`

ALGORITHM VISUALIZER

A project submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in
INFORMATION TECHNOLOGY

Submitted by:
Mr. Amey Gupta
Roll No.:12012015

Group No- 109

Supervised by:
Dr. Mukesh Mann
Assistant Professor

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, SONEPAT -131201, HARYANA, INDIA

ACKNOWLEDGEMENTS

I would like to express my special thanks of gratitude to our mentor and teacher **Prof (Dr) Mukesh Mann, Professor - IT,** and IIIT Sonepat for allowing us to work on this project. The successful completion of this task would be impossible without his ceaseless guidance and assistance. I'd also like to thank my team members for helping, assisting, and providing ideas and motivation at every turn.

Amey Gupta

`

SELF DECLARATION

I am Amey Gupta, B.Tech. (IT) student, hereby declare that the project titled **"Algorithm Visualizer",** which I have submitted to the Department of Information Technology, **Indian Institute of Information Technology Sonepat, Haryana, Sonepat,** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Information Technology has not previously formed the basis for the award of any degree or any other diploma.

We hereby declare that the work contained in the project titled "**Algorithm Visualizer**" is original. I have adhered to all the project ethics requirements to the best of my abilities. All sources of information used in the research have been acknowledged.

Name: Amey Gupta
Roll No.: 12012015

Department of Information Technology,
Indian Institute of Information Technology,
Sonepat-131201, Haryana, India.

`

CERTIFICATE

This is to certify that **Mr.Amey Gupta**, has worked on the project entitled "**Algorithm Visualizer**" under my supervision and guidance.

The contents of the project, being submitted to the **Department of Information Technology**, **IIIT, Sonepat,** for the award of the degree of **B. Tech** in **Information Technology** are original and have been carried out by the candidates themselves. This project has not been submitted for the granting of any other degree or diploma at this or any other university, in whole or in part.

Dr. Mukesh Mann
Supervisor

Department of Computer Science and Engineering,
Indian Institute of Information Technology,
Sonepat-131201, Haryana, India

ABSTRACT

Name: **Amey Gupta** Roll No.: **12012015**

Degree for which submitted **B. Tech (IT),**

Department of Information Technology, **IIIT Sonepat**

Project Title: **Algorithm Visualizer**

Name of the project supervisor: **Dr. Mukesh Mann**

Month and year of the report submission: **December 2021**

According to the Times of India, coding is becoming one of the most crucial skills to learn in the 21st century, helps children sharpen their creativity and think outside the box. Algorithms are highly recurrent in computer science and mathematics to solve or perform various computations. An algorithm, in its simplest form, is a process or set of instructions that are followed in problem-solving operations and other calculations, especially by a computer. The project is meant to focus on W/A and S/w related Development purposes rather than focusing on a research point of view. The prime objective behind this topic is that there is a lot of potential in learning programming languages and more specifically in their associated algorithms and data structures. Our website aids students in learning about algorithms for the first time. A visual representation of how an algorithm function ensures students grasp the concept very well. The platforms and programming languages used till now are CSS, HTML, Bootstrap, Js.

The prime Results which wished to achieve are: -

To provide a user-friendly interface and platform for students learning programming and more specifically algorithms. By ensuring ease of access, we hope students can widen their knowledge on how an algorithm operates and that potential doubts can be cleared.

`

LIST OF ABBREVIATIONS

HTML	Hyper Text markup language
CSS	Cascading Style Sheets
BT	Bootstrap
IT	Information Technology
JS	JavaScript
W/A	Web Application
S/W	Software
UI	User Interface
PUW	Project Under Work
NAV	Navigation

`

LIST OF FIGURES

`

Beginning

TABLE OF CONTENTS

`

`

`

CHAPTER-1

1.1 Introduction

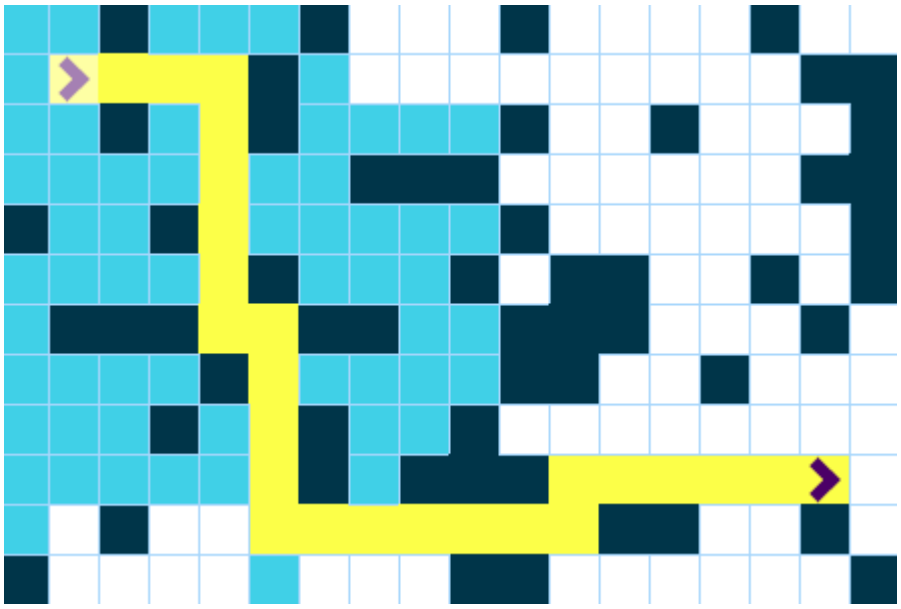


Figure 1.1- *Algorithm visualizer*

The project '**Algorithm Visualizer**' is a website that is free of cost that provides a platform to learn algorithms with help of visual animations and provides you with an innovative and interesting way to learn Coding. It not only provides a visual animation but also codes snippets that help understand the code line by line, thus making your learning easier.

Our website also provides various options like design maze patterns, multiple locations, and various algorithms to make the learning experience much more efficient.

It also provides an opportunity to learn various types of algorithm syntax in a very short time with fully animated explanations, all this on one single platform '**Algorithm Visualizer'**.

1.2 Problem Outline

The number of students pursuing computer science is steadily increasing. There are various online resources that facilitate learning of computer science through a thorough explanation of the material and what it is. One such core component is algorithms. The

`

learning of algorithms is the most essential aspect of computer science and most students find understanding it lethargic and boring. Most resources such as books, lectures, and videos explain what algorithms are, in depth, but not how they work. Thus it is difficult for students to decipher how an algorithm works.

1.3 Project Objective

The main objective of this project is to help you remember and increase your understanding of the path-finding algorithms at zero costs and totally free to every student in the world and improve their approach to real world problems and improve their creativity. The prime objective of the project is to build an easy-to-use or user-friendly interface with some of the most well-known pathfinding algorithms. This will help overcome the limitations of the virtual learning of algorithms for the users.

1.4 Project Methodology

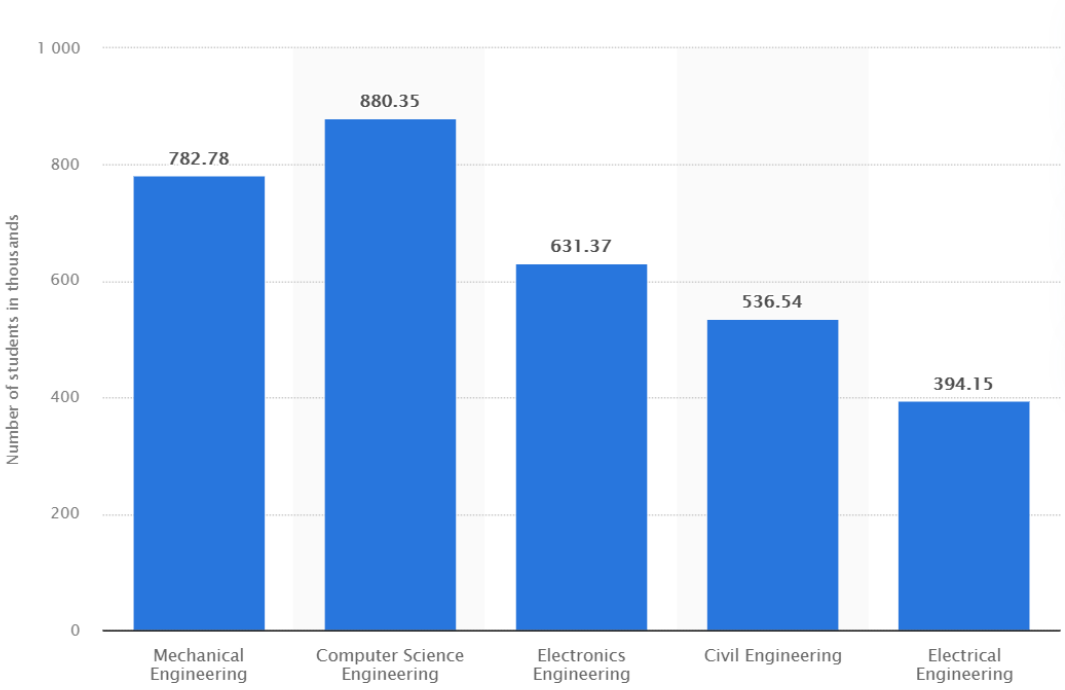
The methodology used to achieve the upcoming objectives are as follows: -

The PUW will have a user-friendly interface and it would be achieved by making the interface a bit modern and different from the existing interfaces and it is achieved by using JS, CSS, HTML, and BT. Predominantly it can have all the popular algorithms and various maze patterns to teach path-finding algorithms in a playful manner.Our platform would be providing every minuscule detail about the algorithm and teaching space complexity, time complexity and Code snippets.

1.5 Scope of Project Work

- This project is helpful for people all over the globe as mostly students are preferring computer science.

- Proper Functioning of the Project requires an active internet connection.



