AI-PATHFINDING -VISUALIZER

Jumail j (32020956026)

UNIVERSITY INSTITUTE OF TECHNOLOGY
PIRAPPANCODE
THIRUVANANTHAPURAM

PROJECT REPORT

Submitted in partial fulfillment of the
Requirements for the award of
BSc (computer science) degree of
University of Kerala

2022

UNIVERSITY INSTITUTE OF TECHNOLOGY
UNIVERSITY OF KERALA

Pirappancode, Thiruvananthapuram

CERTIFICATE

Certified that this report titled "**AI Pathfinding Visualizer**" is a bonafide record of the project work done by **Jumail.j (32020956026)**. Under our supervision and guidance, towards partial fulfillment of the requirements for the award of the Degree of BSC(Computer Science) of the University of Kerala.

Internal Guide

Mrs. Vineetha R V
(Lecturer in Computer Science)

Dr. Sreedevi S
Principal

External Examiner

Date:

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and We are extremely privileged to have got this all along the completion of our project. All that we have done is only due to such supervision and assistance and We would not forget to thank them.

We respect and thank **Dr. Sreedevi P** Principal, University Institute of Technology for providing me an opportunity to do the project work in Computer Science and giving us all support and guidance. Which made us complete the project duly. We are extremely thankful to her for providing such nice support and guidance.

We owe our deep gratitude to our project guide **Mrs. Vineetha R V,** who took keen interest in our project work and guided us all along. Till the completion of our project work by providing All necessary information for developing a good system.

We are thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching staff of the Computer Science Department. Our sincere thanks and apologies to anyone who deserves the credit but whose names fail to appear in the list above.

Sincerely,

Jumail J

DECLARATION

We hereby declare that the work presented in the project **Ai PathFiding Visualizer**
On the original work is down by us under the guidance of **Mrs. VINEETHA R V**
(Lecturer in computer science). The same work had not been submitted elsewhere for another degree for the best of our knowledge.

Place: Pirappancode
Date:

Jumail J

ABSTRACT

Visualization is an efficient way of learning any concept faster than conventional methods. Modern technology allows creating e-Learning tools that also helps in improving computer science education very much. Algorithm Visualizations contribute to improving computer science education. The method of teaching and learning algorithms is commonly complex to understand the problem.
The goal of this project is to create a web based 'PathFinding Visualizer', which can be used to visualize shortest path algorithms. The conceptual application of the project is illustrated by implementation of algorithms like **BFS** (Breadth First Search) ,**Bidirectional Breadth First** and **A***. Path Finding algorithms are build on top of **Graph search algorithms** and explore routes between nodes, starting at one node and traversing through relationships until the destination has been reached

This project aims to develop with **Html, Css, and Javascript** used to design the user interface, version controlled by **git.**

CONTENTS

1. **INTRODUCTION……..**

2. **SYSTEM ANALYSIS……**
   - 2.1 EXISTING SYSTEM……...
   - 2.2 PROPOSED SYSTEM……..
   - 2.3 FEASIBILITY STUDY……

3. **SYSTEM DESIGN……..**
   - 3.1 MODULE……..……
   - 3.2 DATA FLOW DIAGRAM……
   - 3.3 USE CASE DIAGRAM……

4. **SYSTEM REQUIREMENT……...**
   - 4.1 HARDWARE REQUIREMENTS……
   - 4.2 SOFTWARE REQUIREMENTS……..
   - 4.3 TECHNOLOGY USED……

5. **SOFTWARE DESIGN……**
   - 5.1 INPUT DESIGN……...
   - 5.2 OUTPUT DESIGN……...
   - 5.3 USER INTERFACE DESIGN……..

6. **CODE……...**

7. **SYSTEM TESTING……**

8. **SYSTEM IMPLEMENTATION……**

9. **MAINTENANCES……..**

10. **CONCLUSION……**

11. **APPENDIX……..**
    - 12.1 SCREENSHOT……...
    - 12.2 GANTT CHART……

12. REFERENCE……..

1.INTRODUCTION

Algorithm visualization illustrates how algorithms work in a graphical way. By Graphically representing computer algorithms in action, algorithm visualization technology aims to help computer science students understand how algorithms work. Algorithms are an essential part of knowledge in a framework of computer science.

Nowadays PathFinding algorithms are widely used in computer softwares.
Pathfinding algorithms are usually an attempt to solve the shortest path problem in graph theory. At core a pathfinding method searches a graph by starting at one vertex and exploring adjacent nodes until the destination node is reached. Generally with the intent of finding the cheapest route.
For example, they are used in applications like **google maps**, **Satellite Navigation systems**, **routing packets over the internet**.

The main goal of the project was to create a program which would serve as a tool for understanding how most known pathfinding algorithms work. There was an attempt to make the best possible user experience. The demonstration software is made in a user-friendly and easy-to-use style.

The entire system is developed using **html**, **css**, **javascript** and version controlled by **git.** The proposed system help in understanding the working of algorithms like **BFS** (Breadth First Search) ,**Bidirectional Breadth First** and **A*.**

2.SYSTEM ANALYSIS

A system is simply a set of components to accomplish an objective. Developing a new system, investigating into the operation and making possible changes in the existing system are called System Analysis. Analysis comprises a detailed study of the various operations performed by a system and their relationships within and outside the system. It is the process of gathering and interpreting facts, diagnosing problems and improving the system using the information obtained.

2.1 Existing System

FlowCharts:

Flowcharts in the classroom are graphical representations of students' thinking processes. It allows students to chalk out their ideas and thoughts in a logical and organized fashion, giving them the freedom to come back and reflect on it. With this method, a student is able to describe a sequence of events or actions in a step by step fashion leading to its outcome.

While drawing a flowchart, one starts with their first thought or action point and draws it in a box. The remaining action points are placed one by one in boxes in a sequential manner. Arrows connect the boxes in the right order. Ensuring that the boxes are in order is an important element in drawing a flowchart as the idea changes according to the order

Drawback of flowcharts:

- Difficulty in presenting complex programs and tasks
- No scope for alteration or modification
- Reproduction become a problem
- Factors that affect the sequence are not depicted.
- No man to computer communication.
- The complex logic of the program logic is quite complicated to draw out on by using different defined shapes.
- The essentials of what is done can easily be lost in the technical details of how it is done.

Pseudo codes:

Pseudocode is a way of **expressing an algorithm without conforming to specific syntax rules**. It is an informal way of programming description that does not require any strict programming language syntax or underlying technology considerations. It is used for creating an outline or a rough draft of a program. Pseudocode summarizes a program's flow, but excludes underlying details.
By learning to read and write pseudocode, you can easily communicate ideas and concepts to other programmers, simply we can say that it's the cooked up representation of an algorithm. Often, algorithms are represented with the help of pseudo codes as they can be interpreted by programmers no matter what their programming background or knowledge is.
'**Pseudo codes'** as the name suggests, is a false code.

Drawback of Pseudo codes:

- It's not visual
- Create an additional level of documentation to maintain.
- We do not get a picture of the design
- There is no standardized style of format, so one pseudocode may be different from another.
- For a beginner, it is more difficult to follow the logic or write pseudocode as compared to flowchart.

2.2 Proposed System

Within the paper we discuss the possibility of enriching the standard method of teaching algorithms, there is yet another way to study algorithms. It is called algorithm visualization and can be defined as the use of images to convey some useful information about algorithms.That information can be a visual illustration of an algorithm's operation. It mainly aims to simplify and deepen the understanding of algorithm operation. Of its performance on different kinds of inputs. Or of its execution speed versus that of other algorithms for the same problems. To accomplish this goal,
An algorithm visualization uses graphic elements points, line segments, 2-Dimensional grid and bars, and so on to represent some "interesting events" in the algorithm's operation.
In pathfinding making the starting and the end node be able to move around or the user to choose wherever he/she wants it to start or end. The developed e-learning tool visualizes the algorithm rule steps executions.

2.3 Feasibility study

A feasibility study is really a small scale systems analysis. It differs from a full analysis only in its level of detail. The study involves analysts in most of the tasks of a full system analysis but with a narrower focus and more limited time. The result of the study helps the user to decide whether to proceed, amend, postpone or cancel the project particularly important when the project is large, complex and costly. However, a feasibility study is no substitute for a full, detailed and thorough analysis of a system.

In focuses on these major questions:

- What are the user's demonstrable needs and how does a candidate system meet them?
- What resources are available for a given candidate system?
- What are the likely impacts of the candidate system on the organization?
- Is it worth solving the problem?

During feasibility analysis for this project, following primary areas of interest are to be considered. Investigation and generating ideas about a new system does this.

- From a project team and appoint a project leader. Prepare system flowcharts
- Entamerate potential proposed system
- Define and identify characteristics of the proposed system.
- Determine and evaluate performance and cost effective of each proposed system. Weight system performance and cost data - Select the best-proposed system. Prepare and report the final project directive to management.

- **Technical feasibility**
- **Operational feasibility**
- **Functional feasibility**
- **Economical feasibility**

Technical feasibility
A study of resource availability that may affect the ability to achieve an acceptable system. This evaluation determines whether the technology needed for the proposed system is available or not.

- Can the work for the project be done with current equipment, existing software technology & available personnel?
- Can the system be upgraded if developed?
- If new technology is needed then what can be developed
- This is concerned with specifying equipment and software that will successfully satisfy the user requirement.

