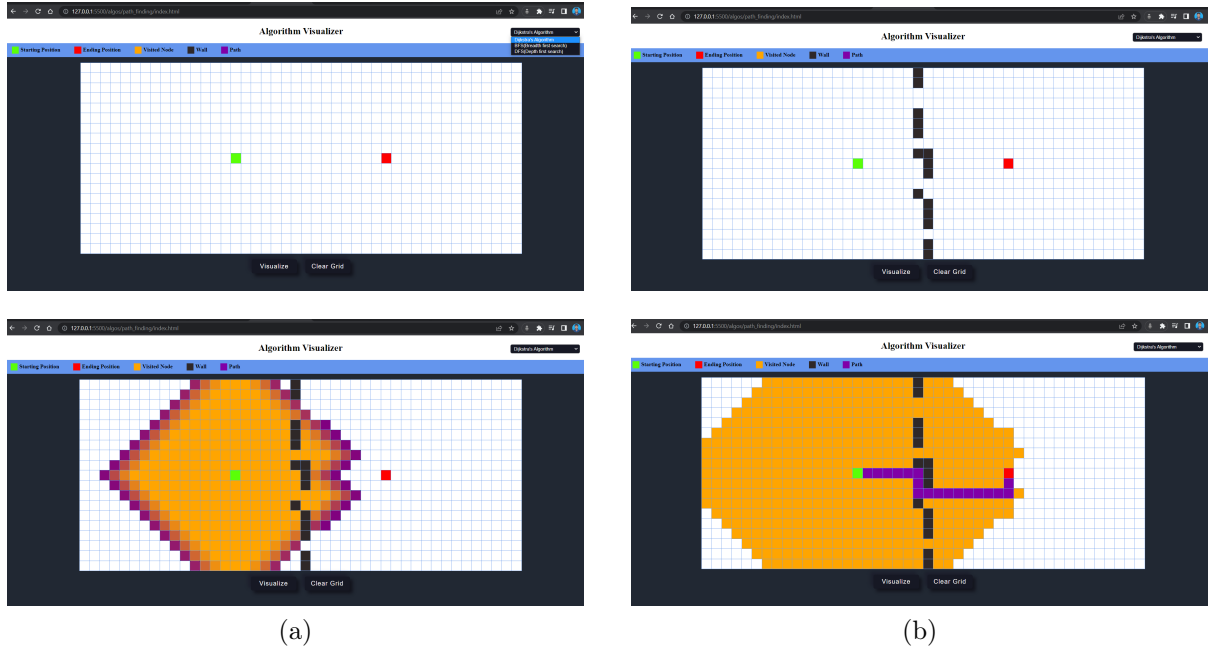


Figure 6.7: Path finding using Dijkstra's Algorithm

## 6.5 Binary Search Tree

The UI of this page is as in figure 6.8 below

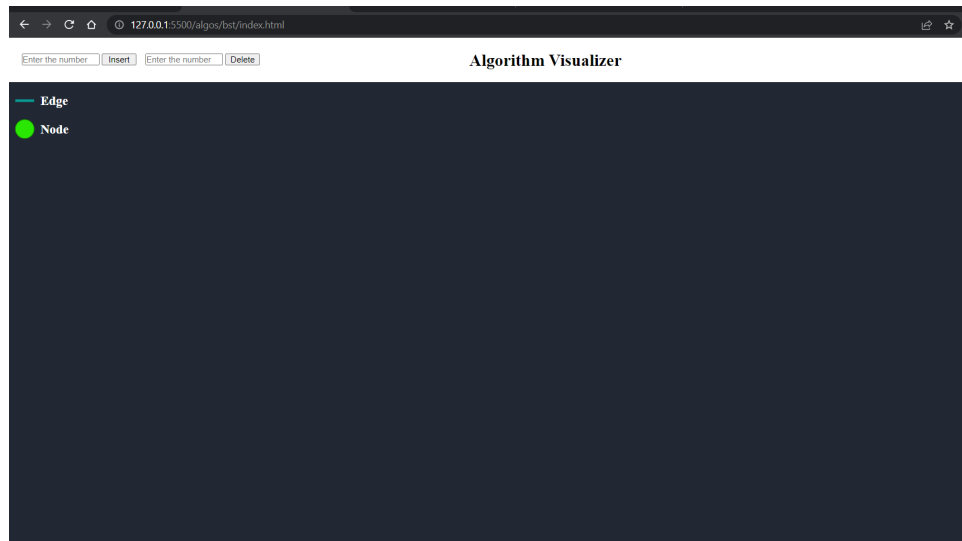


Figure 6.8: BST Visualizer Page

The navbar of binary search tree algorithm page consists of the following options:-