1.6 Organization of project

1.6 Organization of project

The whole project can broadly be categorized into 4 chapters out of which 2 are discussed in the upcoming chapters.

Chapter 1 of this report explains the basic functionality and objectives of the project in brief and the methodologies used to overcome those objectives. Any prerequisite is also discussed under section 1.4 Project Methodologies.

Chapter 2 of this report describes the basic structure and the logic involved in the development of the objectives set under section 1.3 Project Objectives. It includes code snippets and basic building blocks.

`

1.7 Summary

- The project '**Algorithm Visualizer**' is a Web Application.
- Main Aim – To help in understanding pathfinding algorithms with the help of virtual explanations.
- For all people around the globe.
- You can also see space and time complexity with code snippets.
- Development of this entire project is done using-
    1. **HTML and CSS**
    2. **JS and BOOTSTRAP**
- **Scope** of Project
    - For all people around the globe.
    - Because of Free tier usage of GitHub there are no restrictions in accessing the website.
    - Active internet connection

Chapter 2

`

Study and Review of Literature

2.1 INTRODUCTION

An algorithm, in its simplest form, is a process or set of instructions that are followed in problem-solving operations and other calculations, especially by a computer. Algorithms are highly recurrent in computer science and mathematics to solve or perform various computations. Computer-executable instructions in a finite series of well-defined specifications typify algorithms in general. Algorithms are unambiguous in nature so that they can be utilized to process data and automate reasoning.

Algorithms run efficiently when expressed within a finite amount of time and space, and in a defined formal language, leading to calculations of functions. An algorithm initializes from an inceptive input and state(which may even be empty) which eventually produces an output and culminates at the final termination state through a computation that when effectuated, proceeds through a finite number of defined successive states.

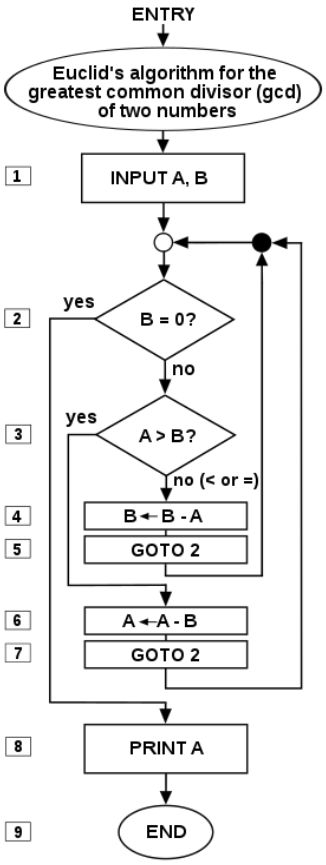


Figure 2.1- *Flowchart of an algorithm*

Figure 2.1 above illustrates an algorithm in action and depicts how the algorithm would function for calculating the greatest common divisor of two numbers.

Algorithm visualization is a method of studying algorithms through the use of images to convey useful information. That information can be a visual illustration of an algorithm's operation, or of its performance on different kinds of inputs, or its execution speed versus that of other algorithms for the same problem. To accomplish this goal, an algorithm visualization uses graphic elements such as points, line segments, two- or three-dimensional bars, etc. to represent an algorithm in operation.

This chapter presents some discussions about the relevant tools and technologies used to develop an algorithm visualizer. Some of the tools and technologies or the pre-requisite which are used in the development of front-end and UI are JavaScript, HTML, and CSS.

2.2 Front-End

Front-end web development, also known as client-side development is the practice of

producing HTML, CSS, and JavaScript for a website or Web Application to allow direct interaction for the user. The front-end is the presentation layer whilst the back-end consists of the data across the software or the physical infrastructure. The front is an abstraction that provides a user-friendly interface on the website. The challenge associated with front-end development is that the tools and techniques used to create the front end of a website change constantly and so the developer needs to constantly be aware of how the field is developing.

HTML AND CSS

HTML stands for HyperText Markup Language. It is used as the standard markup language for building web applications and web pages. It was originally developed by Tim Berners-Lee in 1991. HTML5 is the latest version of HTML.

CSS stands for Cascading Style Sheet. It is used to style web pages for different kinds of devices and screen sizes. Its advantages lie in its efficiency helping save time for the developer whilst creating a webpage and for the user whilst loading the aforementioned webpage. Its latest version is CSS3.

2.3 User Interface Development

A Web user interface or Web app allows the user to interact with content or software running on a remote server through a Web browser. The content or Web page is downloaded from the Web server and the user can interact with this content in a Web browser, which acts as a client. A good User Interface is integral to the success of a website because it can attract more visitors as it facilitates interactions between the user and user website or web application. An interface is a point where a user interacts with the website they're using. UI is the main part of building an engaging website.

Virtual design

Visual design improves a site's value by strategically implementing elements such as fonts, colors, and images among other things. When professionally done, visual design makes a page elegant without compromising on its function or content.

Interactive design

The interactive design looks at how users interact with technology. It uses the understanding of such interactions to create an interface with behaviors that are well-thought-out. Excellent interactive design not only anticipates how a person interacts with a system but also antedates and fixes problems in good time. It may also invent new ways through which a system interacts and responds to users.

Information Architecture

Information architecture is designed to help users find the info they need to complete

`

various tasks. It, therefore, involves labeling, structuring, and organizing the web content in a manner that makes it easily accessible and sustainable. The services and technologies used in the building of the web user interface (UI) is mentioned below.

JavaScript

JavaScript is the world's most popular programming language. JavaScript helps users in developing great front-end as well as back-end software's using different JavaScript-based frameworks like jQuery, Node.JS, React, etc.

2.4 Deployment

Software deployment includes all of the steps, processes, and activities that are required to make a software system or update available to its intended users. The general deployment process consists of several interrelated activities with possible transitions between them. These activities can occur at the server side or at the client-side or both. Today, most IT organizations and software developers deploy software updates, patches, and new applications with certain technologies mentioned below. Software deployment is one of the most important aspects of the software development process. Deployment is the mechanism through which applications, modules, updates and patches are delivered from developers to users. The methods used by developers to build, test and deploy new code will impact how fast a product can respond to changes in customer preferences or requirements and the quality of each change.

Github

GitHub provides hosting for software development and version control using Git. The best and most used features of GitHub are project management, collaborative coding, and public repositories. It offers distributed control and source code management functionality of Git, plus other features. We can easily assign work to our team members on a particular issue or problem. GitHub also helps us to keep track of the individual work and progress.

`

Chapter 3

Implementation

`

3.1 Introduction

This chapter will describe the completed process of implementation and the UML diagrams. Additionally, some code snippets of this pathfinding visualizer will also be delved into. Implementation in software development is the process of realizing an application's requirements and design. This ensures that the transformation of the software technical data package into a fabricated and tested software configuration occurs smoothly. It mainly involves mapping the design into coding in order to achieve the specifications stated for the application. The following sections define the different modules of implementation.

3.2 UML Analysis Model

Modeling involves the designing of software systems before coding takes place. Modeling plays an important role in any software development project. It guarantees the completeness and correctness of a software system and the fulfillment of end-users expectations. In addition, modeling serves as the only reference point to cross-check requirements before coding. A Unified Modeling Language (UML) based tool was used to model this application. UML diagrams give both static and dynamic views of an application and it is well suited for object-oriented languages like Java and C#. The following subsections present the UML diagrams used to model this application.

3.2.1 Use Case Diagrams

The use case diagrams for this application illustrate the interactions that exist between users (actors) and use cases (actions) within the application. There is one actor identified for this application – the user(students). As a result, there is a single-use case diagram for the software application – the user use case diagram. The user utilizes the various features of the web application such as choosing which algorithm to run and at what speed.

