A Project Based Learning Report

On

Binary Tree Visualization

Submitted to CMR Engineering College

In Partial Fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

ASHISH SINGH	**(208R1A0505)**
SAI KRISHNA	**(208R1A0514)**
VINAYKUMAR	**(208R1A0517)**
VISHAL	**(208R1A0532)**
NITHIN REDDY	**(208R1A0536)**
SHIVA	**(208R1A0553)**

Under the Esteemed guidance of

Mr. S. Kiran Kumar

Assistant Professor, Department of CSE

Department of Computer Science & Engineering

CMR ENGINEERING COLLEGE

(Accredited by NBA, Approved by AICTE, NEW DELHI, Affiliated to JNTU, Hyderabad)
Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401)

2022-2023

CMR ENGINEERING COLLEGE

(Accredited by NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)
Kandlakoya, Medchal Road, Hyderabad-501 401

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project entitled "SMART CITY SERVICES" is a bonafide work carried out by

ASHISH SINGH	**(208R1A0505)**
SAI KRISHNA	**(208R1A0514)**
VINAYKUMAR	**(208R1A0517)**
VISHAL	**(208R1A0532)**
NITHIN REDDY	**(208R1A0536)**
SHIVA	**(208R1A0553)**

In partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING** from CMR Engineering College affiliated to JNTU Hyderabad under our guidance and supervision.

The results presented in this project have been verified and are found to be satisfactory. The results embodied in this project have not been submitted to any other university for the award of any other degree or diploma.

Internal Guide
Mr. S. Kiran Kumar

Assistant Professors,
Department of CSE,
CMREC, Hyderabad

Head of the Department
Dr. Sheo Kumar

Professor & HOD
Department of CSE,
CMREC, Hyderabad

ACKNOWLEDGMENT

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Prof. Dr. Sheo Kumar** HOD, **Department of CSE, CMR Engineering College** for their constant support.

We are extremely thankful to **Mr. S. Kiran Kumar**, Assistant Professor, Internal Guide, Department of CSE, for his constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if we do not acknowledge with gratitude thanks to the authors of the references and other literature referred to in this Project.

We express our thanks to all staff members and friends for all help and coordination extended in bringing out this Project successfully in time.

Finally, we are very much thankful to our parents who guided us in every step.

ASHISH SINGH	**(208R1A0505)**
SAI KRISHNA	**(208R1A0514)**
VINAYKUMAR	**(208R1A0517)**
VISHAL	**(208R1A0532)**
NITHIN REDDY	**(208R1A0536)**
SHIVA	**(208R1A0553)**

CONTENTS

ABSTRACT

We introduce an interactive graph visualization scheme that allows users to explore graphs by viewing them as a sequence of spanning trees, rather than the entire graph all at once. The user determines which spanning trees are displayed by selecting a vertex from the graph to be the root. Our main contributions are a graph drawing algorithm that generates meaningful representations of graphs using extracted spanning trees, and a graph animation algorithm for creating smooth, continuous transitions between graph drawings. We conduct experiments to measure how well our algorithms visualize graphs and compare them to another visualization scheme.

LIST OF FIGURES

1. INTRODUCTION

1.1 About the Project

A binary tree is a data type where every node in the graph can have at most two children. The node to the left of any parent node must contain a value less than the value of the parent node, and the node to the right must contain a value greater than the value of the parent node.

Many real world problems can be modeled by graphs. By using graph visualization [34, 39, 71] to represent entities and relationships in these problems, one can see patterns that may be hard to detect none visually [45, 47, 67]. This is because humans are highly attuned to extracting information from a visual representation of a graph [5]. Graph visualization is important in many research areas of computer science, including software engineering [13, 14, 60], database design [1, 12], and networking [2, 3, 9, 26].

Something to consider when referring to the project is the way it has been developed and also the presentation of the information like the various diagrams and system analysis. These details make it not only a good project for the future but also provide a very good if not perfect example to other individuals performing research on similar topics or similar types of projects. The explanations and documentation provided will definitely meet the requirements of most of these researchers and viewers. The basic presentation of the information provided may be more preferable to most people in accordance with time efficient research.

1.2 Purpose

A major concern in graph visualization is how to create meaningful and useful representations of graphs automatically.

1.3 Proposed System

The web application being developed is unique on its own and more reliable for its subjects being interacted with. This project as the name suggests deals with an interface between the service providers and the customers in a way which is dynamic and more efficient than manual/physical methods. Keeping in mind the kind of service providers in context this sort of application will be more efficient and beneficial for them. If you have a look at the customers on the other hand, this application will actually be quintessential for most people stuck in certain situations. Not many other applications target this topic in a general perspective but go for a more specific approach; hence this application may come in handy at more times than many applications combined. Once this project is finished it would be a lively place attracting many customers due to the amount of variety present inside it along with the optimal information.

1.4 Scope

This project helps user to better understanding of binary tree by visualization all concepts will be understood. In computing, binary trees are mainly used for **searching and sorting** as they provide a means to store data hierarchically. Some common operations that can be conducted on binary trees include insertion, deletion, and traversal.

2. INTERFACE REQUIREMENTS:

2.1 Hardware Requirements:

- **Client side:**
  - RAM: 512 GB
  - Hard Disk: 10GB
  - Processor: 1.0 GHz
- **Server Side:**
  - RAM: 1 GB
  - Hard Disk: 20 GB
  - Processor: 2.0 GHz

2.2 Software Requirements:

- **Client Side:**
  - Web Browser: Google Chrome or any other compatible browser
  - Operating System: Windows or any equivalent OS
- **Server side:**
  - Web Server: APACHE
  - Web Browser: Google Chrome or any other compatible browser

2.3 WORKING PRINCIPLE OF PROJECT

STEP 1: Designing:

Using CSS, we develop buttons in order to provide a user-friendly interface. We will create the following buttons:

Clear: This is the first button that the user will see when he visits the website. It will clear all the inserted data.

Quick fill: This button will generate random numbers and inserted accordingly.

Fill: It also generates random numbers and them in specific order slowly so that users are able to see the flow of data.

Add: This button inserts the data in the tree and maintains order.

Search: This button made for searching for any node.

STEP 2: LOGIC

When the user visits the website, he/she can look for the various services that they require specifically on the home page itself. If the user clicks on the add button then the user has to provide the number of insertion he/she wants to perform then the user has to provide the data, by clicking on full or quick fill user can do this task very easily because these buttons generate numbers and are inserted automatically.

STEP 3: Searching the element in the tree

The user can search the element after performing insertion operation, when the user clicks on insertion then one pop up window comes and user has to provide data to search.

STEP 2: Selecting from the categories list

This will allow the user to see all the services available by selecting a particular category present at the home page.

All the services that are offered by the website are listed at the home page.

The user can just go and click on a particular category.

Then the server will check the database for the available options.

If the database contains the particular information, then the webpage will show the various service providers that are available at that instant.

3. FRONT END

Microsoft Visual Basic is the fastest and easiest way to create applications for Microsoft Windows. Whether you are an experienced professional or brand new to windows programming, Visual Basic provides you with a complete set of tools to simplify rapid application development.

The "Visual" part refers to the method used to create the graphical user interface (GUI). Rather than writing numerous lines of code to describe the appearance and location of interface elements, you simply add prebuilt objects into place on screen. If you've ever used a drawing program such as Paint, you already have most of the skills necessary to create an effective user interface.

FOR FRONT END WE USE FOLLOWINGS:

- HTML
- CSS
- JAVASCRIPT

3.1 HTML

The Hypertext Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser.

HTML is the language for describing the structure of Web pages. HTML gives authors the means to: Publish online documents with headings, text, tables, lists, photos, etc. Retrieve online information via hypertext links, at the click of a button.

The History of HTML:

HTML was first created by Tim Berners-Lee, Robert Cailliau, and others starting in **1989**. It stands for Hyper Text Markup Language.

Hypertext means that the document contains **links that allow the reader to jump to other places** in the document or to another document altogether. The latest version is known as HTML5.

A **Markup Language** is a way that computers speak to each other to control how text is processed and presented. To do this HTML uses two things: tags and **attributes**.

What are Tags and Attributes?

Tags and attributes are the basis of HTML.

They work together but perform different functions – it is worth investing 2 minutes in **differentiating the two.**

What Are HTML Tags?

Tags are used to **mark up the start of an HTML element** and they are usually enclosed in angle brackets. An example of a tag is: `<h1>`.

Most tags must be opened `<h1>` and closed `</h1>` in order to function.

What are HTML Attributes?

Attributes contain **additional pieces of information**. Attributes take the form of an opening tag and additional info is **placed inside.**

Editors: **Sublime Text 3** t is free and also offers cross-platform support for Windows, Mac, and Linux users.

3.2 Cascading Style Sheets (CSS)

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, Math ML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is among the core languages of the open web and is standardized across Web browsers according to W3C specifications. Previously, the development of various parts of CSS specification was done synchronously, which allowed the versioning of the latest recommendations. You might have heard about CSS1, CSS2.1, or even CSS3. There will never be a CSS3 or a CSS4; rather, everything is now CSS without a version number.

After CSS 2.1, the scope of the specification increased significantly and the progress on different CSS modules started to differ so much, that it became more effective to develop and release recommendations separately per module. Instead of versioning the CSS specification, W3C now periodically takes a snapshot of the latest stable state of the CSS specification and individual modules progress. CSS modules now have version numbers, or levels, such as CSS Color Module Level 5.

CSS first steps

CSS (Cascading Style Sheets) is used to style and layout web pages — for example, to alter the font, color, size, and spacing of your content, split it into multiple columns, or adds animations and other decorative features. This module provides a gentle beginning to your path towards CSS mastery with the basics of how it works, what the syntax looks like, and how you can start using it to add styling to HTML.