Operational feasibility

It is mainly related to human organizations and political aspects. The points to be consider are:

- What changes will be brought with the system?
- What organization structures are distributed?
- What new skills will be required?

The system is operationally feasible as it is very easy for the end users to operate it. It only needs basic information about the Windows platform.

Functional feasibility

Here we should examine the functions of the system which may work properly when it will be implemented. The proposed system does not involve complex protocols. It has a simple working style. A basic knowledge of computers is all that is required. Hence the proposed system is functionally feasible.

Economical Feasibility

The proposed system will not cause any expenditure since all the requirements are available in the organization. The organization is already having a group of express who can undertake this task without any difficulty

3.SYSTEM DESIGN

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. It implies a systematic approach to the design of a system. It may take a bottom-up or top-down approach.

3.1 Module

- **User**
    - View Grid
    - Add/Remove wall nodes
    - Move Start node and End node
    - Select different pathfinding algorithm
    - Generate Wall
    - Adjust speed

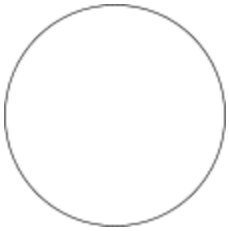
3.2 Data Flow Diagram (DFD)

A DFD is a network that describes the flow of data throughout a system, data stores, and the process that changes or transforms data flows. Data Flow Diagrams are also known as Data Flow Graphs. DFD's are commonly used during the problem analysis stage. They are useful in understanding a system and can be effectively used for partitioning during analysis.

The DFD network is a formal, logical abstract of a system that may have many possible physical configurations. This reason a set of symbols that do not imply a physical form are used to represent data source, data flows, data transformations and data storages.

The basic elements of DFD are:

- **Process**: A process that represents some amount of work being performed on data



Circle or Bubble

- **External Entity**: This represents any outside agency, which interacts with the system. It represents the source or destination of data for the system under consideration

Rectangle

- **Data Flow**: The flow portrays an interface among different components in a DFD. It represents flow of data between a process and an external entity or between a process and data store.

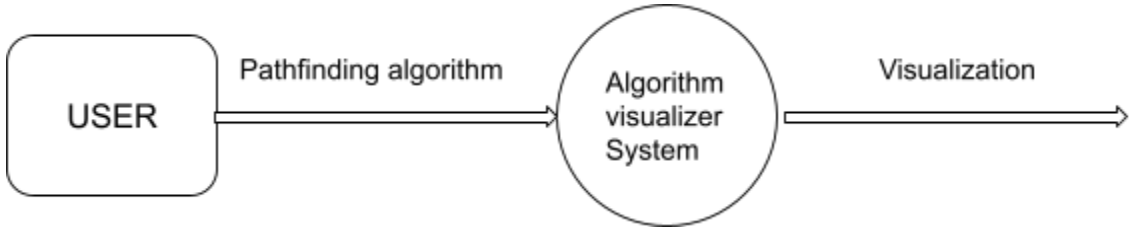
Arrow

- **Data Stores**: A data store is a place for holding information within the system.

One-end Opened rectangle

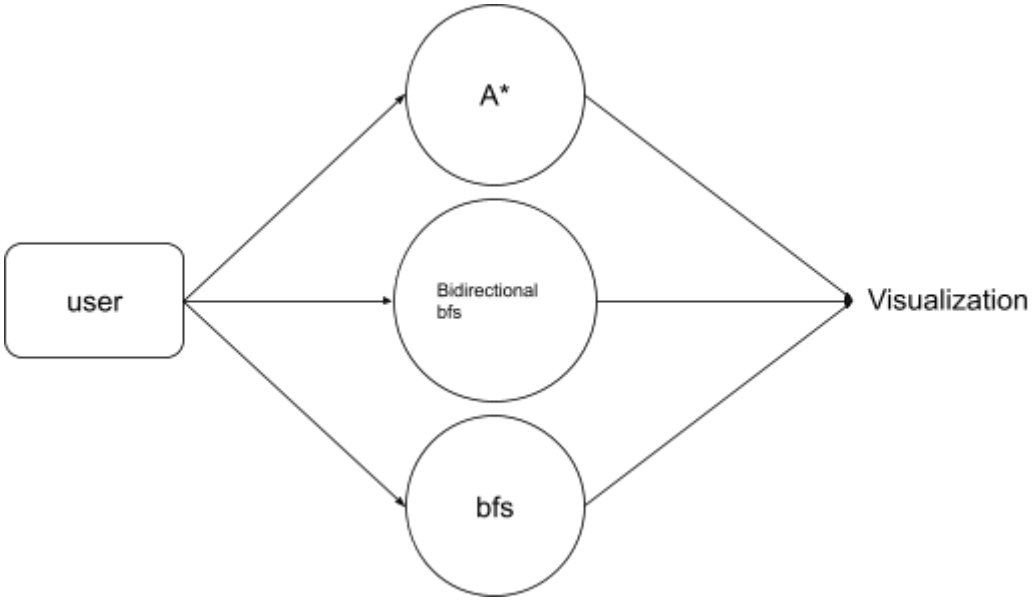
Data Flow Diagrams

- DFD LEVEL-0



In the Data Flow Diagram we show the flow of data in our system, in DFD 0 we show the base DFD in which a rectangle present External entity (an outside system that sends or receives data) and a circle shows a process (process that changes the data, producing an output). Arrows towards the process show input while the arrows away from the process shows output.

- DFD LEVEL 1
  DFD 1 is the further bifurcation of DFD 0



3.3 USE CASE DIAGRAM

A use case Diagram displays the relationship among actors and use cases. Use case diagrams are drawn to capture the functional requirements of a system. Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements.

- To identify functions and how roles interact with them.
- For a high-level view of the system.
- To identify internal and external actors.

The two main components of Use Case Diagrams are actors and cases.

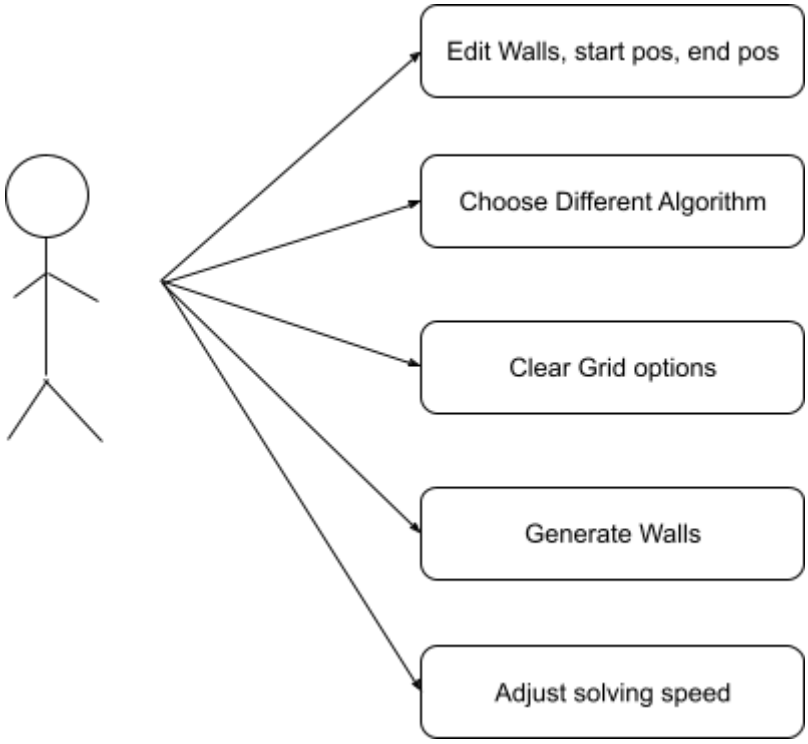
- Actor

  Actor in a use case diagram is any entity that performs a role in one given system. This could be a person, organization or an external system and usually drawn like skeleton shown below:

- Use case

  A use case represents a function or an action within the system. It is drawn as an oval and named with the functions. Symbol of use case is shown below:

User


Edit Walls, start pos, end pos
Choose Different Algorithm
Clear Grid options
Generate Walls
Adjust solving speed

4.SYSTEM REQUIREMENT

System requirements is a statement that identifies the functionality that is needed by a system in order to satisfy the user requirements. System requirements are the most effective way of meeting the user needs and reducing the cost of implementation. System requirements could cause a company to save a lot of money and time.They are the first and foremost important part of any project, because if the system requirements are not fulfilled, then the project is not complete.

4.1 Hardware Requirements

1.	Processor	:	Intel Pentium IV / AMD
2.	RAM	:	256 MB
3.	HardDisk	:	40 GB
4.	Drives	:	Optional
5.	Display Size	:	17 inch, color monitor
6.	Screen Resolution	:	800 x 600 pixels
7.	Color Palette	:	True Color (24bit)
8.	Keyboard	:	PC/AT enhanced type
9.	Mouse	:	Logitech PS/2 port mouse

4.2 Software Requirements

1. IDE	:	Visual Studio Code
2. Language	:	Html, Css, Javascript
3. Photo Editing Tool	:	Adobe Photoshop
4. Web Browsers	:	Google chrome, firefox, brave
5. Document Editor	:	Google doc.
6. Front End	:	Html, Css, Javascript.
7. Operating System	:	Windows 7 or higher.