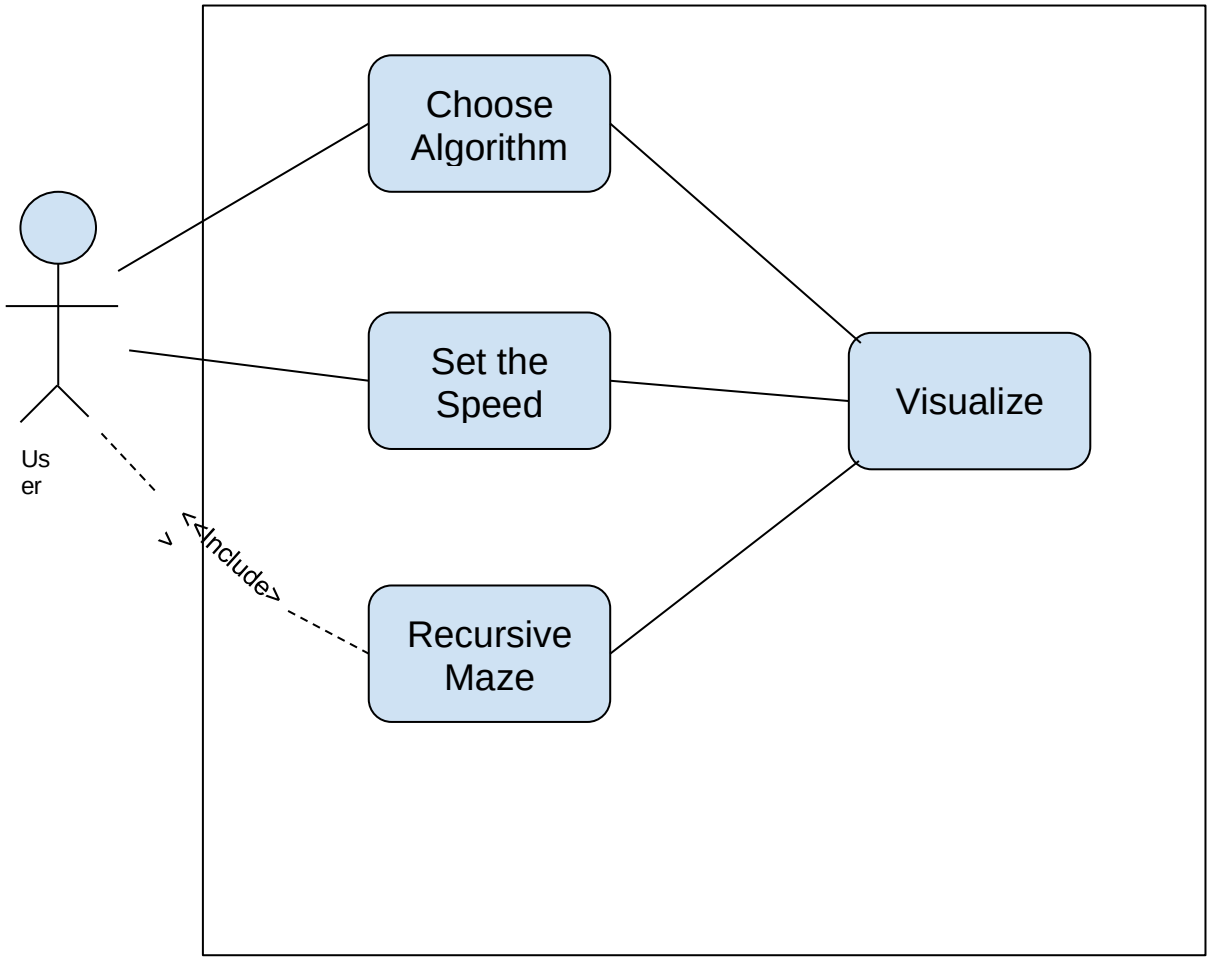


Figure 3.1- *User Use Case Diagram*

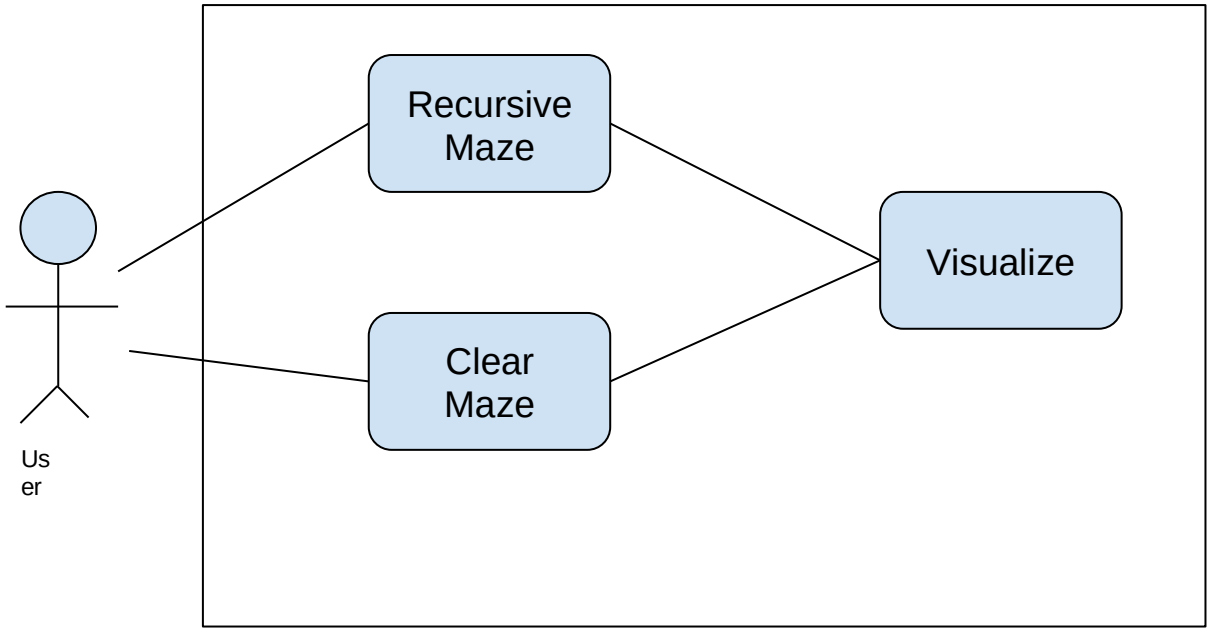


Figure 3.2- *User Case Diagram-2*

Figure 3.1 identifies the different values that can be chosen to allow the algorithm visualizer to begin. It allows the user to choose between various algorithms, set the visualizer speed, and, optionally, set up a maze between the start and target node.

Figure 3.2 depicts the functionality of the maze and how it can be added and removed depending on the wish of the user. The algorithm visualizer works with or without the maze.

3.3 Code Snippets

The code for the Algorithm Visualizer web-application was written in HTML,CSS and JAVASCRIPT using the Visual Studio code IDE. The code snippets are taken as Screen Shots of the code open in Visual Studio to explain the main concepts of implementation.

3.3.1 Home Page

The Home Page of the W/A gives the user a variety of options to use various features of W/A for understanding and visualizing various algorithms. It provides a grid display and NAV bar to view the entire features of W/A and also a demonstration page to explain how this application works.

```
16
17  <body ondragstart="return false;" ondrop="return false;">
18      <header>
19          <div class="navbar">
20              <h1 class="nav-h">
21              <i class="fa fa-cog fa-spin"></i> Algo.Visualizer
22              </h1>
23              <div class="speeddropdown">
24                  <button onclick="DropDown(event)" id='speedbtntext' class='btn-speed'>speed<i style="padding-left:10px"
25                          class="fa fa-caret-down"></i></button>
26                  <div class="speeddropdownContent">
27                      <a onclick="changeSpeed(this)" class='speed-link' href="#">Fast</a>
28                      <a onclick="changeSpeed(this)" class='speed-link' href="#">Slow</a>
29                      <a onclick="changeSpeed(this)" class='speed-link' href="#">Average</a>
30                  </div>
31              </div>
32              <div class="algodropdown">
33                  <button onclick="DropDown(event)" class='btn-algo' id='algobtntext'>Algorithm's<i
34                          style="padding-left:10px" class="fa fa-caret-down"></i></button>
35                  <div class="algodropdownContent">
36                      <a onclick="changeAlgo(this)" class='algo-link' href="#">Breadth first search</a>
37                      <a onclick="changeAlgo(this)" class='algo-link' href="#">Best first search</a>
38                      <a onclick="changeAlgo(this)" class='algo-link' href="#">Dijkstra</a>
39                      <a onclick="changeAlgo(this)" class='algo-link' href="#">A *</a>
40                  </div>
41              </div>
42              <div class="MazeDropDown">
43                  <button onclick="randomMaze()" id="mazebtntext" class='btn-maze'><span></span>Maze
44                      generation</span></button>
45                  <!--<div class="mazaDropDowncontent">
46                      <a class='maze-link' onclick="randomMaze()" href="#">Random Maze</a>
47                  </div>-->
48              </div>
49              <div class="speeddropdown" style="margin: 0;padding: 0;">
50                  <button onclick="clearAll()" class='speeddropdown btn-speed' id='btnclearswalls'
51                      style='border: none; color: rgb(0, 151, 230);'>Clear All</button>
```

Fig 3.3 shows the base code of home page

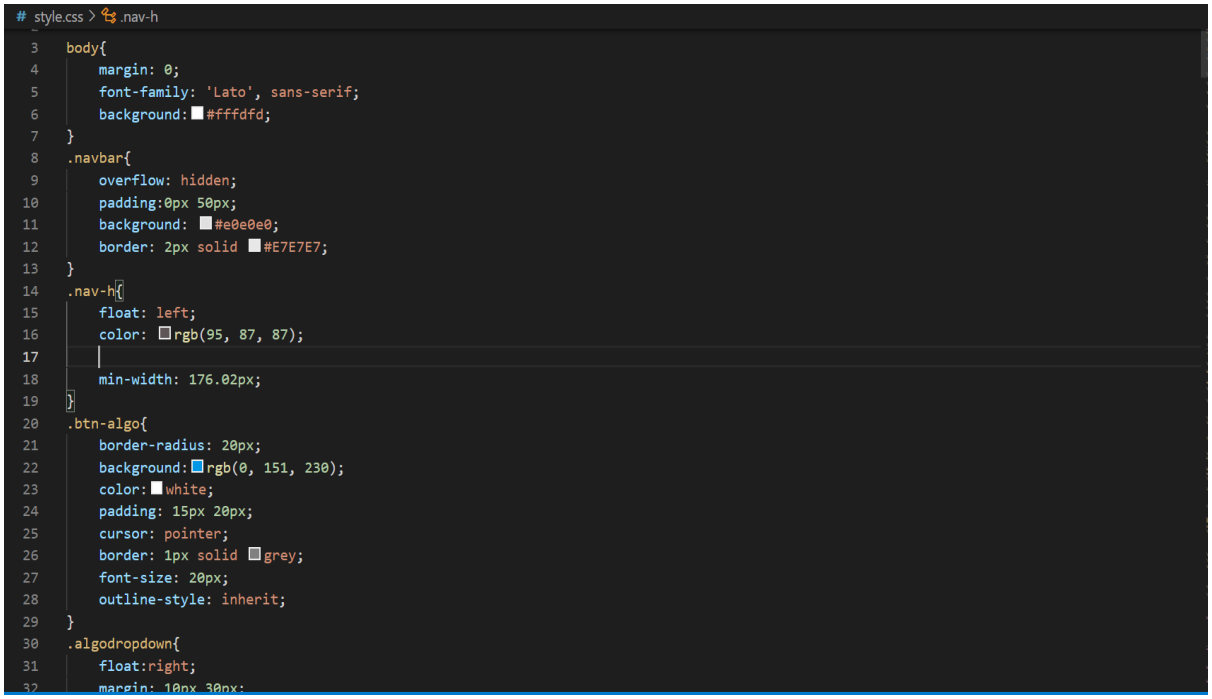
```
<div class="algodropdown">
    <button onclick="DropDown(event)" class='btn-algo' id='algobtntext'>Algorithm's<i
            style="padding-left:10px" class="fa fa-caret-down"></i></button>
    <div class="algodropdownContent">
        <a onclick="changeAlgo(this)" class='algo-link' href="#">Breadth first search</a>
        <a onclick="changeAlgo(this)" class='algo-link' href="#">Best first search</a>
        <a onclick="changeAlgo(this)" class='algo-link' href="#">Dijkstra</a>
        <a onclick="changeAlgo(this)" class='algo-link' href="#">A *</a>
    </div>
</div>
```

```
<div class="speeddropdown" style="margin: 0;padding: 0;">
    <button onclick="clearAll()" class='speeddropdown btn-speed' id='btnclearswalls'
        style='border: none; color: rgb(0, 151, 230);'>Clear All</button>
</div>
<div class="speeddropdown" style="margin: 0;padding: 0;">
    <button onclick="clearPath()" class='speeddropdown btn-speed' id='btnclearswalls'
        style='border: none; color: rgb(0, 151, 230);'>Clear Path & Visited Nodes</button>
</div>
```

```
<div class="MazeDropDown">
    <button onclick="randomMaze()" id="mazebtntext" class='btn-maze'><span></span>Maze
        generation</span></button>
    <!--<div class="mazaDropDowncontent">
        <a class='maze-link' onclick="randomMaze()" href="#">Random Maze</a>
    </div>-->
</div>
```