- **Enter an element and Insert :** There is options in navbar to enter an element in BST and after entering element user can click "Insert" button then a node will be create. If it is the first node of the BST, then this node will be root node otherwise this inserted node will be linked with the BST by maintaining its rule. The picture below(Figure 6.9) shows how new node added to the existing node

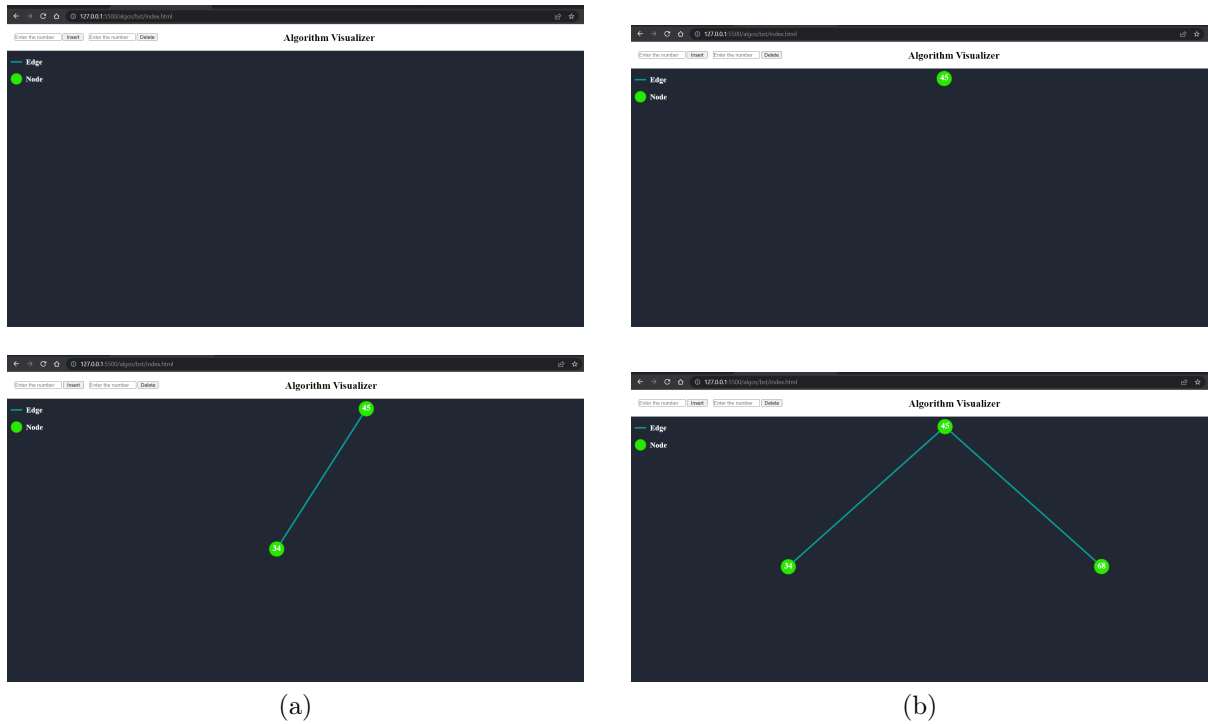


Figure 6.9: Insert elements in a BST

- **Enter an element and Delete :** There is a option in the navbar to delete an element from BST. The element which you want to delete , if it is not present in BST then there will be no effect on the BST otherwise this entering node will be deleted according to properties of BST. The picture below(Figure 6.10) shows how deletion is working when we delete 58 from the bST which are shown in the picture..