4.3 Technology Used

Operating System

Windows XP was considered by many to be the PC industry's best hope to boost demand. Because many older PC's lacked the processing power and memory required for running XP, industry observers expected the product launch to increase sales of new PCs. Windows XP featured higher reliability than earlier versions of Windows, plus new features dealing with instant messaging, digital photography, and security (in the form of a "Personal Firewall" to block Internet intruders). While some feared that the ongoing Microsoft antitrust case might lead government regulators to try to block the introduction of XP or force it to b e altered, that did not happen, and the product was launched as expected.

HTML

The Hyper Text Markup Language or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages.

CSS (Cascading Style Sheet)

CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.
Separation of formatting and content also makes it possible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille- based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.
The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

Java Script

JavaScript is a scripting language. A scripting language is easy and fast to learn. A scripting language is interpreted at run-time. It is not compiled like other languages as C++, C#, VB.net etc. JavaScript is a client side language and it runs on a client browser. Netscape developed it and because of its simplicity it is one of the most known scripting languages. However JavaScript can also be used on the server side. JavaScript can be used on all most known browsers. It can be easily used to interact with HTML elements. You can validate text fields, disable buttons, validate forms, or change the background color of your page. All this is possible with JavaScript. Like each programming language, it contains variables, arrays, functions, operators, objects and much more which can help you to create better scripts for your pages. On the server side you can use JavaScript for example to manage database entry. JavaScript code can be inserted directly in the HTML or you can place it in a separate file with the .js extension and link the webpage with the .js file

Photoshop

Photoshop is photo editing software that edits photos, and images should be in a raster image format. It is available in many different languages, such as English, Chinese, Japanese, Dutch, Polish, and so on. Nowadays, Photoshop is also known as Adobe Photoshop.
Adobe Photoshop is a user-friendly, most powerful, and widely used image/graphics editing software developed by Adobe. Adobe Photoshop is basically a raster-based image editing software. With multiple layers and features such as masking, image wrapping tools, alpha compositing, fluid camera rotation, and file display tools, and much more advanced tools, Photoshop can edit and compose raster images.

5.SOFTWARE DESIGN

System Design develops the architectural details required to build a system or product. The System design process encompasses the following activities:

- Partition the analysis model into subsystems.
- Identify concurrency that is dictated by the problem.
- Develop design for the user interface.
- Choose a basic strategy or implement data management
- Design an appropriate control mechanism for the system, including task management.

System design provides an understanding of the procedural details necessary for implementing the system recommended in the feasibility study. Basically it is all about the creation of a new system. This is a critical phase since it decides the quality of the system has a major impact on the testing and implementation phases

- The allocation of the equipment and the software to be used
- The identification of the user requirement for the system.
- Drawing of the expanded system flow charts to identify all the processing functions required.

System design provides an understanding of the procedural details, necessary for implementing the system recommended in the feasibility study. Basically it is all about the creation of a new system. This is a critical phase since it decides the quality of the system and has a major impact on the testing and implementation phases.

System design is the most creative and challenging phase of the system life cycle. The term design describes the final system and the process by which it is to be developed. During the

system design phase the designers must design how to produce an efficient and effective system. There are two levels of system design, Logical design and Physical design.

In the logical design, the designer produces a specification of the major feature of the system which meets the objectives. The delivered product of logical design includes current requirements of the following following system components:

- Input design.
- Output design.
- Process design.

Physical design takes this logical design blueprint and produces the program specification. Physical design and user interfaces for a selected hardware and software. Structured design is data flow based methodology that partitions a program into a hierarchy of modules organized top-down manner with details at the bottom. The value of using a top-down approach, starting at the general levels to gain an understanding of the system and gradually moving down to levels of greater details, one data flow diagram becomes several at the next lower level. The top-down method is also widely used in systems engineering and software design. Each function the system will perform is first and then developed in greater detail.

Design Objectives and Principles

The two operational design objectives continually sought by the developers are system reliability and maintainability. A system is said to be reliable if it does not produce dangerous or costly failures when it is used in a reasonable manner, in a manner that a typical user expects is normal.

There are two levels of reliability. The first is that the system is meeting the right requirements. For instance, a system might be expected to have specific security features or controls built into it by users. The second level of system reliability involves the actual working of the system, reliability involves workings of the system delivered to the user. At this level, systems reliability is interwoven with software engineering and development.

An error occurs whenever the system does not produce the expected speed. No program is ever neither fully debugged nor fully tested, nor proven correct. The correctness of the design of a system depends upon the level of the precision of the system being built which satisfies the requirement of the system. The aim of the design phase is to produce the best possible design

within the limitations proposed by the requirements and the facilities. Some of the properties of the design are as follows.

It points out how easily the correctness of the design can be argued. All design elements must be traceable to the requirements. All the required components of the design must be specified. There should not be inherent inconsistencies. An efficient system is one that consumes less processing time and requires less memory and so it must be satisfied. Simplicity and understandability are the most important quality criteria for the software system. It would be a great help for the analyst in future while repairing the system.

4.3 Input Design

The user interface design is very important for any applicaiton. The Interface design describes how the software communicates within itself, to systems that interpret it and with humans who use it. The input design is the process of converting the user-oriented inputs into the computer-based format. The data is fed into the system using simple interactive forms. The forms.

Supplied with messages so that users can enter data without facing any difficulty. The data is validated wherever it is required in the project. This ensures that only the correct data have been incorporated into the system. The goal of designing input data is to make the automation as easy and free from errors as possible. For providing a good input design for the applicaiton easy data input and selection features are adopted. The input design requirements such as user friendliness, consistent format and Interactive Dialogue for giving the right message and help for the user at right time are also considered for the development of this project. The most common cause of errors in a system is invalid user input. Maximum care is taken to prevent invalid data from entering into the system. This was achieved by making proper validation checks on the user input Error messages are displayed when and where an incalid user entry/action is encountered

4.3 Output Design

Output refers to the results and information that are generated by the system Here determine information to be preset, decide layout and select output medium, arrange presentation of information in accepted format and decide how to distribute output to intended recipients. Location characteristics and format of column headings and pagination are specified.

Output design plays a major role in providing the user with required format. The major function of the output is to convey information and so its layout and design are careful considerations. Information must be carefully considered to the news of the user.

The design output form, attention is given to proper identification and working readability and use, composition and layout, order of data items and clarity of instructions. A well-designed form with clarity stated captions should be self-instructing. An organization's form must be centrally controlled for efficient handling

Computer output is the most important and direct information source to the user. Output design is a process that involves designing necessary outputs in the form of reports that should be given to the users according to the requirements. Efficient, intelligible output design should improve the systems relationship with the user and help in decision making. Since the reports are directly referred to by the management for taking decisions and to draw conclusions. They must be designed with almost care and the details in the reports must be simple, descriptive and clear to the user. So while designing output the following things are to be considered.

- It is the basis of management's final assessment of the system.
- It is designed by rapidly constructing prototypes
- During system design outputs are modeled data flows.
- Outputs may introduce new aspects to the system.

4.3 User Interface Design

Purpose of a USER INTERFACE DESIGN is to communicate effectively through form designs; There are several major requirements,

- User must able to move the Start node and End node inside Grid
- Users must be able to edit the wall nodes.
- Able to select different kinds of algorithms to solve.
- Clear Button to clear the grid.
- Able to generate Random Wall

6.CODE

Folder Structure Diagram

```
|--index.html
|--Readme.md
|--.git
|--CSS\
|      |--buttons.css
|      |--grid.css
|      |--menu.css
|      |--selection.css
|      |--style.css
|
|--JS\
|     |--algorithms.js
|     |--maze.js
|     |--menu.js
|     |--properties.js
|     |--script.js
|     |--visualizer.js
|
|--Resources\
      |--Shapes\
            |-start.svg
            |-target.svg
```