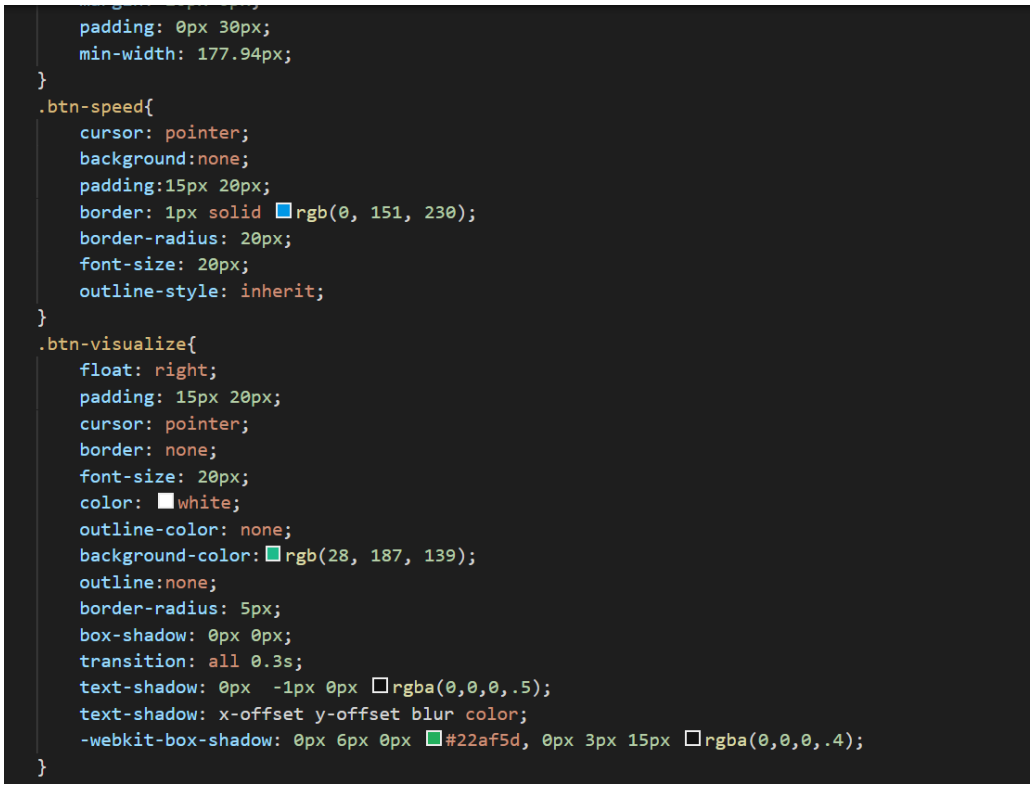
```
<div class="speeddropdown">
    <button onclick="DropDown(event)" id='speedbtntext' class='btn-speed'>speed<i style="padding-left:10px"
            class="fa fa-caret-down"></i></button>
    <div class="speeddropdownContent">
        <a onclick="changeSpeed(this)" class='speed-link' href="#">Fast</a>
        <a onclick="changeSpeed(this)" class='speed-link' href="#">Slow</a>
        <a onclick="changeSpeed(this)" class='speed-link' href="#">Average</a>
    </div>
</div>
```

Fig 3.4 shows the component of the website



```
# style.css > .nav-h
3   body{
4       margin: 0;
5       font-family: 'Lato', sans-serif;
6       background: #fffdfd;
7   }
8   .navbar{
9       overflow: hidden;
10      padding:0px 50px;
11      background: #e0e0e0;
12      border: 2px solid #E7E7E7;
13  }
14  .nav-h{
15      float: left;
16      color: rgb(95, 87, 87);
17
18      min-width: 176.02px;
19  }
20  .btn-algo{
21      border-radius: 20px;
22      background: rgb(0, 151, 230);
23      color: white;
24      padding: 15px 20px;
25      cursor: pointer;
26      border: 1px solid grey;
27      font-size: 20px;
28      outline-style: inherit;
29  }
30  .algodropdown{
31      float:right;
32      margin: 10px 30px;
```

Fig 3.5 the css code for website



```
    padding: 0px 30px;
    min-width: 177.94px;
}
.btn-speed{
    cursor: pointer;
    background:none;
    padding:15px 20px;
    border: 1px solid rgb(0, 151, 230);
    border-radius: 20px;
    font-size: 20px;
    outline-style: inherit;
}
.btn-visualize{
    float: right;
    padding: 15px 20px;
    cursor: pointer;
    border: none;
    font-size: 20px;
    color: white;
    outline-color: none;
    background-color: rgb(28, 187, 139);
    outline:none;
    border-radius: 5px;
    box-shadow: 0px 0px;
    transition: all 0.3s;
    text-shadow: 0px  -1px 0px rgba(0,0,0,.5);
    text-shadow: x-offset y-offset blur color;
    -webkit-box-shadow: 0px 6px 0px #22af5d, 0px 3px 15px rgba(0,0,0,.4);
}
```

Fig 3.6 the css code for website

3.3.2 Create Grid

In Fig 3.11 the code snippet is for the function when the user visits the website and grid with start node and end node is already present there and user can move them according to his choice and visualize various algorithms on it .

```
function make_Grid() {
    let cR = "";
    var nodeNo = 0;
    for (var i = 0; i < HEIGHT; i++) {
        cR += "<tr id='row-" + i + "'>";
        for (var k = 0; k < WIDTH; k++) {
            if (nodeNo == 655) {
                cR += "<td id='" + i + "-" + k + "' draggable='false' onclick='createWall(event)' class='start'></td>";
            } else if (nodeNo == 678) {
                cR += "<td id='" + i + "-" + k + "' draggable='false' onclick='createWall(event)' class='end' style=''></td>";
            }
            else {
                cR += "<td id='" + i + "-" + k + "' draggable='false' onclick='createWall(event)' class='unv'></td>";
            }
            nodeNo++;
        }
        cR += "</tr>"
    }
    gridTable.innerHTML = cR;
}
```

Fig 3.7 Code Snippet for create grid

3.3.3 code for finding the shortest path

Fig 3.12 shows the code for finding the shortest path using the user chosen algorithm and animate it .

```
function findMinDistanceInGraph(graph) {
    let len = graph.length;
    let min = Infinity;
    while ((len--) != 0) {
        if (graph[len].distance < min)
            min = graph[len].distance;
    }
    return graph.find(element => element.distance == min);
}
```

Fig 3.8 Code Snippet for shortest path

3.3.4 button for creating maze

```
<div class="MazeDropDown">
    <button onclick="randomMaze()" id="mazebtntext" class='btn-maze'><span></span>Maze
        generation</span></button>
    <!--<div class="mazaDropDowncontent">
        <a class='maze-link' onclick="randomMaze()" href="#">Random Maze</a>
    </div>-->
</div>
```

```
function randomMaze() {
    if (listM.length > 0) {
        clearMaze(listM, 0);
        listM.length = 0
    }
    const listNodes = document.getElementsByClassName('unv');
    let len = listNodes.length - 1;

    for (var i = 0; i < len / 4; i++) {
        listM.push(listNodes[(Math.floor(Math.random() * len))].id)
    }
    Animation(listM, 0)
}
```

Fig 3.9 shows the code for creating maze

3.3.5 Clearing all button

Fig 3.14 shows the code for removing all animations from the grid by the user by pressing the “Clear all ” button in the NAV.

```
function clearAll() {
    if (!viz) {
        gridTable.style.pointerEvents = 'all';
        startDrag = false;
        endDrag = false;
        visualizing = false;
        ListOfNodes.forEach(element => {
            if (element.visited && element.id != start) {
                document.getElementById(end).classList.remove('shortestPath');
                document.getElementById(element.id).classList.remove('visited');
                document.getElementById(element.id).classList.remove('shortestPath');
                document.getElementById(element.id).classList.add('unv');
            } else if (element.wall && element.id != start) {
                document.getElementById(element.id).classList.remove('wall');
                document.getElementById(element.id).classList.add('unv');
            } else {
                document.getElementById(start).classList.remove('shortestPath');
                document.getElementById(element.id).classList.remove('visited');
            }
        });

        ListOfNodes.length = 0;
        visiteds.length = 0;
        vis.length = 0;
        path.length = 0;
    }
}
```