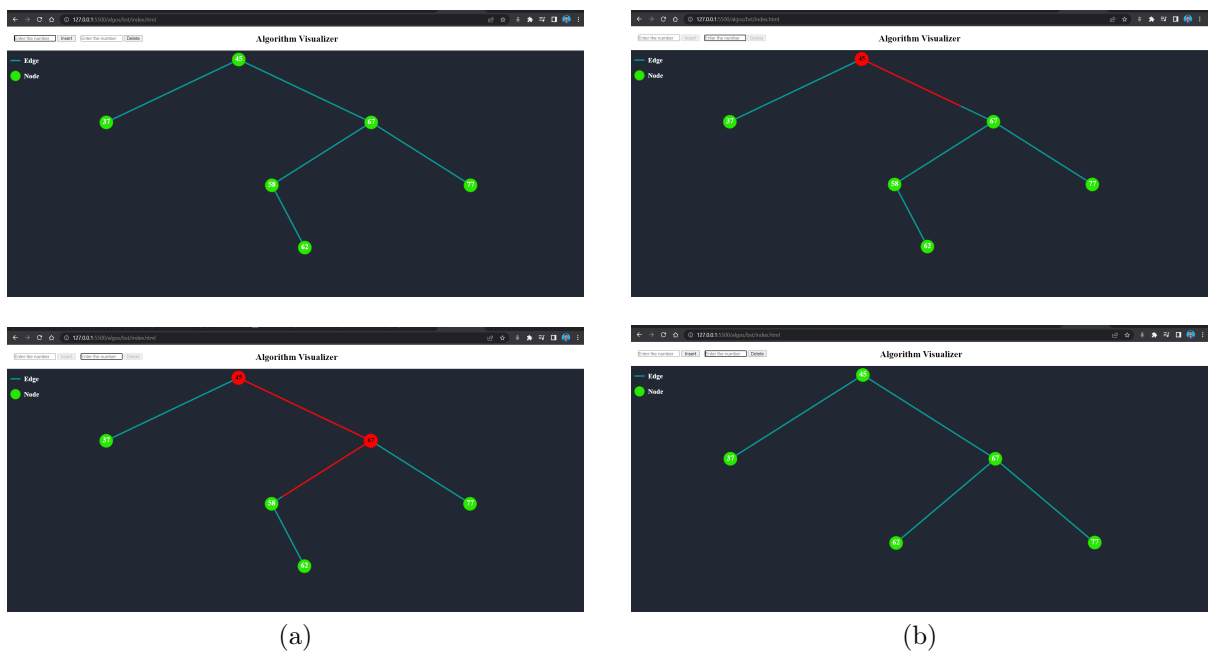


Figure 6.10: Insert elements in a BST

# Chapter 7

## Modeling Analysis

This project implementation is based on the ER model below.