CSS building blocks

This module carries on where CSS first steps left off — now you've gained familiarity with the language and its syntax, and got some basic experience with using it, it's time to dive a bit deeper. This module looks at the cascade and inheritance, all the selector types we have available, units, sizing, styling backgrounds and borders, debugging, and lots more.

The aim here is to provide you with a toolkit for writing competent CSS and help you understand all the essential theory, before moving on to more specific disciplines like text styling and CSS layout.

CSS styling text

With the basics of the CSS language covered, the next CSS topic for you to concentrate on is styling text — one of the most common things you'll do with CSS. Here we look at text styling fundamentals, including setting font, boldness, italics, line and letter spacing, drop shadows, and other text features. We round off the module by looking at applying custom fonts to your page, and styling lists and links

CSS layout

At this point we've already looked at CSS fundamentals, how to style text, and how to style and manipulate the boxes that your content sits inside. Now it's time to look at how to place your boxes in the right place in relation to the viewport, and to each other. We have covered the necessary prerequisites so we can now dive deep into CSS layout, looking at different display settings, modern layout tools like flexbox, CSS grid, and positioning, and some of the legacy techniques you might still want to know about.

Tools for CSS development

We can use the W3C CSS Validation Service to check if your CSS is valid. This is an invaluable debugging tool.
Firefox Developer Tools lets you view and edit a page's live CSS via the Inspector and Style Editor Tools.
The Web Developer extension for Firefox lets you track and edit live CSS on watched sites.

3.3 JAVASCRIPT

JAVASCRIPT: JS is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on users' devices.

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMA Script standard.[10] It has dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM)

Why is it called JavaScript?

When JavaScript was created, it initially had another name: “LiveScript”. But Java was very popular

at that time, so it was decided that positioning a new language as a “younger brother” of Java would help.
But as it evolved, JavaScript became a fully independent language with its own specification called ECMAScript, and now it has no relation to Java at all.

JavaScript first steps

Answers some fundamental questions such as "what is JavaScript?", "what does it look like?", and "what can it do?", along with discussing key JavaScript features such as variables, strings, numbers, and arrays.

JavaScript building blocks

Continues our coverage of JavaScript's key fundamental features, turning our attention to commonly-encountered types of code blocks such as conditional statements, loops, functions, and events.

Introducing JavaScript objects

The object-oriented nature of JavaScript is important to understand if you want to go further with your knowledge of the language and write more efficient code, therefore we've provided this module to help you.

Asynchronous JavaScript

Discusses asynchronous JavaScript, why it is important, and how it can be used to effectively handle potential blocking operations such as fetching resources from a server.

Client-side web APIs

Explores what APIs are, and how to use some of the most common APIs you'll come across often in your development work.

What can in-browser JavaScript do?

Modern JavaScript is a “safe” programming language. It does not provide low-level access to memory or the CPU, because it was initially created for browsers which do not require it.
JavaScript’s capabilities greatly depend on the environment it’s running in. For instance, Node.js supports functions that allow JavaScript to read/write arbitrary files, perform network requests, etc.
In-browser JavaScript can do everything related to webpage manipulation, interaction with the user, and the webserver.

For instance, in-browser JavaScript is able to:

- React to user actions, run on mouse clicks, pointer movements, key presses.
- Send requests over the network to remote servers, download and upload files (so-called AJAX and COMET technologies).
- Get and set cookies, ask questions to the visitor, show messages.
- Remember the data on the client-side ("local storage").

What makes JavaScript unique?

There are at least *three* great things about JavaScript:

1. Full integration with HTML/CSS.
2. Simple things are done simply.
3. Supported by all major browsers and enabled by default.

- ✓ JavaScript was initially created as a browser-only language, but it is now used in many other environments as well.
- ✓ Today, JavaScript has a unique position as the most widely-adopted browser language, fully integrated with HTML/CSS.
- ✓ There are many languages that get "transpiled" to JavaScript and provide certain features. It is recommended to take a look at them, at least briefly, after mastering JavaScript.

4. BACK END

In web development, a back-end developer is responsible for building and maintaining the back-end of a website or web application. The back-end consists of all typical components a typical user does not interact with — including databases, servers, application logic, and APIs. Back-end developers create this infrastructure and work to ensure these components function properly.

4.1 SERVER END:

We assume that the server is well set up and running Apache. What does Apache do? In the simplest terms, it gets a URL from the Internet, turns it into a filename, and sends the file (or its output if it is a program)[9] back down the Internet. That's all it does, and that's all this book is about!
Two main cases arise:

UNIX

The UNIX server has a standalone Apache that listens to one or more ports (port 80 by default) on one or more IP addresses mapped onto the interfaces of its machine. In this mode (known as *standalone mode*), Apache actually runs several copies of itself to handle multiple connections simultaneously.

WIN32

On Windows, there is a single process with multiple threads. Each thread services a single connection. This currently limits Apache 1.3 to 64 simultaneous connections, because there's a system limit of 64 objects for which you can wait at once. This is something of a disadvantage because a busy site can have several hundred simultaneous connections. It has been improved in Apache 2.0. The default maximum is now 1920 — but even that can be extended at compile time.

Server Functions

1. Serving Client Computer Requests
As explained earlier, the server will serve all requests from clients for processing. Be it data requests or applications to be run by the client.
To support these functions, servers usually use a fast and secure operating system. So, clients can work more effectively and of course safely.

2. Storing Data or Information
Another function of the server is as a place to store data sent from the client. The stored data can be in the form of complex types of documents and information.
To be able to accommodate a lot of data, the server must have a large capacity. So, clients can store and access data together with other clients.

3. Provide a Database to Run
The server also has the function of providing a database for data storage and processing. Usually, large companies take advantage of this function to implement big data.
Later, all data stored in the database can be processed and accessed by users. With this ser-

vice, many companies can develop their business products.

4. Manage Data or File Transfer Traffic

The server will manage the communication and transfer of information to the client. Can you imagine how busy the server is when many clients will make requests, right? For this reason, server devices usually have high capacities such as hard drives and RAM.

5. Safeguard from Evil Attack

Finally, the server can also serve to protect a computer or website from hacker attacks.
Every time there is a request for data from the client, the server will check the IP address and other information. If there is anything suspicious, such as a malware threat, the server can prevent access to that IP address. That way, the data stored on the computer or website can be kept safe.

5. SOFTWARE

Programming language

Programming Language for Client Tier we use HTML and JS to design the client tier. Because HTML and JS are easy to use and it will be helpful for us to design the client tier using HTML and JS.

5.1 LIBRARIES AND TOOLS

- p5.js - A library for creating visualizations using the canvas

ABOUT THE LIBRARY

p5.js is a JavaScript library for creative coding, with a focus on making coding accessible and inclusive for artists, designers, educators, beginners, and anyone else! p5.js is free and open-source because we believe software, and the tools to learn it, should be accessible to everyone.

Using the metaphor of a sketch, p5.js has a full set of drawing functionality. However, you're not limited to your drawing canvas. You can think of your whole browser page as your sketch,
including HTML5 objects for text, input, video, webcam, and sound.

6. CASETOOL

6.1 INTRODUCTION TO RATIONAL ROSE:

This is a brief overview to get students started in using Rational Rose to quickly create object-oriented models and diagrams. It is not by any means a complete introduction to Rational Rose, but it should get you started.

A. GETTING STARTED:

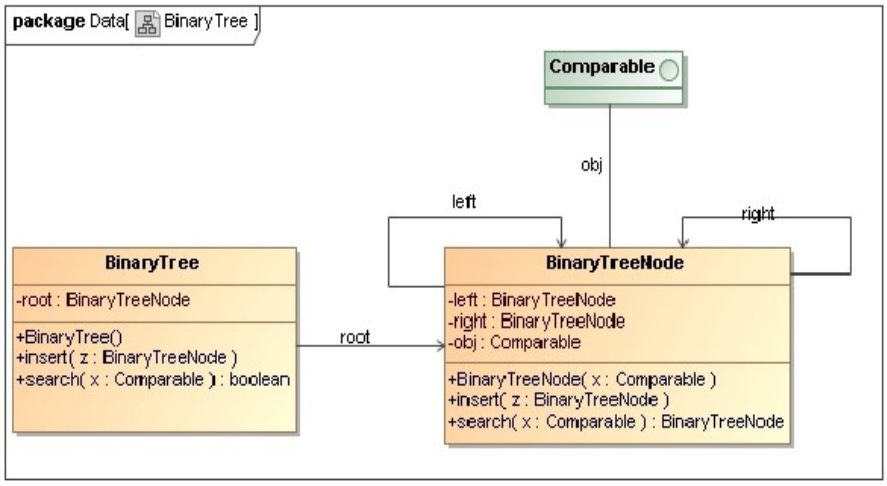
Click on Start, Programs, Rational Suite Development Studio 1.5, and then on Rational Rose 2000 Enterprise Edition (not the one that has the arrow next to it). You will now see a window appearing that says "Create New Model". This is for working with Java, Oracle, and VB. Just press "Cancel". Next, you see the main window called "Rational Rose- [untitled]" and a smaller window within it called "Class Diagram: Logical View / Main". There is also a browser on the left side of the page that has the diagrams that you are working on in the project. If you ever want to see the specifics on any diagram that you are working on, go to the browser, click on the "+ ", and you'll be able to see what is in that view. To close the view, click on the "– ".