Fig 3.10 Code Snippet for “Clear all” button

3.3.6 Clearing Path button

Fig 3.14 shows the code for removing all animations of shortest path from the grid by the user by pressing the “Clear Path ” button in the NAV.

```
function clearPath() {
    if (ListOfNodes.length > 0 && !viz) {
        gridTable.style.pointerEvents = 'all'
        visualizing = false;
        ListOfNodes.forEach(element => {
            if (element.visited && element.id != start) {
                document.getElementById(end).classList.remove('shortestPath');
                document.getElementById(element.id).classList.remove('visited');
                document.getElementById(element.id).classList.remove('shortestPath');
                document.getElementById(element.id).classList.add('unv');
            } else {
                document.getElementById(start).classList.remove('shortestPath');
            }
        });
        vis.length = 0;
        path.length = 0;
    }
}
```

Fig 3.11 Code Snippet for “Clear Path” button

3.3.7 Various Algorithms

Dijkstra's Algorithm (weighted): the father of pathfinding algorithms; guarantees the shortest path.

```
//algorithms
function dijkstra() {
    prev = [];
    visited = [];

    start = document.getElementsByClassName("start")[0].id;
    end = document.getElementsByClassName("end")[0].id;

    let Unvnodes = document.getElementsByTagName('td');

    ListOfNodes = getAllNodes(Unvnodes);
    temp = ListOfNodes;
    updateDistance(getNodeById(start), 0)
    var finished = null;

    while (temp.length != 0) {
        //get min distance node
        let current = findMinDistanceInGraph(temp);
        if (current.id == getNodeById(end).id) {
            //finished becomes the last visited node
            finished = current;
```

```
    ListOfNodes = getAllNodes(Unvnodes);

    let visited = [];

    let finished = null;
    visited.unshift(getNodeById(start));

    while (visited.length != 0) {
        let currentNode = visited.pop();
        currentNode.visited = true;

        if (currentNode.id == end) {
            finished = currentNode;
            break;
        }
        vis.push(currentNode);
        let neighbours = getNeighboursForUnweighted(currentNode);

        for (var i in neighbours) {
            neighbours[i].visited = true;
            neighbours[i].parent = currentNode; //backtracing
            visited.unshift(neighbours[i]);
        }
    }
    while (finished != null) {
        path.unshift(finished);
        finished = finished.parent
    }
    return [vis, path]
}
```

Fig 3.12 Code Snippet for “Dijkstra Algorithm” animation

Astar Algorithm (weighted): arguably the best pathfinding algorithm; uses heuristics to guarantee the shortest path much faster than Dijkstra's Algorithm.

```
function bredthFirstSearch() {
    let vis = [];
    let path = [];

    start = document.getElementsByClassName("start")[0].id;
    end = document.getElementsByClassName("end")[0].id;

    let Unvnodes = document.getElementsByTagName('td');

    ListOfNodes = getAllNodes(Unvnodes);

    let visited = [];

    let finished = null;
    visited.unshift(getNodeById(start));

    while (visited.length != 0) {
        let currentNode = visited.pop();
        currentNode.visited = true;
```

```
function AStar() {
    let vis = [];
    let path = [];

    start = document.getElementsByClassName("start")[0].id;
    end = document.getElementsByClassName("end")[0].id;
    let Unvnodes = document.getElementsByTagName('td');

    ListOfNodes = getAllNodes(Unvnodes);

    sNode = getNodeById(start);
    sNode.distance = 0;
    enode = getNodeById(end);
    pq = new PriorityQueue();
    pq.enqueue(sNode, 0)
    var finished = null;
    while (!pq.isEmpty()) {
        let u = pq.dequeue();
        u.element.visited = true;
        vis.push(u.element);
        // goal found
        if (u.element.id == end) {
            finished = u.element;
            break;
        }
        let neighbours = getNeighboursForGreedy(u.element);
```

Fig 3.13 Code Snippet for “A*star Algorithm” animation

`

Breadth first search (weighted): the father of pathfinding algorithms; guarantees the shortest path.

```
function bredthFirstSearch() {
    let vis = [];
    let path = [];

    start = document.getElementsByClassName("start")[0].id;
    end = document.getElementsByClassName("end")[0].id;

    let Unvnodes = document.getElementsByTagName('td');

    ListOfNodes = getAllNodes(Unvnodes);

    let visited = [];

    let finished = null;
    visited.unshift(getNodeById(start));

    while (visited.length != 0) {
        let currentNode = visited.pop();
        currentNode.visited = true;


        if (currentNode.id == end) {
            finished = currentNode;
            break;
        }
        vis.push(currentNode);
        let neighbours = getNeighboursForUnweighted(currentNode);

        for (var i in neighbours) {
            neighbours[i].visited = true;
            neighbours[i].parent = currentNode; //backtracing
            visited.unshift(neighbours[i]);
        }
    }
    while (finished != null) {
        path.unshift(finished);
        finished = finished.parent
    }
    return [vis, path]
}
```

Fig 3.14 Code Snippet for “Breadth for search Algorithm” animation

`

Best first search (weighted): a faster, more heuristic-heavy version of A*; does not guarantee the shortest path.

```
function bestFirstSearch() {
    let vis = [];
    let path = [];
    start = document.getElementsByClassName("start")[0].id;
    end = document.getElementsByClassName("end")[0].id;
    let Unvnodes = document.getElementsByTagName('td');

    ListOfNodes = getAllNodes(Unvnodes);

    pq = new PriorityQueue();

    Snode = getNodeById(start)
    pq.enqueue(Snode, Snode.distance);

    endNode = getNodeById(end);

    while (!pq.isEmpty()) {
        let u = pq.dequeue();
```

```
        vis.push(u.element);

        u.element.visited = true;

        let neighbours = getNeighboursForGreedy(u.element);

        if (u.element.id == end) {
            finished = u.element;
            break;
        }
        for (var i in neighbours) {
            neighbours[i].visited = true;
            let distance = calculateSignleLineDistance(endNode, neighbours[i]);
            neighbours[i].distance = distance;
            neighbours[i].parent = u.element;
            pq.enqueue(neighbours[i], neighbours[i].distance);
        }
    }
    while (finished != null) {
        path.unshift(finished);
        finished = finished.parent;
    }
    return [vis, path]
}
```

Fig 3.15 Code Snippet for “Best for search Algorithm” animation

`

3.4 Summary

This chapter was about showing and Explaining the Use Case diagrams for the administrator as well as the general user, the Class Diagrams to show the interaction between different parts of the system, also the admin login sequence diagram. Code Snippets have been provided for further showing the implementation of various functionalities.

`

Chapter 4

Result and Testing

4.1 TESTING

Software testing is carried out on a software application mainly to detect software bugs or missing requirements of the application. It involves the process of investigating and evaluating a software product or application in order to make sure that its business and technical requirements are fulfilled. For the purpose of testing, this application was deployed and accessed on a web browser. For simplicity, the testing template in Table 4.1 was used for the testing.

Table 4.1 Software testing template and results.

S/N	Test Description	Steps	Expected System Response	Status
1.	Access home page of the application to view grids and demonstration.	Enter the home page URL of the application on any web browser.	The home page of the application is displayed with demonstration.	Pass.
2.	Click and drag mouse on grid to make walls.	Click on any grid and drag the mouse in any direction to make walls.	Walls are built on the grid according to dragging of mouse.	Pass.
3.	After making walls, select any algorithm out of the four	Click on the algorithm button and a drop-down menu will be displayed and choose one.	On clicking algorithm, a drop-down menu is displayed with four options (algorithms).	Pass.

4.	Select the speed.	Click on the speed button which is nearby algorithm button and choose one.	A drop-down menu is displayed with three options.	Pass.
5.	Generate the maze before visualization.	Click on the maze generation button.	A maze is generated in addition to walls already built on grids.	Pass.
6.	Find the number of visited nodes and path nodes.	Click on the visualize button.	Selected algorithm gets started to visualize and number of visited and path nodes are displayed.	Pass.
7.	Visualization of current wall with same algorithm but other speed.	Click on the clear path and visited nodes button and click on speed button to choose new one.	Visualization of same algorithm gets started with new speed.	Pass.
8.	Visualization of current walls with other types of algorithms.	Click on the clear path and visited nodes button and then click on algorithm button to select one.	Visualization of the selected grid and generated maze through new algorithm.	Pass.
9.	Clear the current wall and maze for new generation.	Click on the clear all button.	Every path nodes and visited nodes gets cleared along with grid walls.	Pass.

10.	Regeneration of grid walls after previous one gets cleared.	Click on any grid and drag the mouse in any direction to generate walls for visualization.	A new generation of grid walls is displayed.	Pass.

11.	Requirement of maze generation for a better visualization.	Click on the maze generation button.	A newly generated maze gets displayed.	Pass.
12.	Selection of type of algorithm for visualization of generated maze and grid wall.	Click on the algorithm button and choose one from the drop-down menu.	After clicking on button, a drop-down menu is displayed with four options.	Pass.
13.	Selection of speed of visualization.	Click on the speed button.	When button gets clicked, a drop-down menu is displayed containing three options.	Pass.
14.	Initialize the visualization process.	Click on the visualize button.	Visualization process gets initiated after clicking on the required button.	Pass.
15.	Determine the visited and path nodes.	After clicking on the visualize button, wait for it to get over.	The number of visited and path nodes are displayed along with the graphical display.	Pass.
16.	Change the speed and algorithm for current maze and grid walls.	Click on the clear visited and path nodes button and select the algorithm and speed from the respective buttons.	A new algorithm and speed gets chosen from the respective drop-down menus.	Pass.
17.	Visualization of newly selected algorithms and speed.	Click on the visualize button.	Visualization of the current walls and maze through new algorithms and speed.	Pass.

18.	Clear all the visited nodes, path nodes and grid walls.	Click on the clear all button.	Everything gets cleared from visited and path nodes to walls and maze.	Pass.

4.2 RESULT AND SCREENSHOTS

After successfully compiling the Final Code and Deploying the Algorithm Visualizer Web application the final outputs achieved are shown below.