Index.html

```
<!DOCTYPE html>

<html>
  <head>
    <meta charset="utf-8" />
    <!--support utf8 character set-->
    <meta name="viewport" contetn="widht=device-width, initial-scale=0.0" />
    <!-- scale page in different screen sizes-->
```

```
  <meta http-equiv="X-UA-Compatible" content="ie-edge" />
  <!-- support older version of IE-->

  <title>AI PATHFINDING VISUALIZER</title>
  <!--  <link rel ="shortcut icon" type="image/png" href = "resource/images/icon.ping"/> -->

  <!--linking css files-->
  <link rel="stylesheet" type="text/css" href="css/style.css" />
  <link rel="stylesheet" type="text/css" href="css/menu.css" />
  <link rel="stylesheet" type="text/css" href="css/selection.css" />
  <link rel="stylesheet" type="text/css" href="css/buttons.css" />
  <link rel="stylesheet" type="text/css" href="css/grid.css" />

  <!--scripts-->
  <script type="text/javascript" src="js/properties.js"></script>
  <script type="text/javascript" src="js/visualizer.js" defer></script>
  <script type="text/javascript" src="js/maze.js" defer></script>
  <script type="text/javascript" src="js/algorithm.js" defer></script>
  <script type="text/javascript" src="js/menu.js" defer></script>
  <script type="text/javascript" src="js/script.js" defer></script>
  <!-- test script-->
  <!--  <script type="text/javascript" src="js/visualizer-master.js" defer></script> -->
</head>

<body>
  <div class="body-container">
    <div class="menu">
      <!-- pathfinding bar -->
      <div class="menu-title-wrap">
        <span class="menu-title">Pathfinding</span>
      </div>

      <!-- solving Algorithm header and it's dropdown menu-->
      <div class="choice" id="choice_1">
        <span class="choice_label" id="choice_label_1"
          >Solving Algorithm</span
        >
        <div class="select" for="slct_1">
          <!-- drop down-->
          <select id="slct_1" required="required">
            <option selected value="1">Breadth-First</option>
            <option value="2">A*</option>
```

```
      <option value="3">Bidirectional Breadth First</option>
    </select>
    <!-- drop down arrow-->
    <svg>
      <use xlink:href="#select-arrow-down"></use>
    </svg>
  </div>
</div>

<!-- solving Algorithm header and it's dropdown menu-->
<div class="slider-bar">
  <span class="choice_label" id="choice_label_1">Solving Speed</span>
  <div class="slider-align">
    <input
      class="slider"
      type="range"
      min="100"
      max="1000"
      value="50"
      id="slider_value"
    />
  </div>

  <div id="set_value"></div>
</div>

<!-- maze Algorithm header and it's dropdown menu-->
<div class="choice" id="choice_2">
  <span class="choice_label" id="choice_label_2">Maze Algorithm</span>
  <div class="select" for="slct_2">
    <!-- drop down -->
    <select id="slct_2" required="required">
      <option hidden selected value="0">Custom</option>
      <option value="1">Randomized Depth-first</option>
    </select>
    <!-- dropdown arrow -->
    <svg>
      <use xlink:href="#select-arrow-down"></use>
    </svg>
  </div>
</div>
```

```
      <!-- buttons---------->
      <div class="button-wrap">
        <!-- clear button-->
        <button class="my_button a" id="clear">Clear</button>
        <!-- start button-->
        <button class="my_button b" id="play">Start</button>
      </div>

      <!-- draw '\/' symbol for dorop down -->
      <svg class="sprites">
        <symbol id="select-arrow-down" viewbox="0 0 10 6">
          <polyline points="1 1 5 5 9 1"></polyline>
        </symbol>
      </svg>
    </div>

    <!-- visualizer view-->
    <div class="visualizer">
      <div id="grid"></div>
      <!-- generated grid-table will assign to grid div-->
    </div>
  </div>
 </body>
</html>
```

buttons.css

```
/* slider */
.slider {
  -webkit-appearance: none;
  width: 100%;
  height: 10px;
  background: #d3d3d3;
  outline: none;
  opacity: 0.7;
  -webkit-transition: 0.2s;
  transition: opacity 0.2s;
}

.slider:hover {
```

```
  opacity: 1;
}

.slider::-webkit-slider-thumb {
  -webkit-appearance: none;
  appearance: none;
  width: 16px;
  height: 16px;
  background: #04aa6d;
  cursor: pointer;
}

.slider-align {
  padding-top: 10px;
}
```

grid.css

```
#my_table
{
	border-spacing: 0;
	border-collapse: collapse;

	width: 100vh;
	height: 100vh;

}

.webkit-keyframes algo_in
{
	from { background-size: calc(0%) calc(0%); }
	to { background-size: calc(100%) calc(100%); }
}

.keyframes algo_in
{
	from { background-size: calc(0%) calc(0%); }
	to { background-size: calc(100%) calc(100%); }
}
```

```
.cell
{
	background-position: center;
	background-repeat: no-repeat;
	margin: 3px 3px;
}

.cell_1
{
	/*background-color: rgb(255, 255, 255); */
	background-color: rgb(172, 172, 172);

}

.cell_2
{
	/* background-color: rgb(231, 231, 233); */
	background-color: rgb(240, 240, 240);
}

.cell_wall
{
	background-color: rgb(63, 63, 63);
}

.cell_algo
{
	background-image: url(../resources/shapes/algo.svg);
	background-size: calc(100%) calc(100%);
	-webkit-animation: algo_in 0.3s;
	animation: algo_in 0.3s;
}

.cell_1.cell_path
{
	background-color: rgb(251, 244, 79);
}

.cell_2.cell_path
{
	background-color: rgb(245, 238, 73);
}
```

```
.cell_1.visited_cell
{
	background-color: rgb(255, 124, 92);
}

.cell_2.visited_cell
{
	background-color: rgb(255, 124, 92);
}

/* start and target style on grid */
.start
{
	background-image: url(../resources/shapes/start.svg);
	background-size: calc(100%) calc(100%);
}

.target
{
	background-image: url(../resources/shapes/target.svg);
	background-size: calc(100%) calc(100%);}
```

Menu.css

```
.menu-title-wrap
{
   width: 100%;
	background-color: #404d68;
	padding:20px 0px; /*top,down. left,right*/
	margin-bottom: 20px;

}

.menu-title
{
	color: rgb(255, 255, 255);
	font-size: 25px;
}

.choice {
	padding: 20px;
	text-align: left;
```

```
}

.choice_label{
	white-space:nowrap;
	color:#313c52;
	font-size:18px;
	left:0px;
	top:0px;
}

.sprites
{
	position: absolute;
	width: 0;
	height: 0;
	pointer-events: none;
	user-select: none;
	box-sizing: border-box;
}
```

selection.css

```
.select
{
	position: relative;
	margin-top: 5px;

}

.select svg
{
	position: absolute;
	right: 12px;
	top: calc(50% - 3px);
	width: 10px;
	height: 6px;
	stroke-width: 2px;
	stroke: #9098a9;
	fill: none;
	stroke-linecap: round;
	stroke-linejoin: round;
	pointer-events: none;
	box-sizing: border-box;
```

```
}

.select select
{
	-webkit-appearance: none;
	padding: 7px 40px 7px 12px;
	width: 100%;
	border: 1px solid #c9cfd8;
	border-radius: 5px;
	background: #fff;
	box-shadow: 0 1px 3px -2px #9098a9;
	cursor: pointer;
	font-family: inherit;
	font-size: 18px;
	transition: all 150ms ease;
	box-sizing: border-box;
	color: #404d68;
}

.select select option[value=""][disabled]
{
	display: none;
	box-sizing: border-box;
}

.select select:focus
{
	outline: none;
	border-color: #07f;
	box-shadow: 0 0 0 2px rgba(0,119,255,0.2);
	box-sizing: border-box;
}

.select select:hover + svg
{
	stroke: #07f;
	box-sizing: border-box;
}
```

style.css

```
*{
```

```
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

.body-container{
    min-height: 100vh;
    display: flex;
}

.menu{
    background-color: rgb(238, 238, 238);
    min-height: 100vh;
    width: 350px;
    z-index:10;
    text-align: center;
    box-shadow:8px 0px 30px rgba(6,13,29,0.418);
    font-family: 'Roboto', sans-serif;
}

.visualizer
{
    flex:1;
            z-index: 1;
}
```