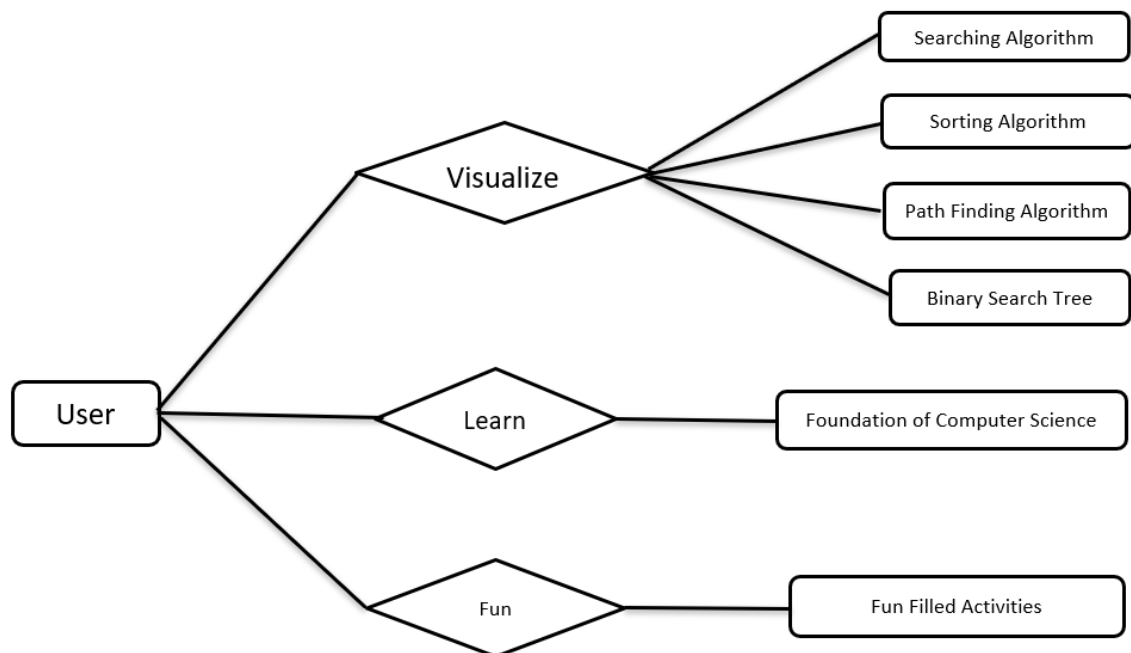


Figure 7.1: ER model of Algorithm Visualizer

It's nice to read that you prioritised user experience when developing your app and that you picked JavaScript as your foundation language because of its lightweight nature and wide range of frameworks. It's also encouraging to hear that you experimented with several JavaScript frameworks before settling on React.js due to its reusability, simplicity of testing and debugging, and component-based approach.

In terms of application structure, it is critical to have a well-defined architecture in place to guarantee that the application is well-organized and easy to maintain. It's fantastic that you researched current designs and picked an architecture that best meets



your requirements. As long as the design is scalable, modular, and simple to comprehend, it should help your application perform better.

While having a strong architecture is vital, it is equally critical to constantly monitor and optimise the performance of your application. Keep a watch on metrics like load times, user engagement, and error rates, and be ready to make adjustments to improve the user experience as needed.

# Chapter 8

## Conclusions

The conclusion of a project on an algorithm visualization web application would typically summarize the key findings and insights gained from the development and evaluation of the application. Here are some potential conclusions that could be drawn:

1. **Improved Understanding of Algorithms:** The algorithm visualization web application has proven to be an effective tool for improving the understanding of complex algorithms. Through interactive visualizations, users are able to gain insights into how algorithms work, their steps, and their behavior in different scenarios. This can help users better understand the inner workings of algorithms and how they solve problems.
2. **Enhanced Learning Experience:** The web application has the potential to enhance the learning experience for students studying computer science or related fields. By providing a visual representation of algorithms, students can grasp difficult concepts more easily and quickly. The ability to experiment with different inputs and observe how algorithms respond in real-time can aid in the comprehension of algorithmic concepts.
3. **Potential for Further Development:** The project on the algorithm visualization web application may have identified areas for further improvement and future development. For example, additional features, such as support for more algorithms, improved user interface, and mobile responsiveness, could be considered in future iterations of the web application to enhance its usability and effectiveness.
4. **Practical Applications:** The algorithm visualization web application could have practical applications in various fields, such as computer science education, software development, and data analysis. The insights gained from the project may have identified potential use cases and applications of the web application in real-world scenarios.

## REFERENCES

- [1] <https://www.geeksforgeeks.org/fundamentals-of-algorithms/1>
- [2] “Algorithm Visualizer” by Barnini Goswami, Anushka Dhar, Akash Gupta, Antriksh Gupta, Volume 3 Issue 03 March 2021, International Research Journal of Mordernization in Engineering Technology and Science.
- [3] “E-learning Tool for Visualization of Shortest Paths Algorithms” by Daniela Borissova and Ivan Mustakerov, ResearchGate, July 2015.
- [4] “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.
- [5] C.D. Hundhausen, S.A. Douglas, J.T. Stasko, A MetaStudy of Algorithm Visualization Effectiveness, J. Visual Lang. Comput. 13, 259–290, 2002
- [6] Alexander, S. (2001), “e-Learning developments and experiences”, Education and Training, Vol. 43 Nos 4/5, pp. 240-8
- [7] Sorting Out Sorting, Ronald M. Baecker with the assistance of David Sherman, 30 minute color sound film, Dynamic Graphics Project, University of Toronto, 1981. Excerpted and reprinted in SIGGRAPH Video Review 7, 1983. Distributed by Morgan Kaufmann, Publishers.
- [8] “ViSA: Visualization of Sorting Algorithms” by Tihomir Orehovački, ResearchGate, May 2012.