6.2 CLASS DIAGRAM:

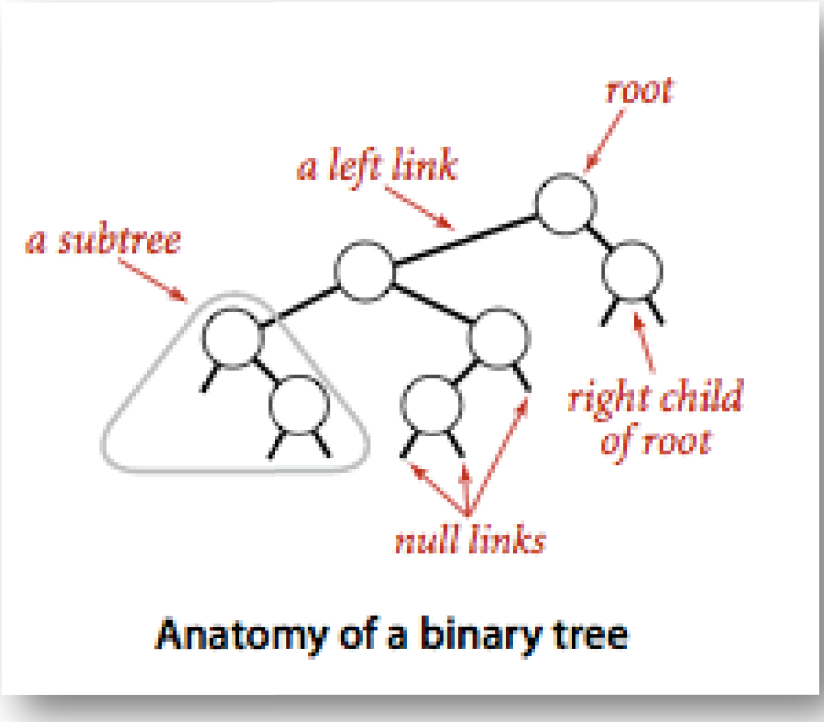
Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints.It is also known as a structural diagram.



6.3 Anatomy of a binary tree



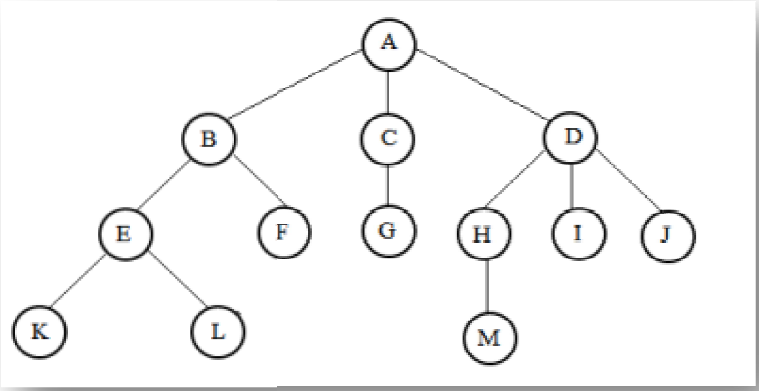
Basic implementation.

Program BST.java implements the ordered symbol-table API using a binary search tree. We define a inner private class to define nodes in BST. Each node contains a key, a value, a left link, a right link, and a node count. The left link points to a BST for items with smaller keys, and the right link points to a BST for items with larger keys. The instance variable `N` gives the node count in the subtree rooted at the node. This field facilitates the implementation of various ordered symbol-table operations, as you will see.

7. MODULE DESCRIPTION

DEFINITION A tree is a finite set of one or more nodes such that

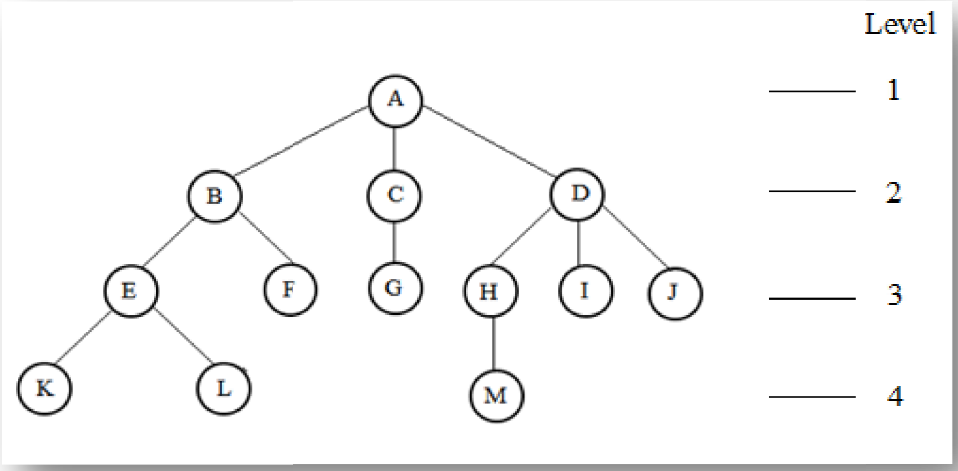
- There is a specially designated node called root.
- The remaining nodes are partitioned into n>= 0 disjoint set T1,…,Tn, where each of these sets is a tree. T1,…,Tn are called the subtrees of the root.



7.1 TERMINOLOGY

- **Node:** The item of information plus the branches to other nodes
- Degree: The number of sub trees of a node
- Degree of a tree: The maximum of the degree of the nodes in the tree.
- Terminal nodes (or leaf): nodes that have degree zero or node with no successor
- Nonterminal nodes: nodes that don't belong to terminal nodes.
- Parent and Children: Suppose N is a node in T with left successor S1 and right
- Successor S2, then N is called the Parent (or father) of S1 and S2. Here, S1 is called left child (or Son) and S2 is called right child (or Son) of N. Siblings: Children of the same parent are said to be siblings.
- Edge: A line drawn from node N of a T to a successor is called an edge
- Path: A sequence of consecutive edges from node N to a node M is called a path.
- Ancestors of a node: All the nodes along the path from the root to that node.
- The level of a node: defined by letting the root be at level zero. If a node is at level l, then it children are at level l+1.
- Height (or depth): The maximum level of any node in the tree

7.2 Example



- A is the root node
- B is the parent of E and F
- C and D are the sibling of B
- E and F are the children of B
- K, L, F, G, M, I, J are external nodes, or leaves
- A, B, C, D, E, H are internal nodes
- The level of E is 3
- The height (depth) of the tree is 4
- The degree of node B is 2
- The degree of the tree is 3
- The ancestors of node M is A, D, H
- The descendants of node D is H, I, J, M

8. IMPLEMENTATION

8.1 **VISUALIZATION:**

1. When the visualization is first opened, the screen will be empty

Fig: 1.1-Visualization

2. To add a node to the tree, click the Add button, and enter an integer for the new node to hold. Watch as the animation adds the node to the tree. In this case, green represents the node that was just added. To pan on the visualization, click and drag. To zoom in and out on the visualization, scroll up or down.

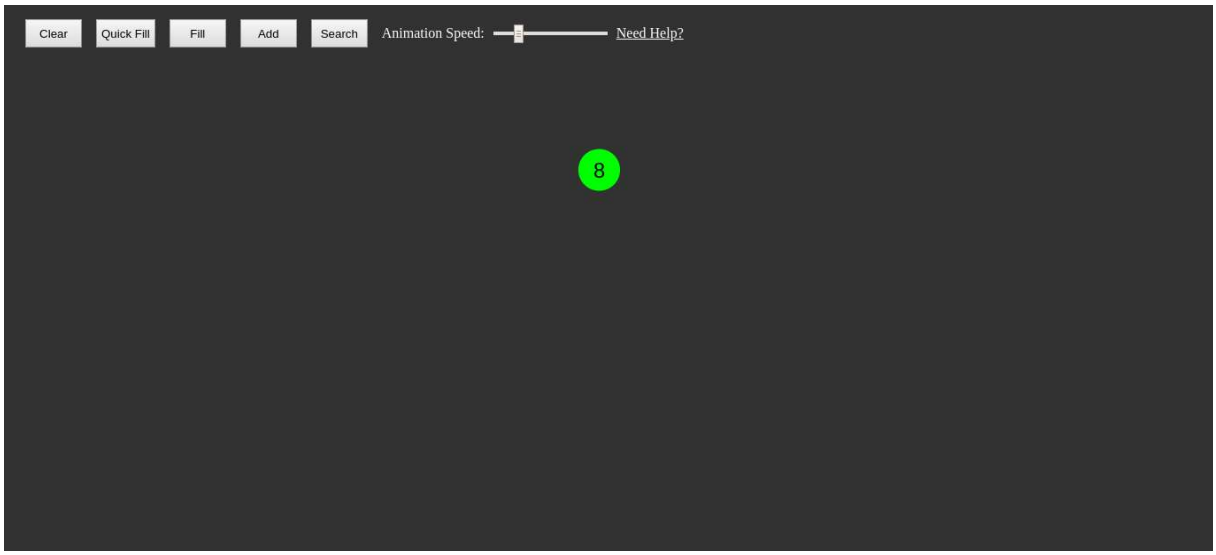


Fig: 1.2-Visualization add

3. Click the Add button a few more times to add more nodes to the tree, and watch as their insertion is animated. To make the animation progress faster, move the Animation Speed slider farther to the right. Blue represents nodes that must be visited to insert the node into the correct location