Demonstration page

Fig 4.1 shows the demonstration page of the web application which displays demonstration in starting. All the buttons like algorithm, speed, etc. can also be seen here.

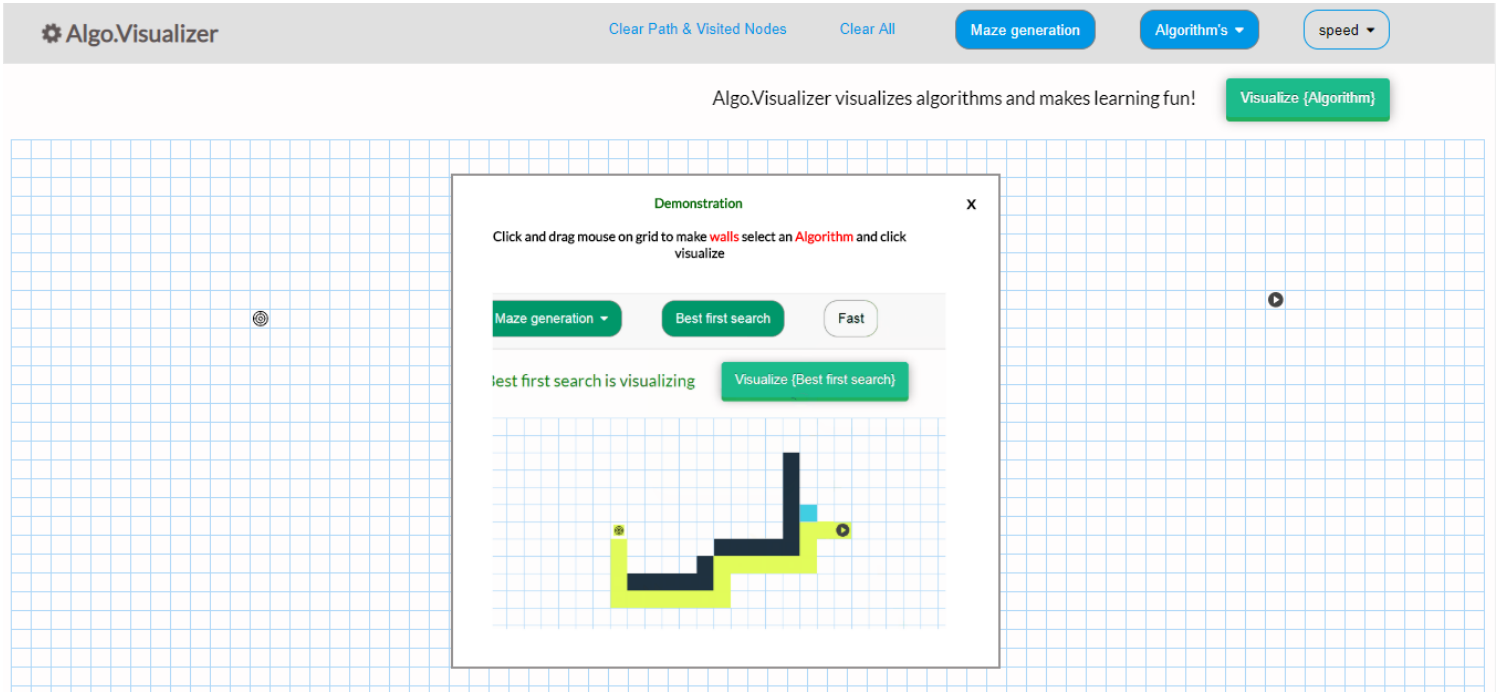


Fig 4.1 Demonstration page

Wall generation on grid

Fig 4.2 shows the output when any of the grid is clicked and mouse is dragged in any direction on the grid to generate a wall for the visualization of algorithms.

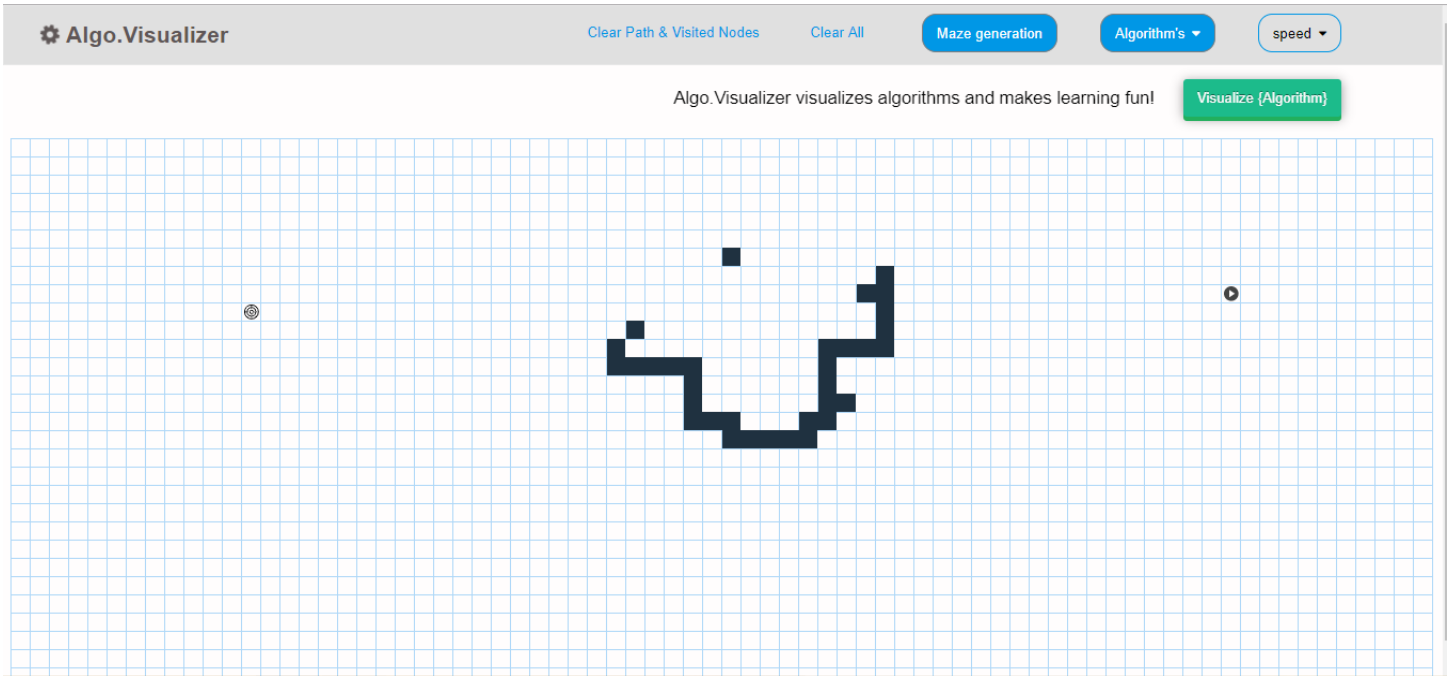


Fig 4.2 wall generation

Maze Generation

Fig 4.3 shows the output of the generated maze after the maze generation button is clicked.

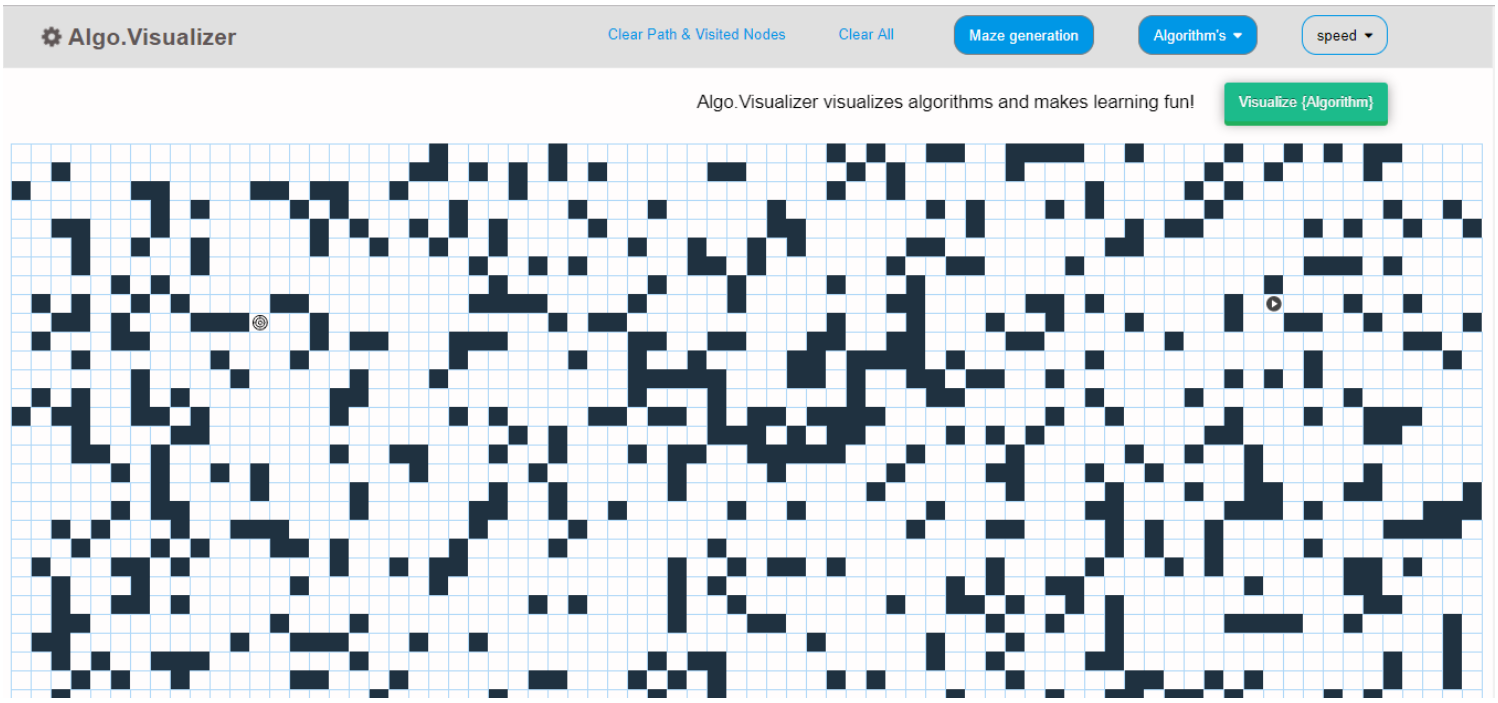


Fig 4.3 Maze generation

Breadth first search algorithm

Fig 4.4 shows the output of visualization of generated maze and grid walls through breadth first search algorithm in fast speed.