algorithm.js

```
"use strict";

function distance(point_1, point_2) {
  return Math.sqrt(
    Math.pow(point_2[0] - point_1[0], 2) + Math.pow(point_2[1] - point_1[1], 2)
  );
}

// time interval between two consecutive path searches!
function maze_solvers_interval() {
  my_interval = window.setInterval(function () {
    if (!path) {
      place_to_cell(
        node_list[node_list_index][0],
        node_list[node_list_index][1]
```

```
      ).classList.add("cell_algo");
      node_list_index++;

      if (node_list_index == node_list.length) {
        if (!found) clearInterval(my_interval);
        else {
          path = true;
          place_to_cell(start_pos[0], start_pos[1]).classList.add("cell_path");
        }
      }
    } else {
      if (path_list_index == path_list.length) {
        place_to_cell(target_pos[0], target_pos[1]).classList.add("cell_path");
        clearInterval(my_interval);
        return;
      }

      place_to_cell(
        path_list[path_list_index][0],
        path_list[path_list_index][1]
      ).classList.remove("cell_algo");
      place_to_cell(
        path_list[path_list_index][0],
        path_list[path_list_index][1]
      ).classList.add("cell_path");
      path_list_index++;
    }
  }, maze_solver_algo_speed);
}

// breadth-first-algo
function breadth_first() {
  node_list = [];
  node_list_index = 0;
  path_list = [];
  path_list_index = 0;
  found = false;
  path = false;
  let frontier = [start_pos];
  grid[start_pos[0]][start_pos[1]] = 1;

  do {
```

```
  let list = get_neighbours(frontier[0], 1);
  frontier.splice(0, 1);

  for (let i = 0; i < list.length; i++)
    if (get_node(list[i][0], list[i][1]) == 0) {
      frontier.push(list[i]);
      grid[list[i][0]][list[i][1]] = i + 1;

      if (list[i][0] == target_pos[0] && list[i][1] == target_pos[1]) {
        found = true;
        break;
      }

      node_list.push(list[i]);
    }
} while (frontier.length > 0 && !found);

if (found) {
  let current_node = target_pos;

  while (current_node[0] != start_pos[0] || current_node[1] != start_pos[1]) {
    switch (grid[current_node[0]][current_node[1]]) {
      case 1:
        current_node = [current_node[0], current_node[1] + 1];
        break;
      case 2:
        current_node = [current_node[0] - 1, current_node[1]];
        break;
      case 3:
        current_node = [current_node[0], current_node[1] - 1];
        break;
      case 4:
        current_node = [current_node[0] + 1, current_node[1]];
        break;
      default:
        break;
    }

    path_list.push(current_node);
  }

  path_list.pop();
```

```
    path_list.reverse();
  }

  maze_solvers_interval();
}
// a-star
function a_star() {
  node_list = [];
  node_list_index = 0;
  path_list = [];
  path_list_index = 0;
  found = false;
  path = false;

  let frontier = [start_pos];
  let cost_grid = new Array(grid.length)
    .fill(0)
    .map(() => new Array(grid[0].length).fill(0));
  grid[start_pos[0]][start_pos[1]] = 1;

  do {
    frontier.sort(function (a, b) {
      let a_value =
        cost_grid[a[0]][a[1]] + distance(a, target_pos) * Math.sqrt(2);
      let b_value =
        cost_grid[b[0]][b[1]] + distance(b, target_pos) * Math.sqrt(2);
      return a_value - b_value;
    });

    let current_cell = frontier[0];
    let list = get_neighbours(current_cell, 1);
    frontier.splice(0, 1);

    for (let i = 0; i < list.length; i++)
      if (get_node(list[i][0], list[i][1]) == 0) {
        frontier.push(list[i]);
        grid[list[i][0]][list[i][1]] = i + 1;
        cost_grid[list[i][0]][list[i][1]] =
          cost_grid[current_cell[0]][current_cell[1]] + 1;

        if (list[i][0] == target_pos[0] && list[i][1] == target_pos[1]) {
          found = true;
```

```
        break;
      }

      node_list.push(list[i]);
    }
  } while (frontier.length > 0 && !found);

  if (found) {
    let current_node = target_pos;

    while (current_node[0] != start_pos[0] || current_node[1] != start_pos[1]) {
      switch (grid[current_node[0]][current_node[1]]) {
        case 1:
          current_node = [current_node[0], current_node[1] + 1];
          break;
        case 2:
          current_node = [current_node[0] - 1, current_node[1]];
          break;
        case 3:
          current_node = [current_node[0], current_node[1] - 1];
          break;
        case 4:
          current_node = [current_node[0] + 1, current_node[1]];
          break;
        default:
          break;
      }

      path_list.push(current_node);
    }

    path_list.pop();
    path_list.reverse();
  }

  maze_solvers_interval();
}
// bi-directional-breadth-first
function bidirectional_breadth_first() {
  node_list = [];
  node_list_index = 0;
  path_list = [];
```

```
path_list_index = 0;
found = false;
path = false;
let current_cell;
let start_end;
let target_end;
let frontier = [start_pos, target_pos];
grid[target_pos[0]][target_pos[1]] = 1;
grid[start_pos[0]][start_pos[1]] = 11;

do {
  current_cell = frontier[0];
  let list = get_neighbours(current_cell, 1);
  frontier.splice(0, 1);

  for (let i = 0; i < list.length; i++) {
    if (get_node(list[i][0], list[i][1]) == 0) {
      frontier.push(list[i]);

      if (grid[current_cell[0]][current_cell[1]] < 10)
        grid[list[i][0]][list[i][1]] = i + 1;
      else grid[list[i][0]][list[i][1]] = 11 + i;

      node_list.push(list[i]);
    } else if (get_node(list[i][0], list[i][1]) > 0) {
      if (
        grid[current_cell[0]][current_cell[1]] < 10 &&
        get_node(list[i][0], list[i][1]) > 10
      ) {
        start_end = current_cell;
        target_end = list[i];
        found = true;
        break;
      } else if (
        grid[current_cell[0]][current_cell[1]] > 10 &&
        get_node(list[i][0], list[i][1]) < 10
      ) {
        start_end = list[i];
        target_end = current_cell;
        found = true;
        break;
      }
```

```
    }
  }
} while (frontier.length > 0 && !found);

if (found) {
  let targets = [target_pos, start_pos];
  let starts = [start_end, target_end];

  for (let i = 0; i < starts.length; i++) {
    let current_node = starts[i];

    while (
      current_node[0] != targets[i][0] ||
      current_node[1] != targets[i][1]
    ) {
      path_list.push(current_node);

      switch (grid[current_node[0]][current_node[1]] - i * 10) {
        case 1:
          current_node = [current_node[0], current_node[1] + 1];
          break;
        case 2:
          current_node = [current_node[0] - 1, current_node[1]];
          break;
        case 3:
          current_node = [current_node[0], current_node[1] - 1];
          break;
        case 4:
          current_node = [current_node[0] + 1, current_node[1]];
          break;
        default:
          break;
      }
    }

    if (i == 0) path_list.reverse();
  }

  path_list.reverse();
}

maze_solvers_interval();
```

```
}

function maze_solver() {
  clear_grid();
  grid_clean = false;

  if (
    (Math.abs(start_pos[0] - target_pos[0]) == 0 &&
      Math.abs(start_pos[1] - target_pos[1]) == 1) ||
    (Math.abs(start_pos[0] - target_pos[0]) == 1 &&
      Math.abs(start_pos[1] - target_pos[1]) == 0)
  ) {
    place_to_cell(start_pos[0], start_pos[1]).classList.add("cell_path");
    place_to_cell(target_pos[0], target_pos[1]).classList.add("cell_path");
  } else if (document.querySelector("#slct_1").value == "1") {
    console.log(":: first button pressed ::");
    breadth_first();
  } else if (document.querySelector("#slct_1").value == "2") {
    console.log("::----------fuck button pressed----------::");
    a_star();
  } else if (document.querySelector("#slct_1").value == "3") {
    bidirectional_breadth_first();
  }
}
```

maze.js

```
"use strict";
console.log("::: Maze Loaded::: ");

function get_neighbours(cell, distance) {
  let up = [cell[0], cell[1] - distance];
  let right = [cell[0] + distance, cell[1]];
  let down = [cell[0], cell[1] + distance];
  let left = [cell[0] - distance, cell[1]];
  return [up, right, down, left];
}

function random_int(min, max) {
  min = Math.ceil(min);
  max = Math.floor(max);
  return Math.floor(Math.random() * (max - min)) + min;
}
```

```
function fill() {
  for (let i = 0; i < grid.length; i++)
    for (let j = 0; j < grid[0].length; j++) add_wall(i, j);
}

function fill_walls() {
  for (let i = 0; i < grid.length; i++)
    for (let j = 0; j < grid[0].length; j++)
      if (i % 2 == 0 || j % 2 == 0) add_wall(i, j);
}

function enclose() {
  for (let i = 0; i < grid.length; i++) {
    add_wall(i, 0);
    add_wall(i, grid[0].length - 1);
  }

  for (let j = 0; j < grid[0].length; j++) {
    add_wall(0, j);
    add_wall(grid.length - 1, j);
  }
}

function randomized_depth_first() {
  fill(); // fill wall fully;

  let current_cell = [1, 1];
  remove_wall(current_cell[0], current_cell[1]);
  grid[current_cell[0]][current_cell[1]] = 1;
  let stack = [current_cell];

  my_interval = window.setInterval(function () {
    if (stack.length == 0) {
      clearInterval(my_interval);
      clear_grid();
      generating = false;
      return;
    }

    current_cell = stack.pop();
    let neighbours = [];
```

```
    let list = get_neighbours(current_cell, 2);

    for (let i = 0; i < list.length; i++)
      if (
        get_node(list[i][0], list[i][1]) == -1 ||
        get_node(list[i][0], list[i][1]) == 0
      )
        neighbours.push(list[i]);

    if (neighbours.length > 0) {
      stack.push(current_cell);
      let chosen_cell = neighbours[random_int(0, neighbours.length)];
      remove_wall(
        (current_cell[0] + chosen_cell[0]) / 2,
        (current_cell[1] + chosen_cell[1]) / 2
      );
      remove_wall(chosen_cell[0], chosen_cell[1]);
      grid[chosen_cell[0]][chosen_cell[1]] = 1;
      stack.push(chosen_cell);
    } else {
      remove_wall(current_cell[0], current_cell[1]);
      grid[current_cell[0]][current_cell[1]] = 2;
      place_to_cell(current_cell[0], current_cell[1]).classList.add(
        "visited_cell"
      );

      for (let i = 0; i < list.length; i++) {
        let wall = [
          (current_cell[0] + list[i][0]) / 2,
          (current_cell[1] + list[i][1]) / 2,
        ];

        if (
          get_node(list[i][0], list[i][1]) == 2 &&
          get_node(wall[0], wall[1]) > -1
        )
          place_to_cell(wall[0], wall[1]).classList.add("visited_cell");
      }
    }
  }, maze_solver_speed);
}
```

```
function test_gen() {
  console.log("calling test_gen() go generate maze");
}

function maze_generator() {
  let start_temp = start_pos;
  let target_temp = target_pos;
  // hidden_clear();
  generating = true;

  if (start_temp[0] % 2 == 0) {
    if (start_temp[0] == grid.length - 1) start_temp[0] -= 1;
    else start_temp[0] += 1;
  }

  if (start_temp[1] % 2 == 0) {
    if (start_temp[1] == 0) start_temp[1] += 1;
    else start_temp[1] -= 1;
  }

  if (target_temp[0] % 2 == 0) {
    if (target_temp[0] == grid.length - 1) target_temp[0] -= 1;
    else target_temp[0] += 1;
  }

  if (target_temp[1] % 2 == 0) {
    if (target_temp[1] == 0) target_temp[1] += 1;
    else target_temp[1] -= 1;
  }

  grid_clean = false;

  if (document.querySelector("#slct_2").value == "1") {
    console.log("calling reandomized_depth_first");
    randomized_depth_first();
  } else if (document.querySelector("#slct_2").value == "2") {
    console.log("calling test_gen()");
    test_gen();
  }
}
```