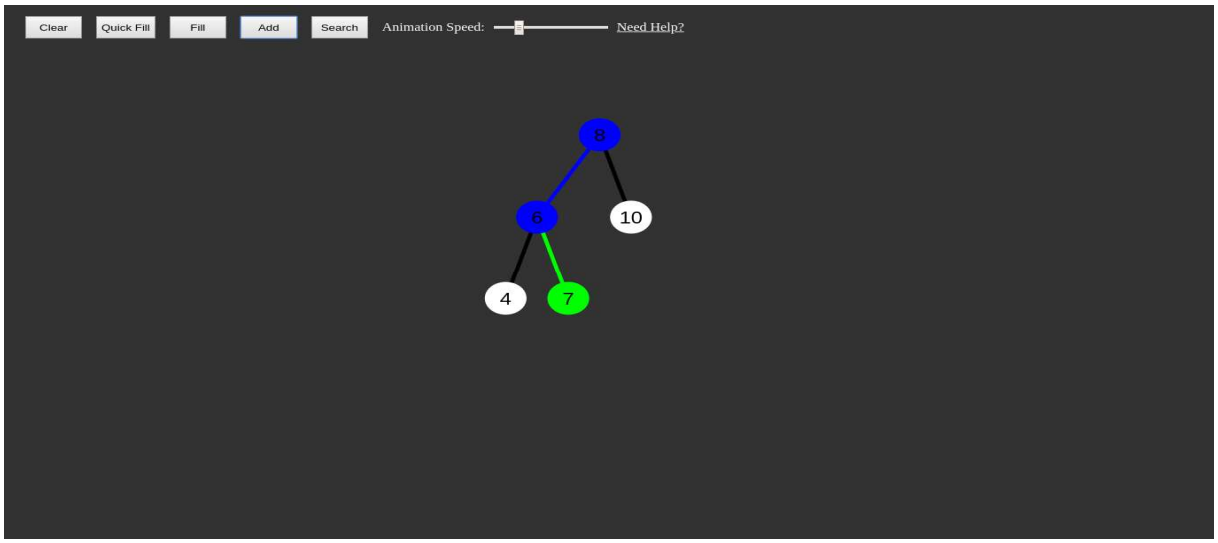


Fig: 1.3-Working of add

4. To watch the binary tree search for a value in itself, click the Search button, and enter an integer that you previously inserted into the tree. Blue represents nodes that must be visited to find the value, red represents sections of the tree where the value will not be found, and green represents the node that was being searched for. Note that if no node is green, the value was not found.

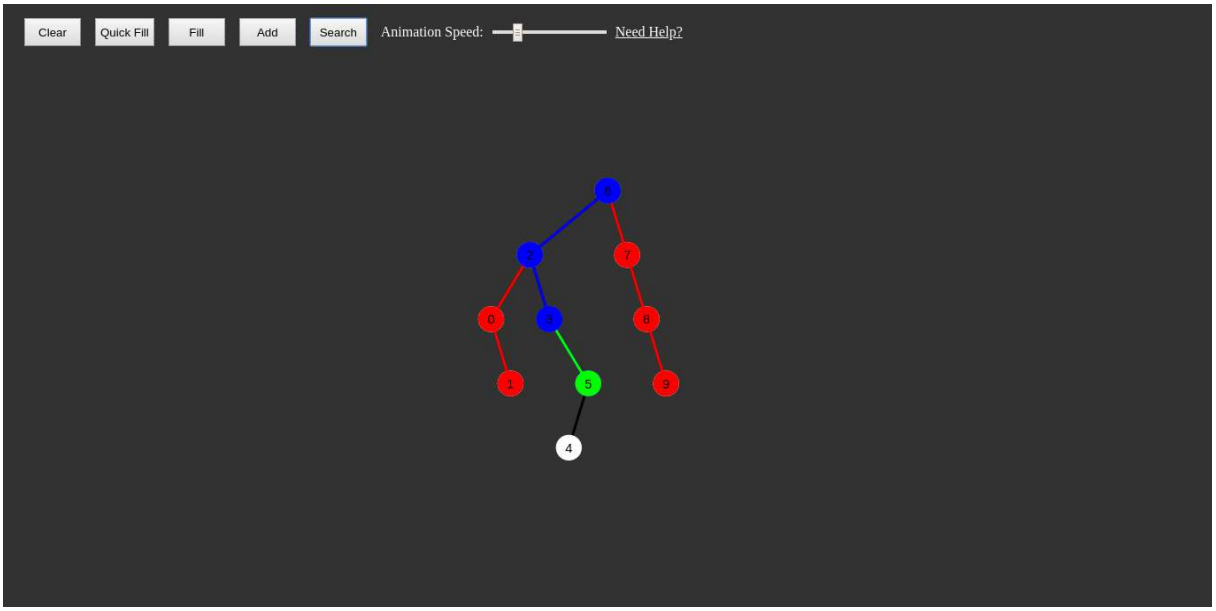


Fig: 1.4-Working of search

5. To clear the entire tree, click the Clear button. This will make your screen appear as it did when the visualization was first opened. Rather than manually adding nodes, you can fill the entire tree with a certain number of nodes. To do this, click the Fill button, enter the desired number of nodes in the prompt, and watch as nodes are added one at a time. Note: this clears the current tree If you do not want to wait for the insertion animation, click the Quick Fill button, which functions the same as the Fill button, but does not animate the insertion process. Note: this also clears the current tree.

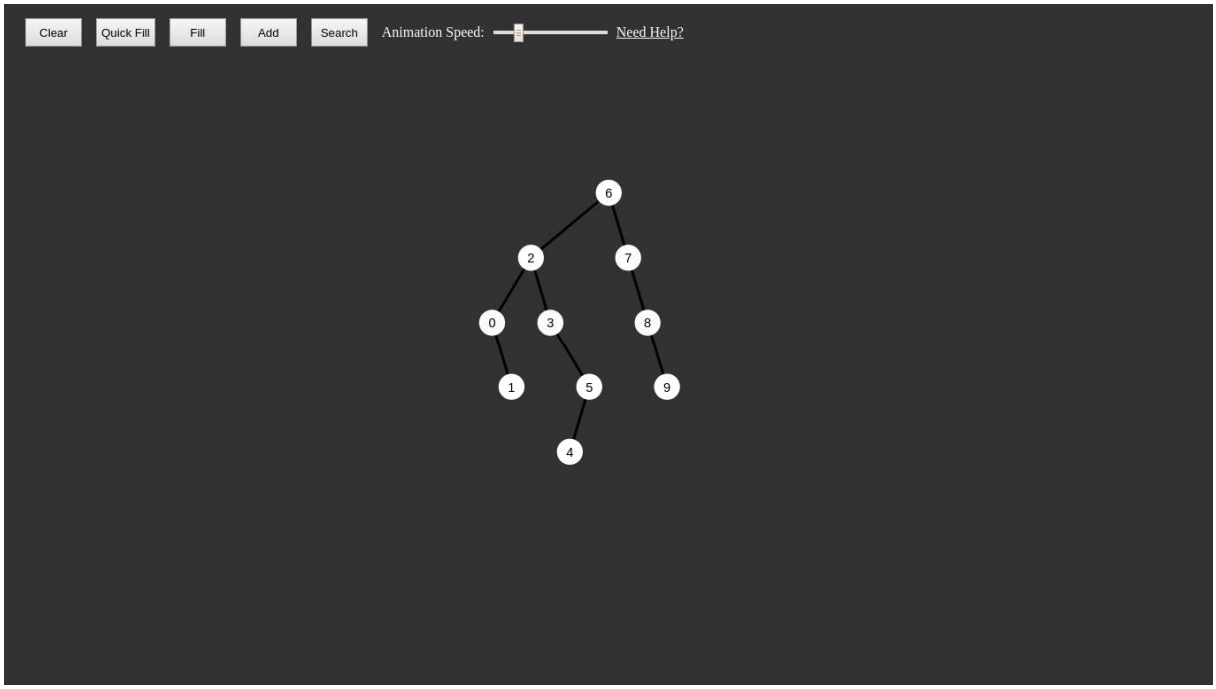


Fig: 1.5 -Working of clear

6. Add and search for nodes in a binary tree with an easy-to-use, web-based visualization

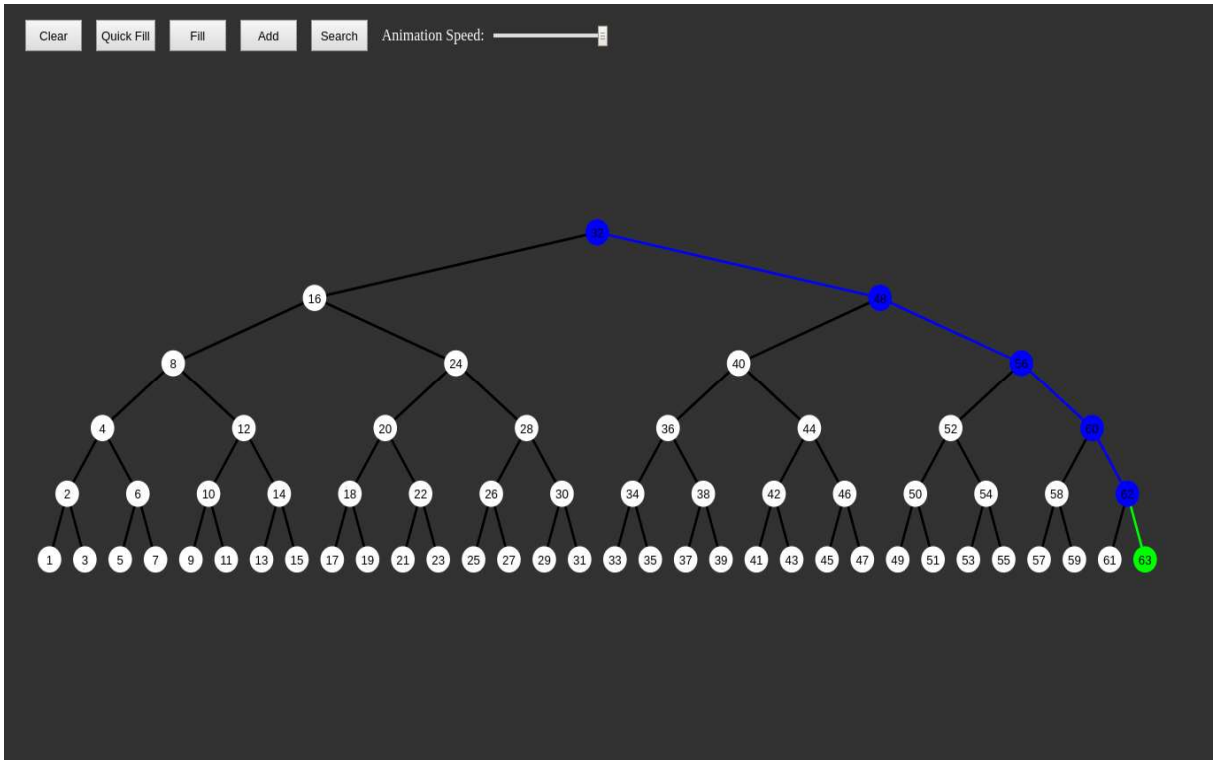


Fig: 1.6 – Searching visualize

8.4 Code Structure

Node.js - Defines the Node class, the building block of the binary tree. Nodes store their value, references to both of their child nodes, and information to draw them on-screen (e.g. x and y-coordinates, colors, etc.).