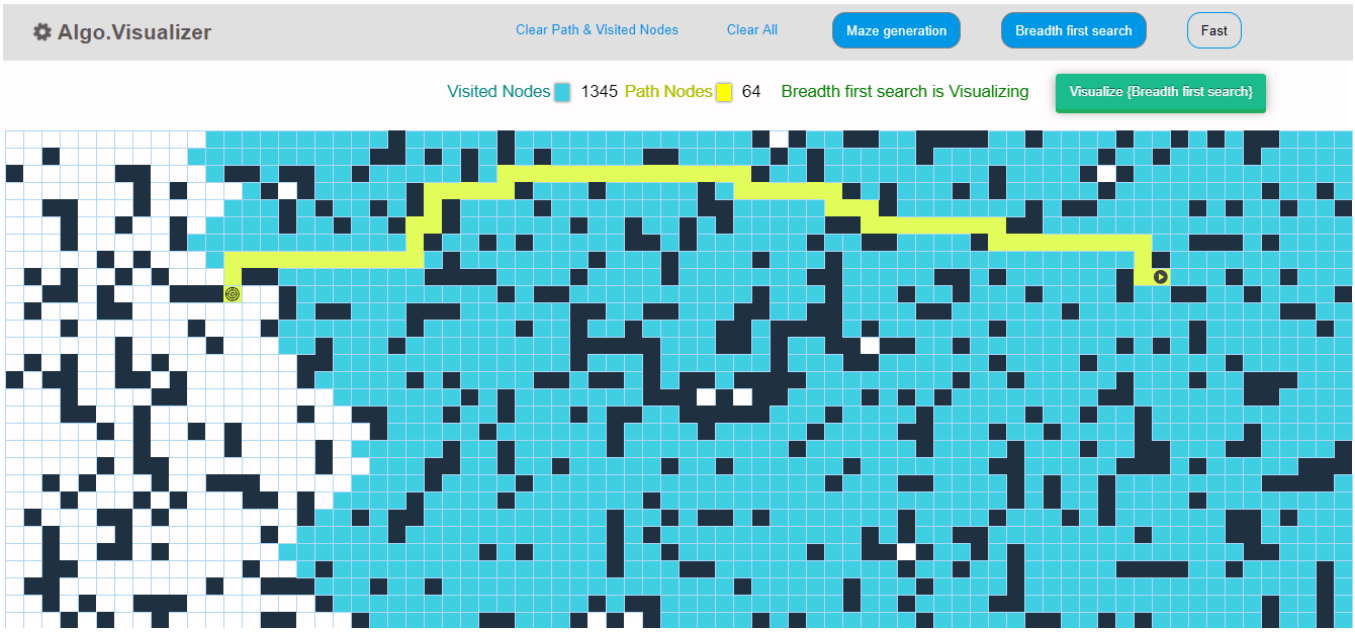


Fig 4.4 Breadth first search (fast)

Best first search algorithm

Fig4.5 shows the output of visualization of generated maze and grid walls through best first search algorithm in fast speed.

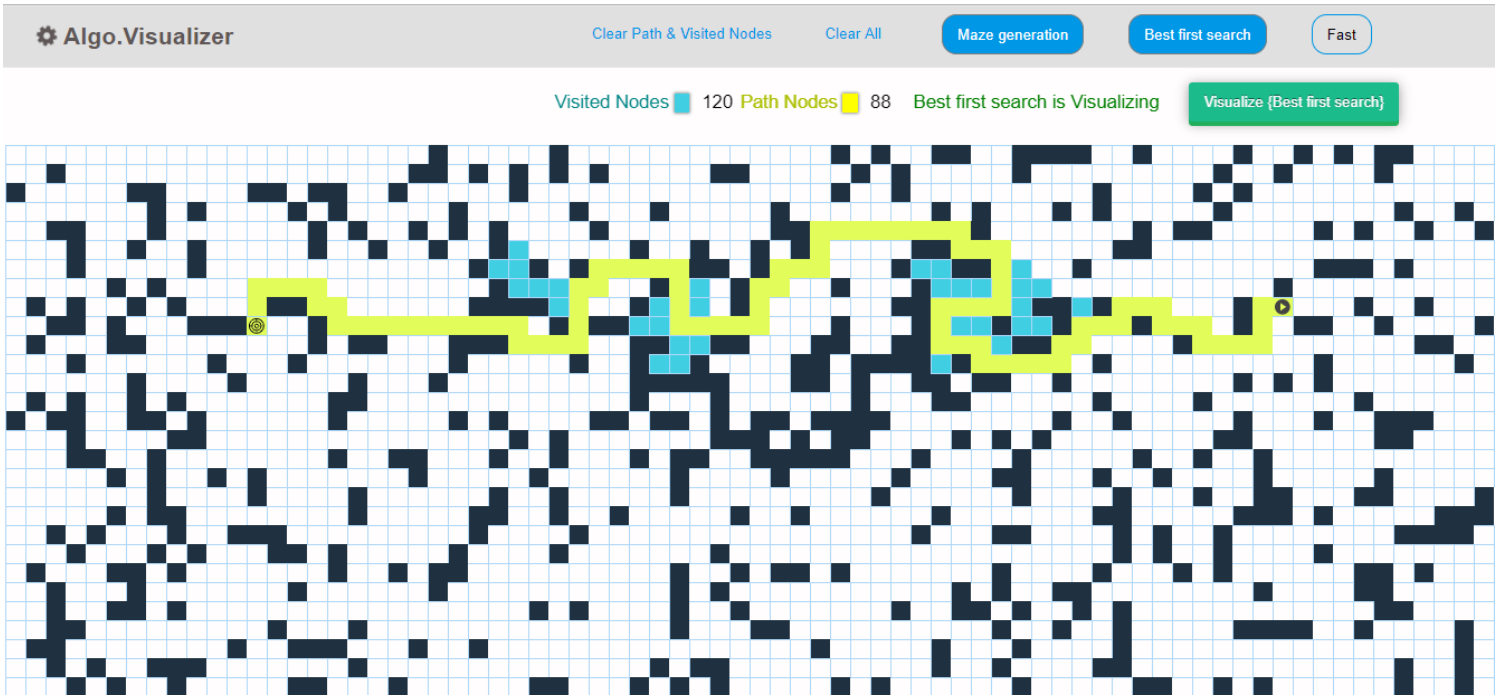


Fig 4.5 Best first search algorithm

`

Dijkstra Algorithm

Fig4.10 shows the output of visualization of generated maze and grid walls through dijkstra algorithm at a fast speed.

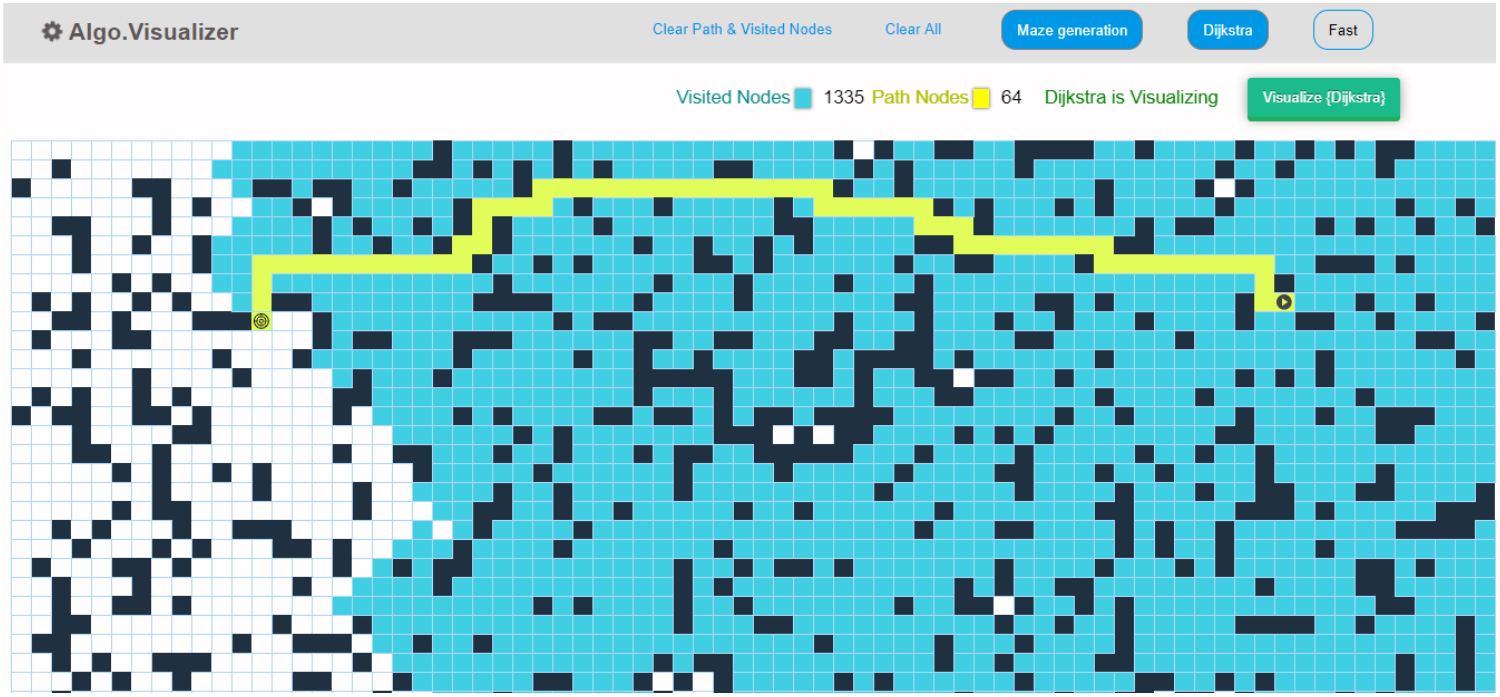


Fig 4.6 Dijkstra algorithm

`

A* Algorithm

Fig4.13 shows the output of visualization of generated maze and grid walls through dijkstra algorithm at a fast speed.