menu.js

```
"use strict"
console.log("::: menu Loaded::: ");

function hidden_clear() {
    for (let i = 0; i < timeouts.length; i++ ) {
        clearTimeout(timeouts[i]);
    }

    // timeouts[];
    clearInterval(my_interval);

    // delete pre-grid;
    delete_grid();
    // generate new grid;
    generate_grid();
    // handle mouse inputs;
    visualizer_event_listeners();
}

function clear(){
    console.log('clear function called');
    document.querySelector("#slct_2").value = "0";

    // whiche clear pre-grid, create new grid, handle mouse input
    hidden_clear();
}


function menu_event_listeners()
{
    document.querySelector("#slct_2").addEventListener('change',event=>{
        console.log('.....clearing..... previous algo');
        hidden_clear();
        maze_generator();
    })

    document.querySelector("#clear").addEventListener('click', event=>{
        console.log(':::::button pressed::::');

        let start_temp = start_pos;
                    let target_temp = target_pos;
```

```
        // directly calling hidden_clear() function;
                clear();

                place_to_cell(start_pos[0], start_pos[1]).classList.remove("start");
                place_to_cell(start_temp[0], start_temp[1]).classList.add("start");
                place_to_cell(target_pos[0], target_pos[1]).classList.remove("target");
                place_to_cell(target_temp[0], target_temp[1]).classList.add("target");

                start_pos = start_temp;
                target_pos = target_temp;
    });


    document.querySelector("#play").addEventListener('click', event=>{
        console.log('pressed play button');

        if(generating) {
            document.querySelector("#slct_2").value = "0";
        }

        generating = false;
        clear_grid();
        maze_solver();

    });
}
```

properties.js

```
"use strict";

console.log("::: properties Loaded::: ");

// properties.js
const initial_max_grid_size = 47;
const menu_width = 350;

// speed of the maze_generation
const maze_solver_speed = 100; // value > speed
// speed of the maze_ solver {affect all the algo}
const maze_solver_algo_speed = 100; // value > speed
```

```
let cell_size;
let grid_size_x;
let grid_size_y;
let grid;

let start_pos;
let target_pos;

let grid_clean = true;
let my_interval;

let moving_start = false;
let moving_target = false;

let clicking = false;

let generating = false;
let timeouts = [];

// algorithm variables;
let node_list;
let node_list_index;

let path_list;
let path_list_index;

let found = false;
let path = false;
```

script.js

```
"use strict";
console.log("::: script Loaded::: ");

slider = document.getElementById("slider_value");
output = document.getElementById("set_value");

window.onload = function () {
  // functions calls.
  generate_grid();
  visualizer_event_listeners();
```

```
  console.log("vlue output " + output);
  output.innerHTML = slider.value + " :M/s";

  // should not call here.
  //  randomized_depth_first();
  menu_event_listeners();
};

slider.oninput = function () {
  output.innerHTML = slider.value + " :M/s";
  maze_solver_algo_speed = slider.value;};
```

visualizer.js

```
"use strict";

console.log("::: visualizer Loaded::: ");
// 0 = > clear path
// -1 = > wall
//  start_pos[][]- > start
//  target_pos[][] -> target
function generate_grid() {
  // x,y value should be the same.
  grid_size_x = 10;
  grid_size_y = 10;

  // create new table and assign "my_table" id to table;
  let table = document.createElement("table");
  table.id = "my_table";

  for (let i = 0; i < grid_size_y; i++) {
    // create a new table row
    let row = document.createElement("tr");
    for (let j = 0; j < grid_size_x; j++) {
      // create a new table data;
      let cell = document.createElement("td");
      let class_name = "";

      if ((i + j) % 2 == 0) {
        class_name = "cell cell_1"; // dark grey
      } else {
        class_name = "cell cell_2"; // light grey
      }
```

```
    class_name += " x_" + j.toString(10) + " y_" + i.toString(10);
    cell.className = class_name;
    // adding data to row;

    // cell.innerHTML=" ";
    row.append(cell);
  }
  // adding each row to the table;
  table.appendChild(row);
}

document.querySelector("#grid").appendChild(table); // append grid table to "id="#grid" div

grid = new Array(grid_size_x)
  .fill(0)
  .map(() => new Array(grid_size_y).fill(0)); // assign array to grid.
/*
     [1] [2]
     [3] [4]
     [5] [6]
  */

// generate start-pos coordinate { from x,y grid size value};
start_pos = [Math.floor(grid_size_x / 4), Math.floor(grid_size_y / 2)];

// generate target-pos coordinate { from x,y grid size value};
target_pos = [Math.floor((3 * grid_size_x) / 4), Math.floor(grid_size_y / 2)];

/*
  |Debug| to print: startpos, target pos value;
  console.log("startpos: "+ start_pos);
  console.log("target pos: "+target_pos);
  */

// adjust start pos, targe pos, in odd.
if (start_pos[0] % 2 == 0) {
  start_pos[0] += 1;
}
if (start_pos[1] % 2 == 0) {
  start_pos[1] -= 1;
}
```

```
  if (target_pos[0] % 2 == 0) {
    target_pos[0] += 1;
  }
  if (target_pos[1] % 2 == 0) {
    target_pos[1] -= 1;
  }

  // place the :start point:
  place_to_cell(start_pos[0], start_pos[1]).classList.add("start"); // < add  id="start" to  table
td[*]>

  // place the :target point:
  place_to_cell(target_pos[0], target_pos[1]).classList.add("target"); // <add id="target to table
td[*]">
}

// to place the element;
function place_to_cell(x, y) {
  // return the (.x_1 .y_2);
  return document.querySelector(
    ".x_" + x.toString(10) + ".y_" + y.toString(10)
  );
}

// to delete entire grid(table);
function delete_grid() {
  document.querySelector("#my_table").remove();
}

// return the cell, x,y coordinates.
function cell_to_place(cell) {
  let text_x = cell.classList[2];
  let text_y = cell.classList[3];

  console.log("cell = " + cell);

  text_x = text_x.split("x_")[1];
  text_y = text_y.split("y_")[1];

  return [parseInt(text_x, 10), parseInt(text_y, 10)];
}
```

```
// adding wall;
function add_wall(x, y) {
  let cell = place_to_cell(x, y);
  // cell = > return documet.querySelection();
  // get id like "start" "target" values;

  if (!cell.classList.contains("start") && !cell.classList.contains("target")) {
    grid[x][y] = -1;

    // change grid coordinate to -1, walls;
    // change the td id " cell_wall";
    cell.classList.add("cell_wall");
  }
}

// removing wall;
function remove_wall(x, y) {
  // set back -1 to 0 for grid { pathfinding algo to work with};
  grid[x][y] = 0;
  // remove td id" cell_wall";
  place_to_cell(x, y).classList.remove("cell_wall");
}

function clear_grid() {
  if (!grid_clean) {
    // default value is 'true', if grid is generation value must be 'false';
    for (let i = 0; i < timeouts.length; i++) clearTimeout(timeouts[i]);

    timeouts = [];
    clearInterval(my_interval);

    for (let i = 0; i < grid.length; i++) {
      for (let j = 0; j < grid[0].length; j++) {
        if (grid[i][j] > -1) {
          remove_wall(i, j);
          // if the algorithm runs, the id containing "cell_algo" "cell_path" must be removed. so gird
must be clear
          place_to_cell(i, j).classList.remove("cell_algo");
          place_to_cell(i, j).classList.remove("cell_path");
        } else if (grid[i][j] < -1) {
          add_wall(i, j); /// if the [i][j] 0.
        }
```

```
      // remove visisted_cell
      place_to_cell(i, j).classList.remove("visited_cell");
     }
    }
    grid_clean = true; // other functions to should know if the grid is clear or not,
  }
}

function get_node(x, y) {
  if (x >= 0 && x < grid.length && y >= 0 && y < grid[0].length) {
    return grid[x][y];
  }
  return -2;
}

function click_event(event) {
  // even should be passed;
  event.preventDefault();

  if (clicking && event.target.classList.contains("cell")) {
    clear_grid(); // remove all the grid. and data. to place new target node.
    let place = cell_to_place(event.target); // return coordinate value of the node. (1,0), we were
clicking on table.
    console.log("target node: " + place);

    // default value of moving_start: false, and also the node should not be a same node. as
"target".

    if (moving_start && !event.target.classList.contains("target")) {
      start_pos = place; // change start_pos value to the current button click coordinate.

      document.querySelector(".start").classList.remove("start");
      event.target.classList.add("start"); // assign "start" id to target. node. where mouse placed;

      if (grid[place[0]][place[1]] < 0) {
        document.querySelector("#slct_2").value = "0"; // maze algorithm change to custom. =
>[0];
        remove_wall(place[0], place[1]); // remove the wall and place the start node[x] [y].
      }

      if (generating) {
```

```
        // if maze is generating, then value of maze selection list => 0
        document.querySelector("#slct_2").value = "0";
      }
      // switch back to false;
      generating = false;
    } else if (moving_target && !event.target.classList.contains("start")) {
      target_pos = place;
      document.querySelector(".target").classList.remove("target");
      event.target.classList.add("target");

      if (grid[place[0]][place[1]] < 0) {
        document.querySelector("#slct_2").value = "0";
        remove_wall(place[0], place[1]);
      }

      if (generating) {
        document.querySelector("#slct_2").value = "0";
      }

      generating = false;
    } else {
      document.querySelector("#slct_2").value = "0";

      if (grid[place[0]][place[1]] == 0) {
        add_wall(place[0], place[1]);
      } else {
        remove_wall(place[0], place[1]);
      }
    }
  }
}

function visualizer_event_listeners() {
  document.querySelector("#my_table").addEventListener("mousedown", (event) => {
    event.preventDefault();
    clicking = true;

    if (event.target.classList.contains("start")) moving_start = true;

    if (event.target.classList.contains("target")) moving_target = true;

    click_event(event);
```

```
});

document.querySelector("#my_table").addEventListener("mouseup", (event) => {
  event.preventDefault();
  clicking = false;
  moving_start = false;
  moving_target = false;
});

document.querySelector("#my_table").addEventListener("mouseover", (event) => {
  click_event(event);
});

document
  .querySelector("#my_table")
  .addEventListener("mouseleave", (event) => {
    event.preventDefault();
    clicking = false;
    moving_start = false;
    moving_target = false; }); }
```

7.SYSTEM TESTING

Software testing is the process used to help identify the correctness, completeness, security and quality of developed computer software. Testing is vital to the success of the system. System Testing makes a logical assumption that if all the parts of the system are correct, the goal will be successfully achieved.

System Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. The user tests the developed system and changes are made according to their needs. The Testing phase involves the testing of a developed system using various kinds of data.

There are many approaches to software testing, to software testing, but effective testing of complex products is essentially a process of investigation, not merely a matter of creating a following rote procedure. One definition of testing is “the process of questioning a product in order to evaluate it”, where the “questions” are things the tester tries to do with the product, and the product answers with its behavior in reaction to the probing of a tester. The quality of the applicaiton can, and normally does, vary widely from system to system but some of the common quality attributes include reliability, stability, portability, maintainability and usability.

The Software and Hardware are integrated and a full range of system tests is conducted in an attempt to uncover errors at the Software/Hardware Interface. System testing is a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all the work should verify that all system elements have been properly integrated and performed allocated functions. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently before live operation comments. Testing is vital to the success of the system. If the system is correct, the goal will be successfully achieved. The entire software has been tested with available data and it has been found that the expected output forms are generated by the system. It is now ready for implementation.

Unit Testing

Unit testing focuses verification effort on the smallest unit of software design that is the module. The Unit Testing is always White Box oriented and the step can be conducted in parallel for modules. Boundary conditions are tested to ensure that the module operates properly. In white box testing the test developer has access to the source code and can write code that links into the libraries, which are linked into the target software. This is typical of unit tests, which only test parts of a software system. They ensure that components used in the construction are functional and robust to some degree. Unit testing is an equivalent to the coding step. After the source code level code has been developed, reviewed and verified for correct syntax, unit test case design begins since a module is not a stand-alone program. In most applications a driver program is nothing more than a main program that accepts test case data. Passes such data to the module to be tested, and prints the relevant result

1) The test cases guarantee that all independent paths within a Module have been exercised at least once.
2) The test case exercises all logical decisions on their true and false sides.
3) The test case execute all loops at their boundaries and within operational bounds
4) The test cases exercise internal data structures to ensure their validity.

Integration Testing

Integration testing (sometimes called integration and testing and abbreviated I&T) is the phase of software testing in which individual software modules are combined and tested as a group. It follows unit testing and precedes system testing. Integration testing is a systematic technique for constructing, while at the same time conducting test to uncover errors associated with interfacing. The purpose of Integration testing is to verify functional, performance and reliability requirements placed on major design items. All test cases are constructed to test that all components within assemblages interact correctly The objective is to take unit testing modules and build a program structure that has been dictated by design. The steps involved in this testing include:

- The main control module is used as a test driver and status is substituted for all modules directly subordinate to the main control module.
- Depending on the integration approach selected, that is depth or breadth first, subordinate stubs are replaced one at time eight action modules.
- Test is conducted as each module is integrated.

- On the completion of each area of tests, another stub is placed with the real module.
- The low-level modules are combined into clusters that perform a specific software sub function.
- A drive that is the control program for testing is written to coordinate test case input and output
- The cluster is tested.

Functionality Testing

A black box testing geared to functional requirements of the system.In this functionality each module is tested. Test focused on verifying the target of test functions as intended, providing the required services, methods and use cases.

Validation Testing

This provides the final assurance that software meets all the functional, behavioral and performance requirements. The software is completely assembled on a package. Validation succeeds when the software functions in a manner in which the user expects. Validation refers to the process of using software in a live environment in order to find errors.

During the course of validating the system, failures may occur and sometimes the coding has to change according to the requirements. Thus the feedback from the validation phase generally precedes changes in the software. Once the application was made free of all logical and interface errors, inputting dummy data ensured that the software developed satisfies all the requirements of the user. This dummy data is usually known as the test case.

Output Testing

After performing the validation testing the next step is the output testing of the proposed system since no system should be useful if it does not provide the required output in the specific. Asking the users about the format required by them test the outputs generated or displayed by the system under consideration. Here, the output format on the screen is formed to be correct as the format was designed in the system phase according to the user's needs. For the hard copy also, the output comes out as the specified requirements by the user in any correction in the system.

Output Testing

After performing the validation testing the next step is the output testing of the proposed system since no system should be useful if it does not provide the required output in the specific. Asking

the user about the format required by them tests the outputs generated or displayed by the system under consideration. Here, the output format is considered in to ways, one on screen and another in the oriented format. The output format on the screen is formed to be correct as the format was designed in the system phase according to the user's needs. For the hard also, the output comes out as the specified requirements by the user in any correction in the system.

User Acceptance Testing

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system. Users can make changes whenever required. This is done regarding the following points.

- Input screen Design
- Output screen Design

The above testing is done taking various kinds of test data Preparation and test data plays a vital role in the system testing. After preparing the test data, the system under study is teste using that test data. While testing the system by which test data errors are agan uncovered and corrected by using testing steps and corrections are also noted for future use.

8.SYSTEM IMPLEMENTATION

Implementation is the process of converting a new or revised system design into operation. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system. It must therefore be carefully planned and controlled. Apart from planning the two major task of preparing for implementation are education and training of users and testing of the system.Education of users hould really take place much earlier in the project ie. when they are involved in the investigation and design work. Training has been given to the staff regarding the new system. Once staff has been trained, the system can be tested. Implementation in the stage in achieving a successful new system and giving the user confidence that the new system will work and be effective.

Implementation is the final and important phase. It is the phase where theoretical design is turned into a working system, which works for the user in the most effective manner. It involves careful playing investigation of the present system and the constraints involved, user training, system testing and successful running of the developed proposed system.The Implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out regarding the equipment and resources and the additional equipment have been acquired to implement the new system. The user test the developed system and the changes are made according to the needs. The testing phase involves the testing of a system using various kinds of data. The method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain types of transaction while using the new system. An elaborate testing of data is prepared and the system is tested using the data. While testing errors are noted and corrections are made. The users are trained to operate the developed system. Both hardware and software are made to run the developed system successfully in nature. The method of implementation and the scale to be adopted are found out initially. Next the system is tested properly and the users are trained in the new procedures.

9.MAINTENANCE

Maintenance is a characteristic of design and implementation, which is expressed as the probability that an item will be retained in or restored to a specific condition within a given period of time, when maintenance is performed in accordance with the prescribed procedures and resources.

Maintenance is the enigma of system development. It holds the software industry captive, tying up programming resources. Analysts and programmers spend far more time maintaining programs than they do writing them.

Maintenance can be classified as corrective, adaptive or prefecture. Corrective maintenance means repairing processing or performance failures or making changes because of previously uncorrected problems or false assumptions. Adaptive maintenance means repairing processing or performance or modifying the program to respond to the users additional or changing needs. Of this type more time and money are spent on prefecture than on corrective and adaptive maintenance

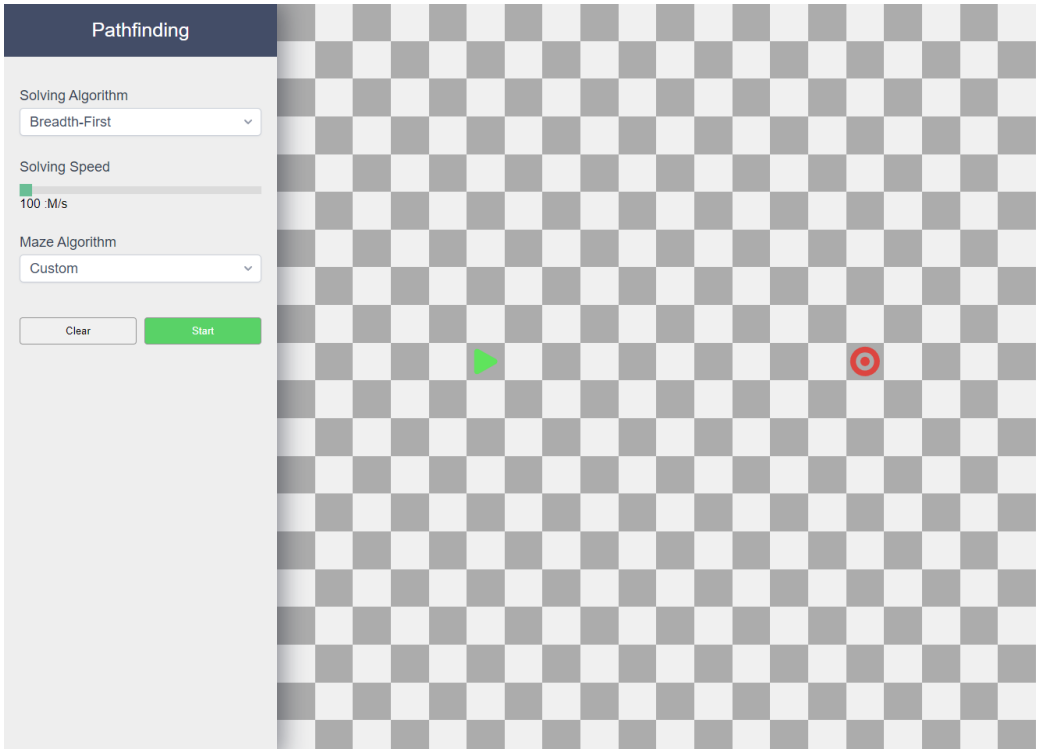
Technical and management approaches to the maintenance phase can be implemented with little upheaval. However tasks performed during the software engineering process define maintainability and have an importance on the success of any maintenance approach.