Tree.js - Defines the Tree class, which serves as both a wrapper for the root node of the binary tree (e.g. providing functions for searching the tree for values), and the primary class responsible for animating the binary tree.

Controls.js - Defines the Controls class, which connects the buttons (e.g. Clear, Quick Fill, etc.) to the animation functions of the Tree class.

Explorer.js - Defines the Explorer class, which adjusts the size and position of the tree to allow for panning and zooming over the entire tree.

sketch.js - Instantiates all the necessary objects to run the visualization.

9. CODING

Index.html

```
<!DOCTYPE html>
<html>
  <head>
    <script src="libraries/p5.min.js"></script>
    <script type="text/javascript">
      new p5(); // On-demand global mode to use color() outside of setup
    </script>
    <link rel="stylesheet" type="text/css" href="src/style.css">
    <meta charset="utf-8">

    <title>Binary Tree Visualization</title>
  </head>
  <body>
    <script src="src/Node.js"></script>
    <script src="src/Tree.js"></script>
    <script src="src/Controls.js"></script>
    <script src="src/Explorer.js"></script>
    <script src="src/sketch.js"></script>

    <div id = "canvas-placeholder"></div>
    <div id = "controls">
      <button id = "clear-btn">Clear</button>
      <button id = "quick-fill-btn">Quick Fill</button>
      <button id = "slow-fill-btn">Fill</button>
      <button id = "add-btn">Add</button>
      <button id = "search-btn">Search</button>
      <span id = "speed-label">Animation Speed:</span>
      <input id = "speed-slider" type="range" min = "0" max = "1.5" step = "0.1" value = "0.3"></input>
    </div>

  </body>
</html>
```

Controls.js

```
class Controls {
  static CLEARID = "clear-btn";
  static QUICKFILLID = "quick-fill-btn";
  static SLOWFILLID = "slow-fill-btn";
  static ADDID = "add-btn";
  static SEARCHID = "search-btn";
  static SLIDERID = "speed-slider";
  static NODELIMIT = 500;

  constructor(tree) {
    this.tree = tree;
    this.tree.bindControls(this);
```

```
    this.animationInterval = null;
    this.clearBtn = document.getElementById(Controls.CLEARID);
    this.quickFillBtn = document.getElementById(Controls.QUICKFILLID);
    this.slowFillBtn = document.getElementById(Controls.SLOWFILLID);
    this.addBtn = document.getElementById(Controls.ADDID);
    this.searchBtn = document.getElementById(Controls.SEARCHID);
    this.speedSlider = document.getElementById(Controls.SLIDERID);
    this.setAnimationSpeed();
    this.clearBtn.addEventListener('click',
        () => this.triggerAnimation(this.clear));
    this.quickFillBtn.addEventListener('click',
        () => this.triggerAnimation(this.quickFill));
    this.slowFillBtn.addEventListener('click',
        () => this.triggerAnimation(this.slowFill));
    this.addBtn.addEventListener('click',
        () => this.triggerAnimation(this.add));
    this.searchBtn.addEventListener('click',
        () => this.triggerAnimation(this.search));
    this.speedSlider.addEventListener('input', this.setAnimationSpeed.bind(this));
}
clear() {
    this.tree.clear();
    this.tree.stopAnimation(() => {})
    this.tree.draw();
}

triggerAnimation(animation) {
    if(this.tree.running) {
        alert('Please wait for the current animation to finish');
    } else {
        animation.bind(this)();
    }
}
getNumber(text) {
    var value = prompt(text);

    if(value === null) {
        return null;
    } else if(isNaN(parseInt(value)) || value === "" || parseInt(value) < 0) {
        alert('Please enter a positive integer');
        return null;
    } else {
        return parseInt(value);
    }
}
quickFill() {
    var count = this.getNumber("Number of nodes: ");

    if(count !== null && (count < Controls.NODELIMIT ||
            confirm(count + ' nodes may reduce performance. Continue anyways?'))) {
        this.tree.fill(count);
```

```
        }
    }
    slowFill() {
        var count = this.getNumber("Number of nodes: ");

        if(count !== null && (count < Controls.NODELIMIT ||
              confirm(count + ' nodes may reduce performance. Continue anyways?'))) {
            this.tree.fillVisual(count);
        }
    }
    add() {
        var value = this.getNumber("Value to add: ");

        if(value !== null && this.tree.search(value)) {
            alert(value + ' is already in the tree');
        } else if(value !== null){
            this.tree.addValueVisual(value);
        }
    }
    search() {
        var value = this.getNumber("Value to search for: ");

        if(value !== null) {
            this.tree.searchVisual(value)
        }
    }
    setAnimationSpeed() {
        this.animationInterval= 1000/Math.pow(10, this.speedSlider.value);
    }
}
```

Tree.js

```
class Tree {
    constructor(x, y, backgroundColor) {
        this.graphicsBuffer = createGraphics(CANVASWIDTH, CANVASHEIGHT);
        this.root = new Node(this.graphicsBuffer);
        this.controls = null;
        this.x = x;
        this.y = y;
        this.backgroundColor = backgroundColor;
        this.running = false;
        this.timeout = null;
        this.node = null;
        this.draw();
    }
    bindControls(controls) {
        this.controls = controls;
    }
    clear() {
```

```
    this.root = new Node(this.graphicsBuffer);
}
uniqueRandom(max) {
    while(true) {
        var value = Math.floor(random(0, max));

        if(!this.search(value)) {
            return value;
        }
    }
}
fill(count) {
    this.clear();

    for(var i = 0; i < count; i++) {
        var value = this.uniqueRandom(count);

        this.addValue(value);
    }

    this.draw();
}
addValue(value) {
    var shiftedNode = this.root.addValue(value);

    this.setCoordinates(shiftedNode);
}
search(value) {
    return this.root.search(value);
}
setCoordinates(node) {
    if(node === this.root) {
        node.setCoordinates(this.x, this.y);
    } else {
        node.setCoordinates();
    }
}
draw() {
    this.graphicsBuffer.background(this.backgroundColor);

    if(this.root.isFilled()) {
        this.root.draw();
    }

    this.updateDrawing();
}
updateDrawing() {
    image(this.graphicsBuffer, 0, 0);
}
resetVisuals() {
    this.root.resetVisuals();
```

```
    this.draw();
}
startAnimation(frame, ...args) {
    if(this.running) {
        throw Error('Animation is currently running');
    } else {
        this.running = true;
        this.node = this.root;

        this.resetVisuals();
        this.continueAnimation(frame.bind(this), ...args)
    }
}
continueAnimation(frame, ...args) {
    this.timeout = setTimeout(() => frame.bind(this)(...args),
        this.controls.animationInterval);
}
stopAnimation(complete = () => {}, ...callbackArgs) {
    this.running = false;
    this.node = null;

    clearTimeout(this.timeout);

    setTimeout(() => complete(...callbackArgs), this.controls.animationInterval);
}

// Call for starting an addValue animation
// value is the value to add
// complete is a callback called when the animation finishes
addValueVisual(value, complete = () => {}, ...callbackArgs) {
    this.startAnimation(this.addValueFrame, value, complete, ...callbackArgs);
}

// Single frame for the addValue animation
// Arguments match addValueVisual
addValueFrame(value, complete, ...callbackArgs) {
    if(!this.node.isFilled()) {
        this.addValue(value);         // Add the value to the data structure

        this.node.paint(Node.SUCCESS); // Mark this node as inserted

        this.draw();                  // Show the tree with the new value

        this.stopAnimation(complete, ...callbackArgs);

    } else {
        this.node.paint(Node.VISITED); // Mark this node as visited

        this.updateDrawing();         // Display the new color

        // Determine the node for the next frame
        if(value < this.node.value) {
```

```
            this.node = this.node.leftNode;

        } else if(value > this.node.value) {
            this.node = this.node.rightNode;
        }
        this.continueAnimation(this.addValueFrame, value, complete, ...callbackArgs)
    }
}
searchVisual(value, complete = () => {}, ...callbackArgs) {
    this.startAnimation(this.searchFrame, value, complete, ...callbackArgs);
    console.log('searching visually')
}
searchFrame(value, complete, ...callbackArgs) {
    if(this.node.color !== Node.VISITED) {
        // Mark the root node as visited first, then continue the search
        this.root.paint(Node.VISITED);

        this.updateDrawing();

        this.continueAnimation(this.searchFrame, value, complete, ...callbackArgs);

    } else if(!this.node.isFilled()) {
        // The value isn't in the tree, stop the animation

        this.stopAnimation(complete, ...callbackArgs);

    } else if(this.node.value === value) {
        // The value is in this node

        this.node.paint(Node.SUCCESS);  // Mark the node as found

        this.updateDrawing();           // Display the new color

        this.stopAnimation(complete, ...callbackArgs);

    } else {
        // The value may be in another node

        var nextHalf; // The half of the tree being searched next
        var cutHalf;  // The hal of the tree that can be cut from search

        // Set the two variables correctly
        if(value < this.node.value) {
            nextHalf = this.node.leftNode;
            cutHalf = this.node.rightNode;

        } else if(value > this.node.value) {
            nextHalf = this.node.rightNode;
            cutHalf = this.node.leftNode;
        }
        this.node = nextHalf;
        cutHalf.recursivePaint(Node.FAILURE);
```

```
            cutHalf.draw();
            nextHalf.paint(Node.VISITED);
            this.updateDrawing();

            this.continueAnimation(this.searchFrame, value, complete, ...callbackArgs);
        }
    }
    fillVisual(count, complete = () => {}) {
        this.clear();

        this.startAnimation(this.fillFrame, count, 0, complete);
    }
    fillFrame(count, filled, complete) {
        if(filled === count) {
            this.stopAnimation(complete);
        } else {
            this.stopAnimation();

            var value = this.uniqueRandom(count);
            this.startAnimation(this.addValueFrame, value,
                this.fillFrame.bind(this), count, filled + 1, complete);
        }
    }
}
```