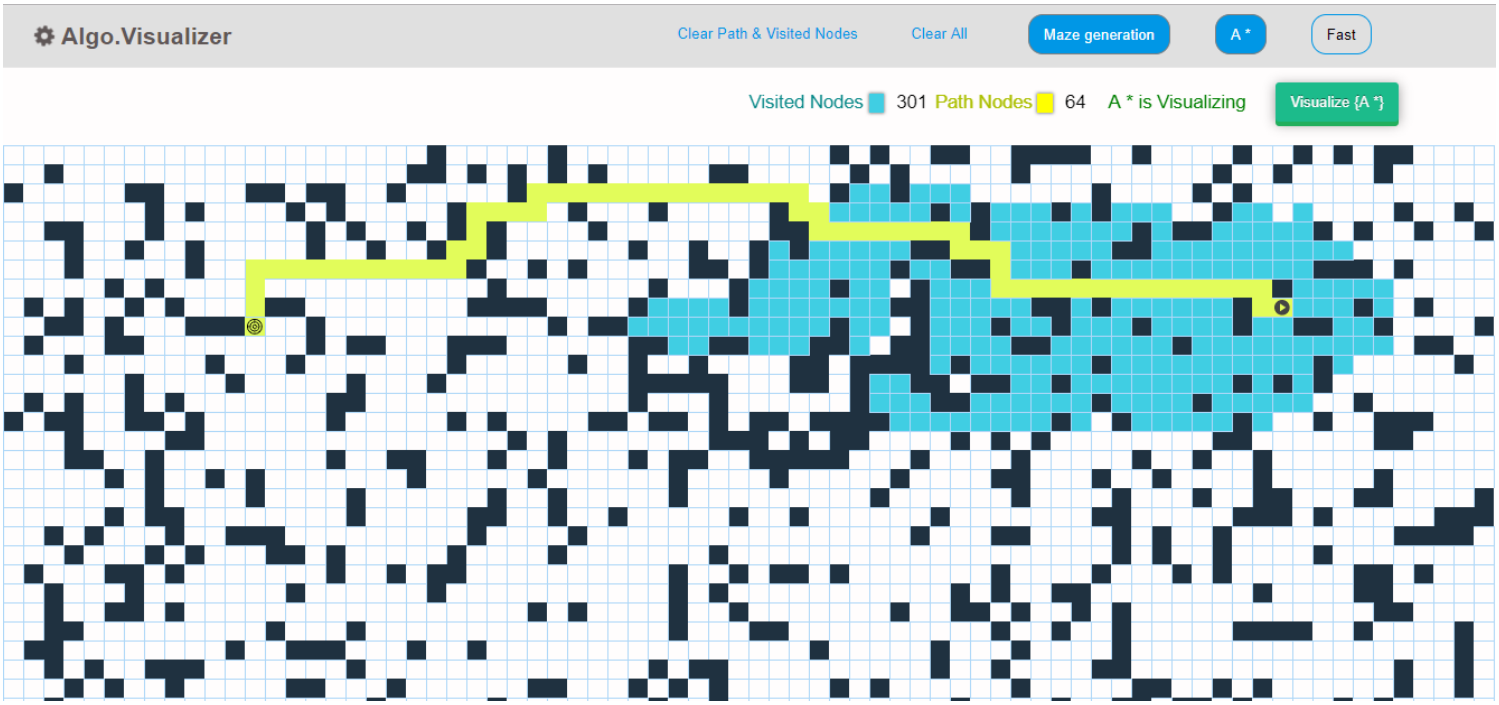


Fig 4.7 A* Algorithm

Cleared maze and grid walls

Fig4.16 shows the output of the situation when the clear button is clicked and all the maze generated along with visited and path nodes get cleared.

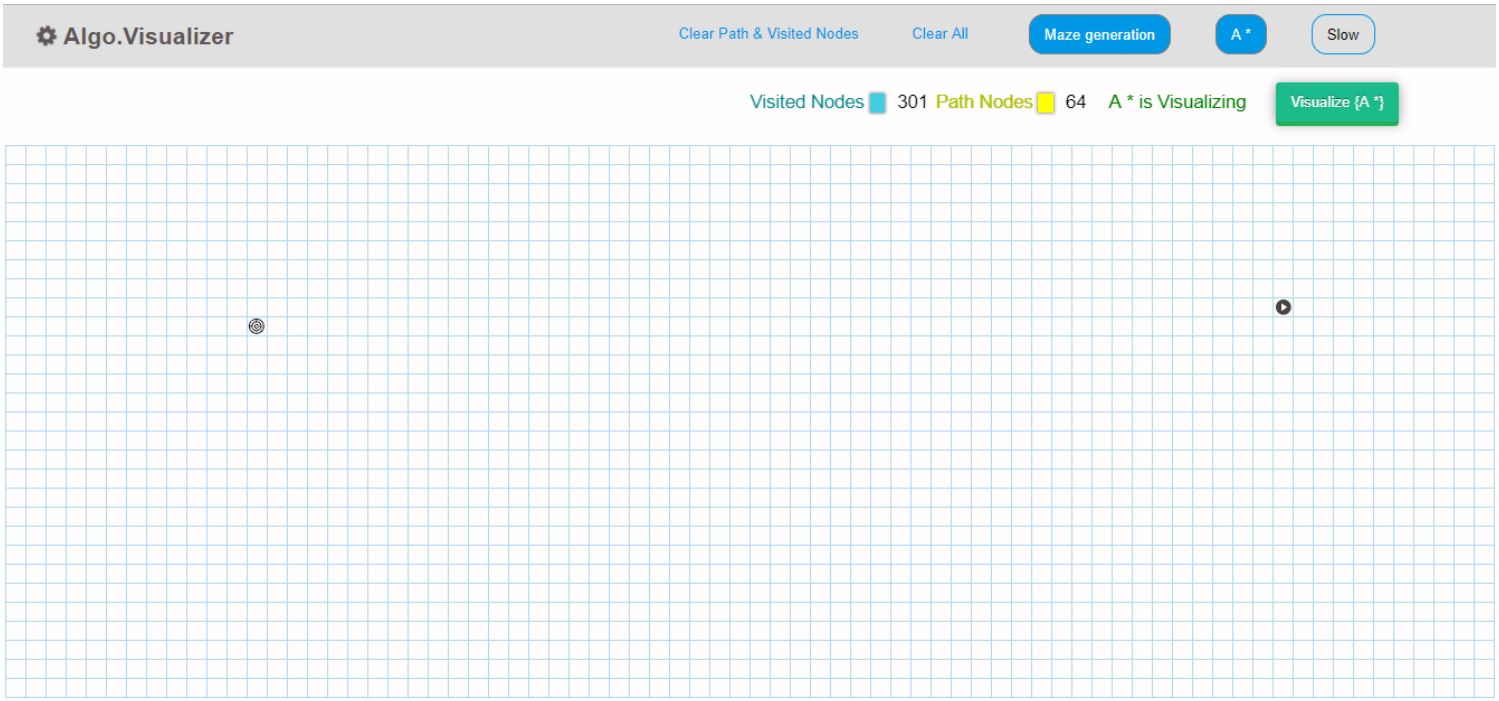


Fig 4.8 Cleared maze and grid walls

4.3 CONCLUSION AND FUTURE SCOPE

The main goal of this project is to learn and improve the understanding of the path-finding algorithms at free of cost to every student of the world. It is taken care in this project that students improve their approach to real-world problems and their creativity can be improved a lot.

The prime objective of this project is create an easy-to-use or user-friendly interface with some most popular path-finding algorithms. This will help overcome the drawbacks of virtual learning of algorithms for the users.

Challenges

One of the biggest challenges faced during the development of the W/A was integration of the concept of number of visited nodes and path nodes in the algorithm. It seemed to be somewhat more difficult than other parts of coding but managed with that by finding output of it again and again.

`

Future Scope

Although the requirements set out for the web application have been met, there are still some future scopes and areas which can be improved later on. A mobile version can be developed for the application so that the users can have access to the application from their mobile phones as well. Current web applications are developed using HTML, CSS, Bootstrap and JavaScript so there is a scope of adding backend to it. It can be achieved through the Node.js environment and it can lead to increased productivity of the web application.

References

1. https://www.brainkart.com/article/Algorithm-Visualization_8008/
2. Blass, Andreas; Gurevich, Yuri (2003). "Algorithms: A Quest for Absolute Definitions" (PDF). *Bulletin of European Association for Theoretical Computer Science*
3. https://content.techgig.com/6-things-front-end-developers-should-learn-in-2020/articleshow/76496481.cms
4. https://medium.com/digio-australia/user-interface-development-in-the-next-5-10-years-f19edcd6d0b0
5. https://medium.com/edureka/continuous-deployment-b03df3e3c44c

Appendix

GITHUB: https://github.com/jacksonjj2-0/JJ