10. TESTING

10.1 INTRODUCTION:

Testing is a set of activities that can be planned in advance and conducted systematically. Testing requires that the developer discard preconceived notions of the correctness of the software just developed and overcome a conflict of interest that occurs when errors are encountered. Testing also provides the main objective of our project and understands the risk of implementation. Testing is a process of technical investigation, performed on behalf of stakeholder, that is intended to reveal quantity related information about the product with respect to the context in which it is intended to operate. Testing is the process of executing a program or an application with the intent of finding an error or bugs. Testing can be stated as the process of validating and verifying that a software program/application/product:

10.2 TEST PLAN:

Test plan will describe the scope and activities of our modules in the project. We must plan the test plans at the starting of our project. It will provide a unique identifier for our document. Testing should begin in "small" and proceeds in the "large". Exhaustive testing is not possible; it provides an overview of our test plan. The goal of the system testing process was to determine all faults in our project. The program was subjected to a set of test inputs and many explanations were made and based on these explanations it will be decided whether the program behaves as expected or not. Our Project went through two levels of testing

1. Unit testing

2. Integration testing

10.2.1 UNIT TESTING:

Unit testing is commenced when a unit has been created and effectively reviewed. In order to test a single module we need to provide a complete environment i.e. besides the section we would require

1. The procedures belonging to other units that the unit under test calls
2. Non local data structures that module accesses
3. A procedure to call the functions of the unit under test with appropriate parameters

1. Test for the searching

Testing searching module-This test whether the searching is done in the right way or not like if we give input as any alphabet instead of numbers then its should not crash the program it should have to give any result.

2. Test for the add

It test whether the add button works fine or not like instead of adding element it should not delete any element and while taking user input it should accept only numbers

3. Test for the Quick fill

It tests whether Quick fill works fine or not means if we give input as 10 nodes to be filled then it should generate ten random elements and insert them.

4. Test for the Fill

Test whether Fill button working properly or not it also do similar work as Quick fill but it animate the element while filling so that user have good understanding of binary insertion

5. Test for the clear

Clear is important if the user wants to restart the binary tree so we are checking whether clear is working or not.

10.2.2 INTEGRATION TESTING:

In the Integration testing we test various combinations of the project module by providing the input. The primary objective is to test the module interfaces in order to confirm that no errors are occurring when one module invokes the other module.

10.2.3 VALIDATION TESTING:

While verification is a quality control process, the quality assurance process carried out before the software is ready for release is known as validation testing. The validation testing goal is to validate and be confident about the software product or system, that it fulfills the requirements given by the customer. The acceptance of the software from the end customer is also a part of validation testing. Validation testing answers the question, "Are you building the right software system". Another question, which the entire process of validation testing in software engineering answers is, "Is the deliverable fit for purpose". In other words, does the software system provide the right solution to the problem? Therefore, often the testing activities are introduced early in the software development life cycle. The two major areas, when validation testing should take place are in the early stages of software development and towards the end, when the product is ready for release.In other words, it is acceptance testing which is a part of validation testing.

11. OUTPUT SCREENS

LANDING PAGE

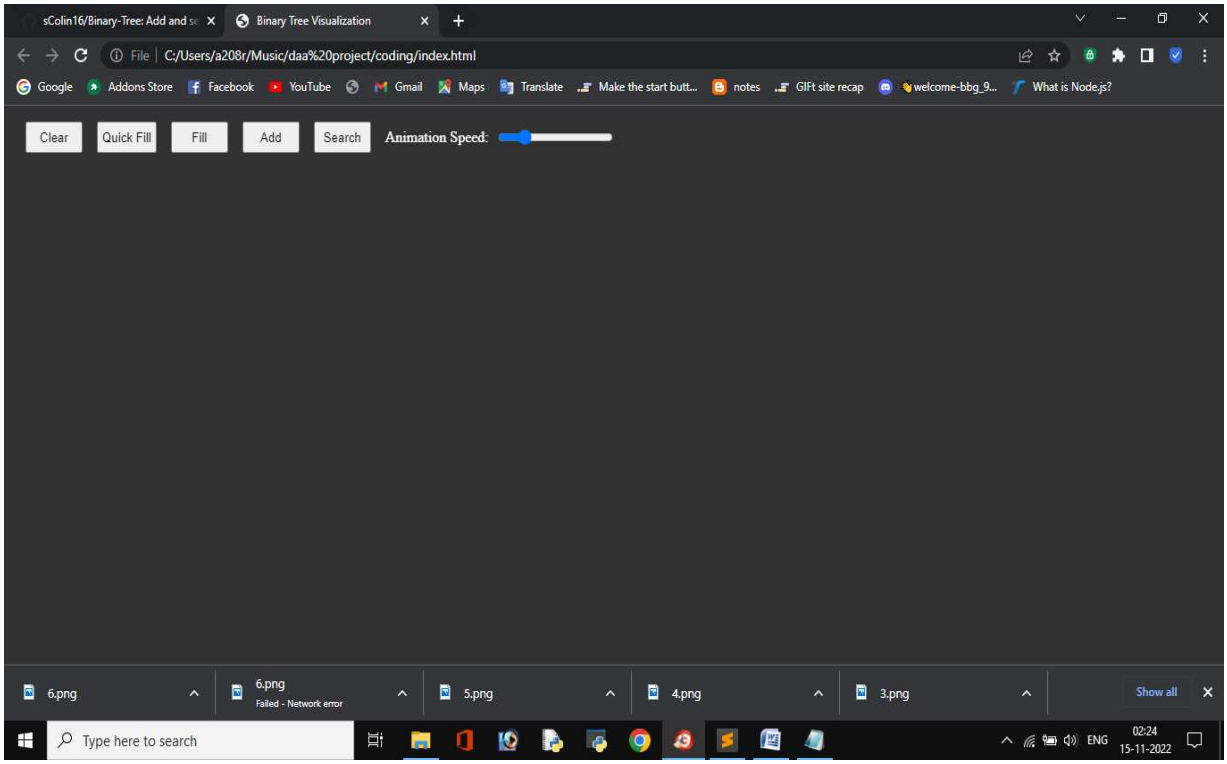


Fig: 2.1–Landing page

This is landing page where user will see the first look of our project

QUICK FILL

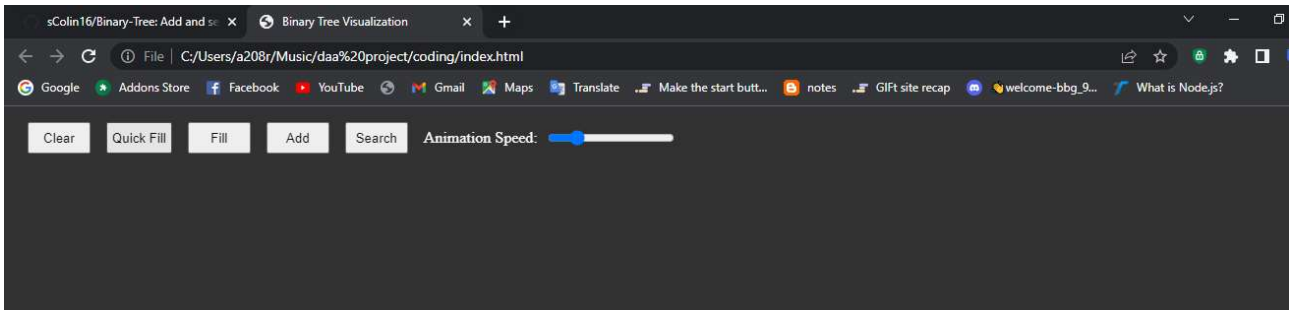


Fig: 2.2-Quick fill

In Quick fill it will ask for number of nodes to generate with binary tree as follows:

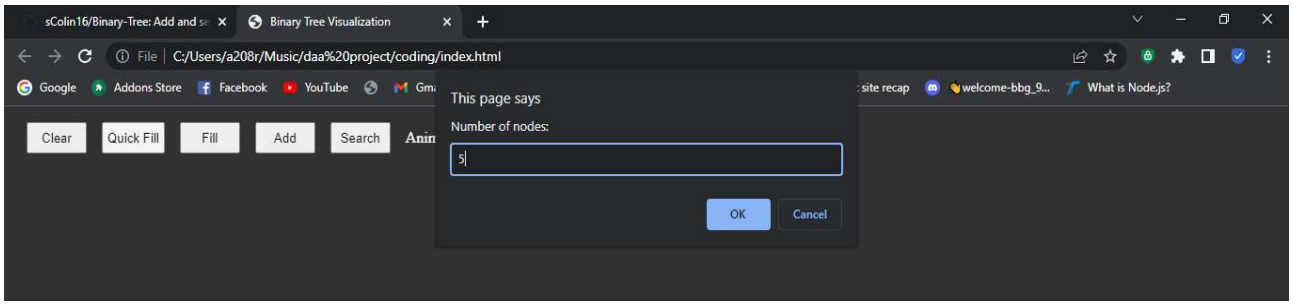


Fig: 2.3-Quick fill input

After taking input from user it generates nodes as follows:

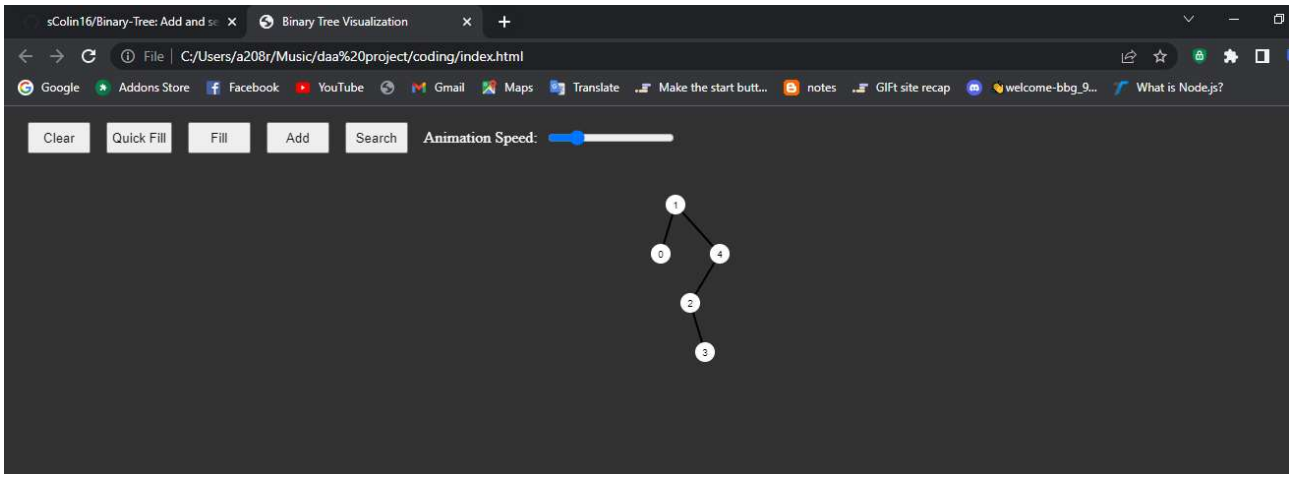


Fig: 2.4-Quick fill output

FILL

Fill also does similar work as Quick fill but it generates random numbers and fill the binary tree slow so that user are able to see how insertion is going on in binary tree.

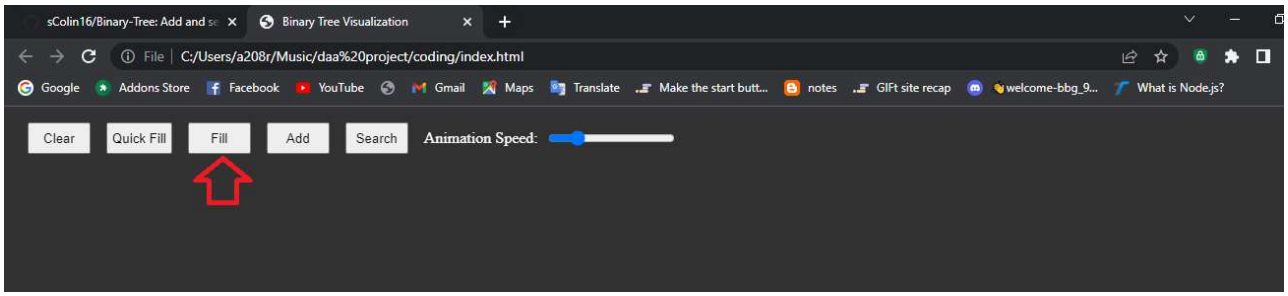


Fig: 2.5- Fill Representation

Fill requires an input it as number of nodes to be placed in binary tree as follows:

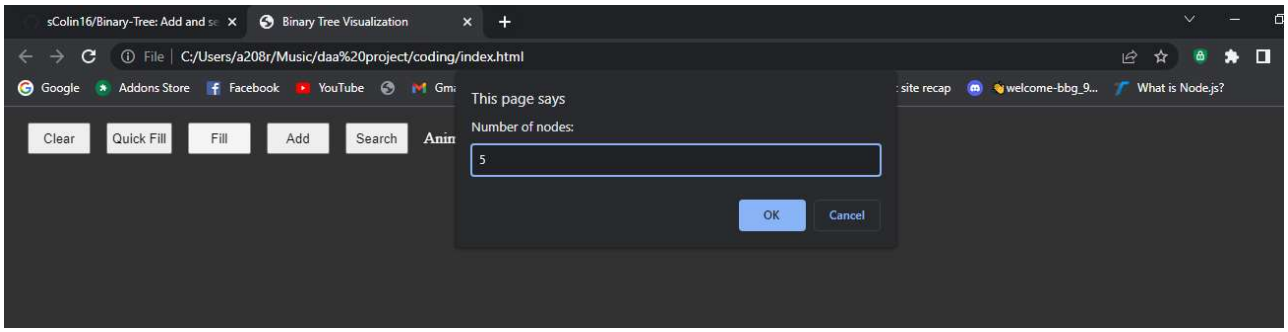


Fig: 2.6- Fill input

Now it insets all the randomly generated nodes as follows:

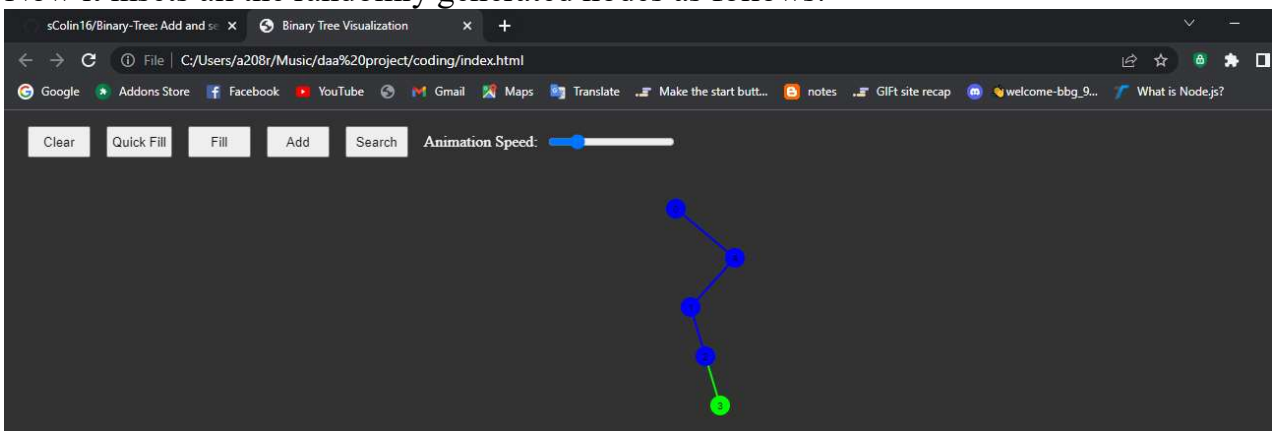


Fig: 2.7- Fill output

ADD

Add button will insert the given node provided by the user:

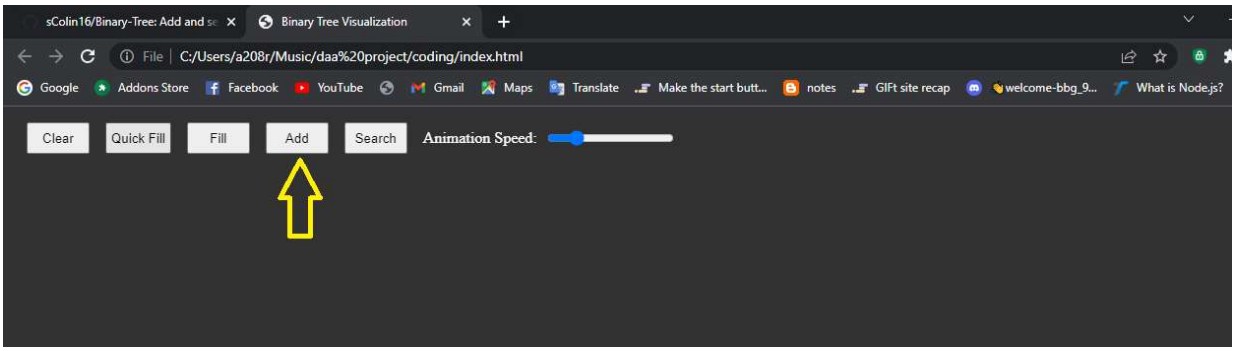


Fig: 2.8 – Add Representation

Add requires an input from user that what value to be inserted in binary tree as:

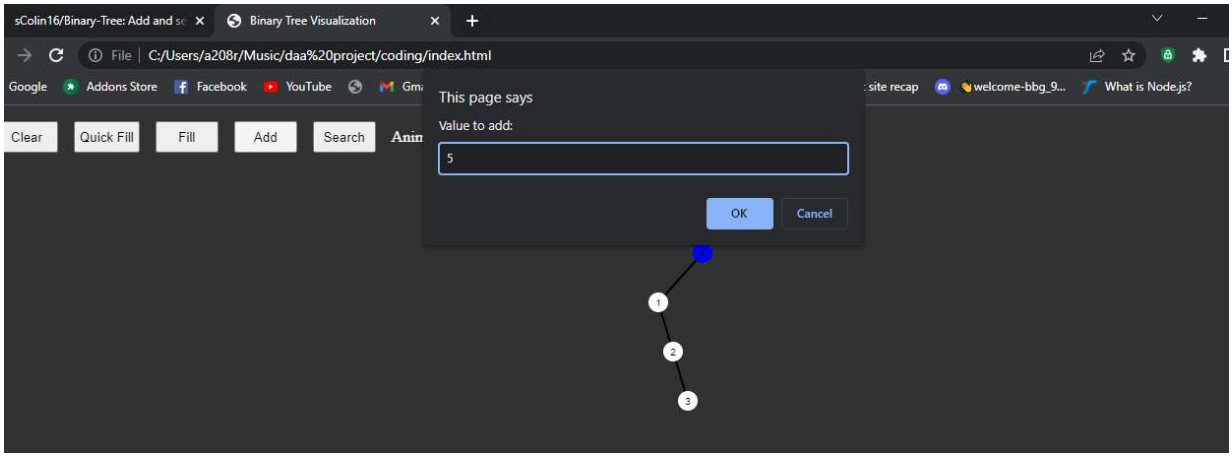


Fig: 2.9 – Add Input

After insertion our nodes inserted successfully

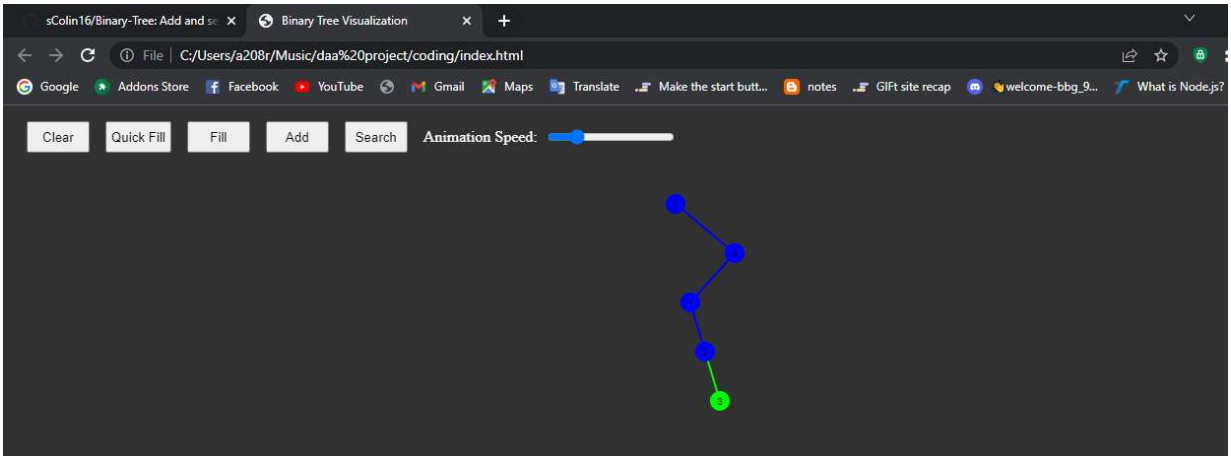


Fig: 3.0 – Add Output

SEARCH

Search will find out whether the specific node is present or not.

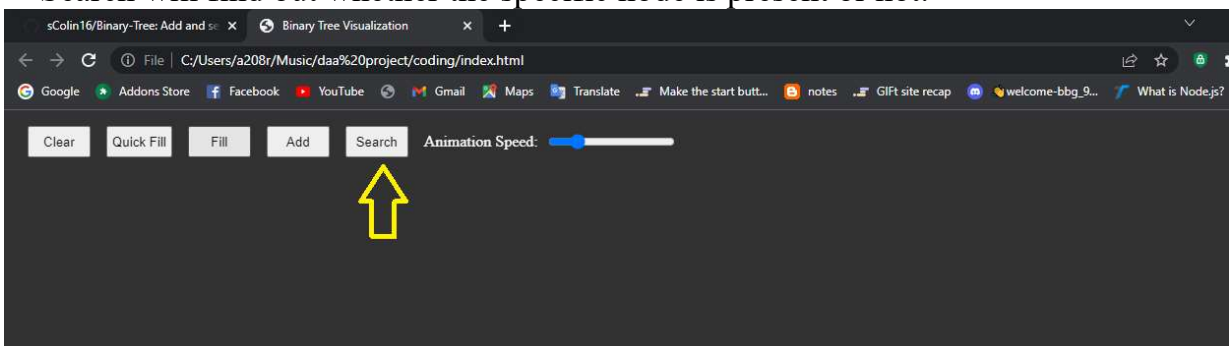


Fig : 3.1- Search Representation

Search requires one input as which element to search as follows:

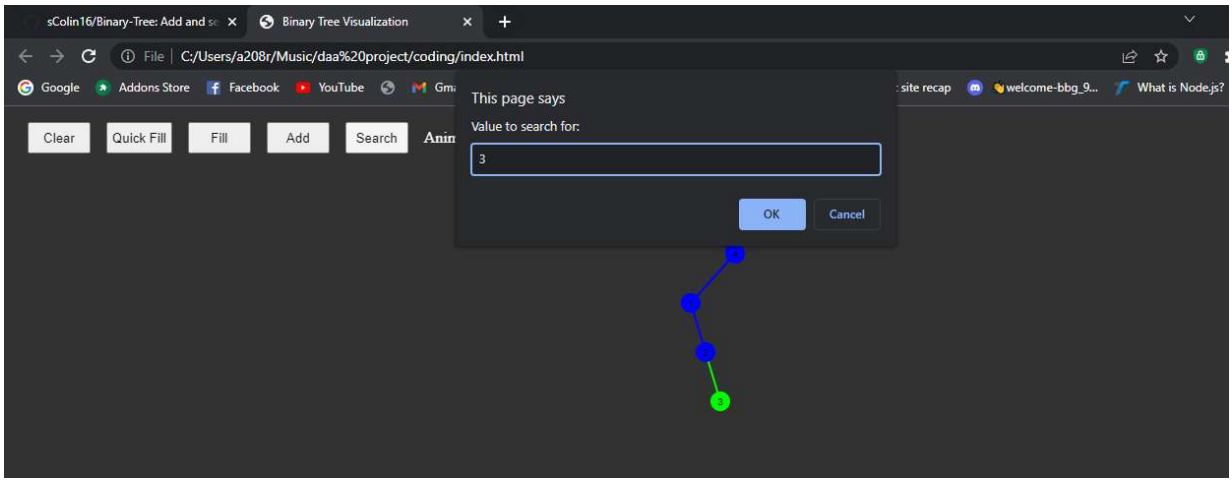


Fig:3.2 – Search Input

Based on user input binary tree search whether given input is present or not:

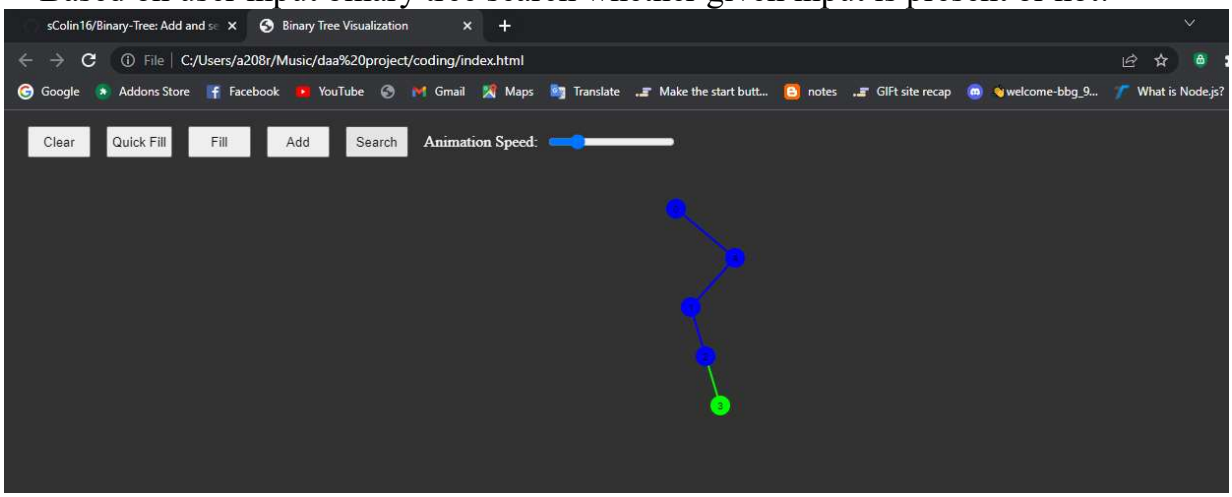


Fig: 3.3 – Search Output

Our element is present as it is represented in green color.

12. FUTURE ENHANCEMENTS

A project is never actually complete as it will always find some room for improvements or have a need to catch up to the latest demand for technology. A few points can be observed to provide effective enhancements to the project. Such as the precision of the search results for searching we can implement many algorithms but here we use binary search tree method based upon user requirement they can change in future.

As in this we are using the p5min library for drawing and canvas purposes. In the future we can use those libraries for various purposes.

Here the searching algorithm which we are implementing can be implemented in websites for searching products in websites and many more.

13. CONCLUSION

The users who will be using this can be beneficial as learning by doing. By this user will have great understanding of Binary tree as it visualizes the insert and search. The manually done work will be drastically avoided making the application more feasible. This Application provides an online in the version of Local Service System which will benefit the users who want to access admissible records of serviceman and also help to search serviceman according to his/her requirement in their own locality. The project was designed in such a way that future changes can be done easily.

14. REFERENCES

- https://www.w3schools.com/html/html5_canvas.asp
- https://www.w3schools.com/js/
- http://www.mysqltutorial.org
- https://www.geeksforgeeks.org/tree-traversals-inorder-preorder-and-postorder/
- https://www.tutorialspoint.com/data_structures_algorithms/tree_traversal.htm
- https://www.cs.usfca.edu/~galles/visualization/BST.html