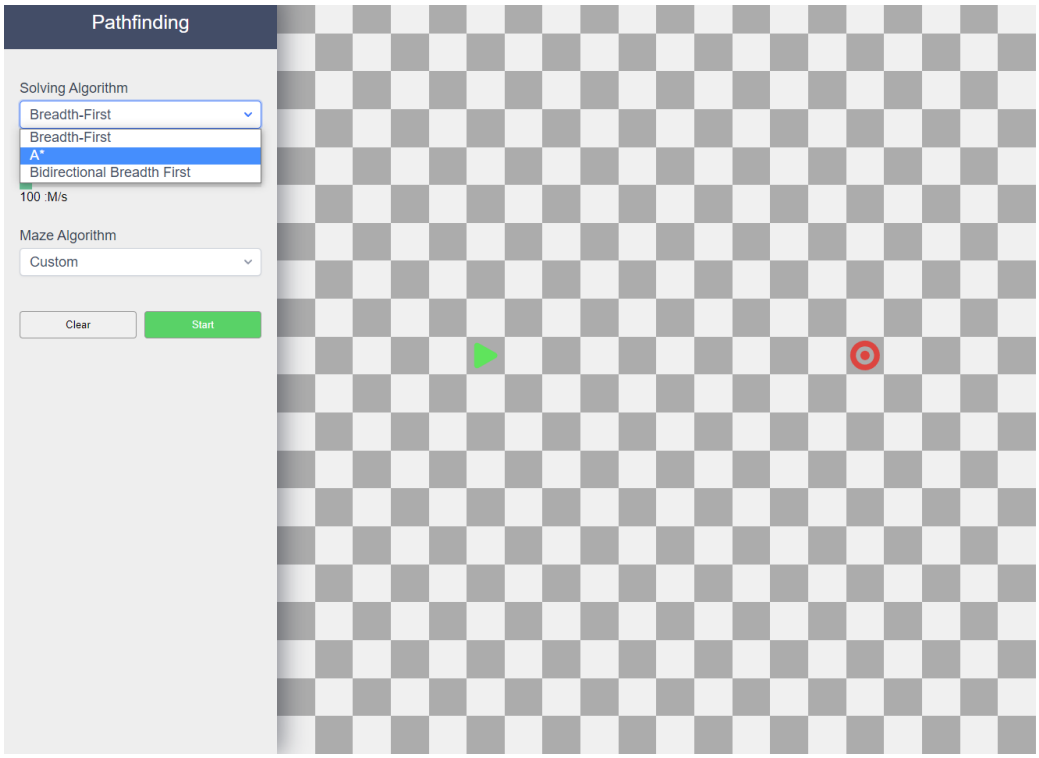
10.CONCLUSION

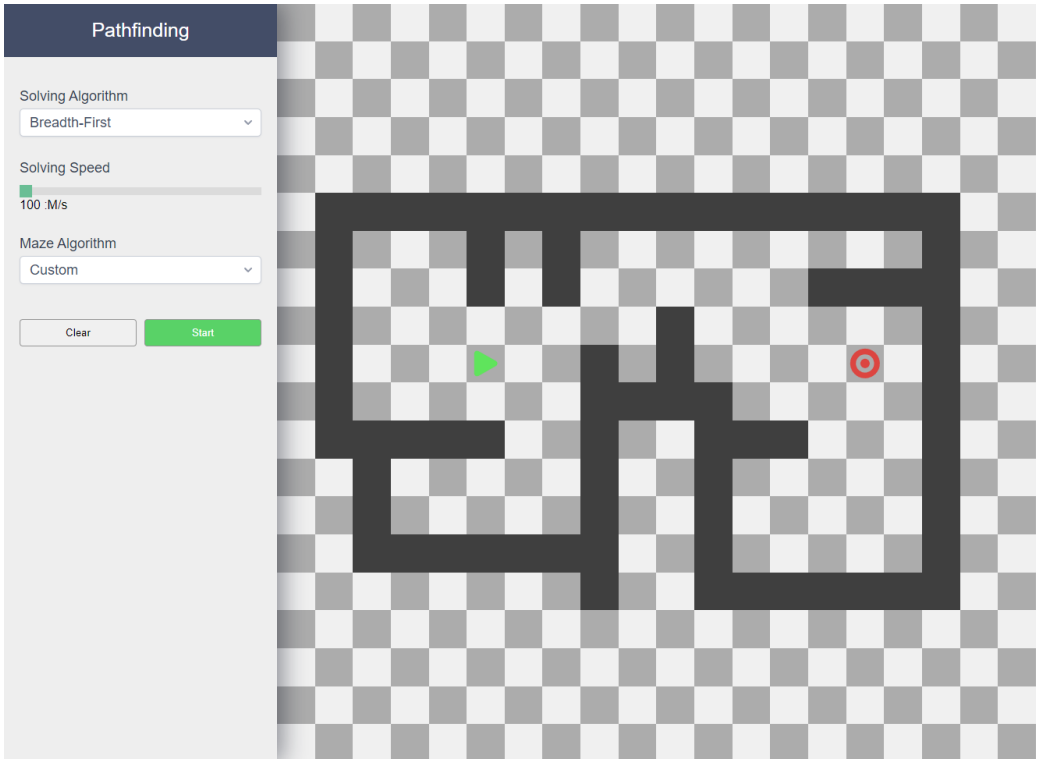
We started our project by studying a number of the well-known algorithm visualizations that are developed over a few years. According to our findings, algorithmic visualization is often seen as a valuable supporting tool. The main goal of the project is to use it from research educators and students for teaching and studying the existing known graph algorithms. The main goal of the project is to use it from research educators and students for teaching and studying the existing known graph algorithms. The main plan of the system is to provide an associated educational environment for both instructors and students to facilitate the learning process in an economical way.

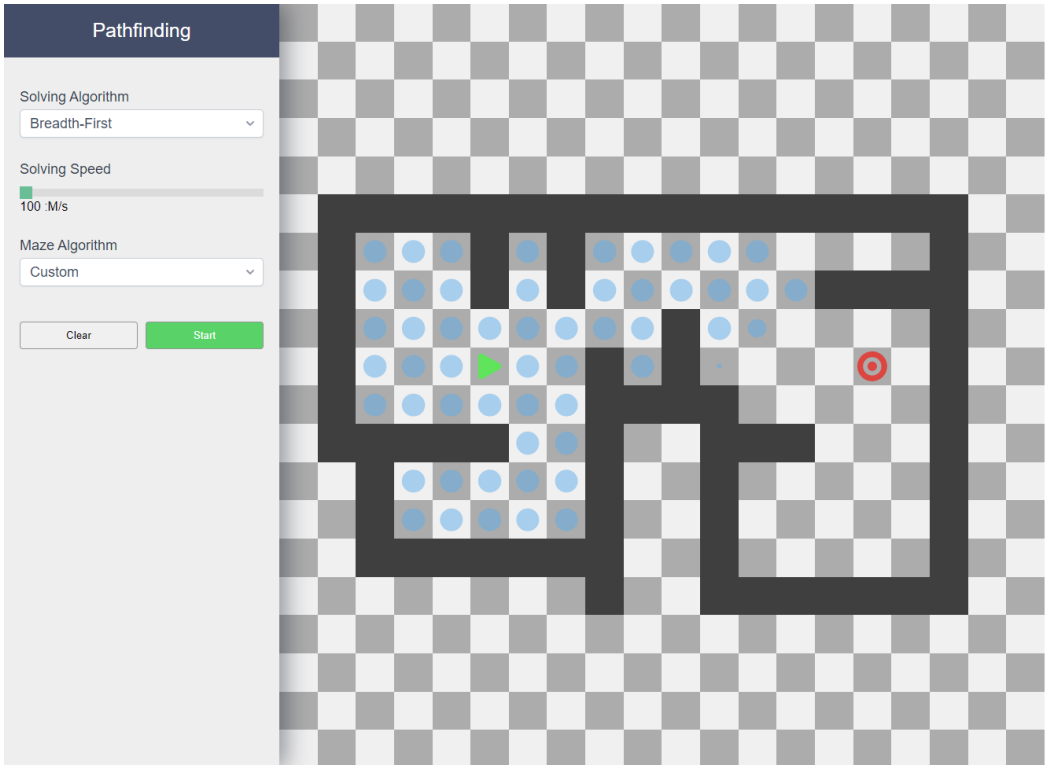
11.APPENDIX

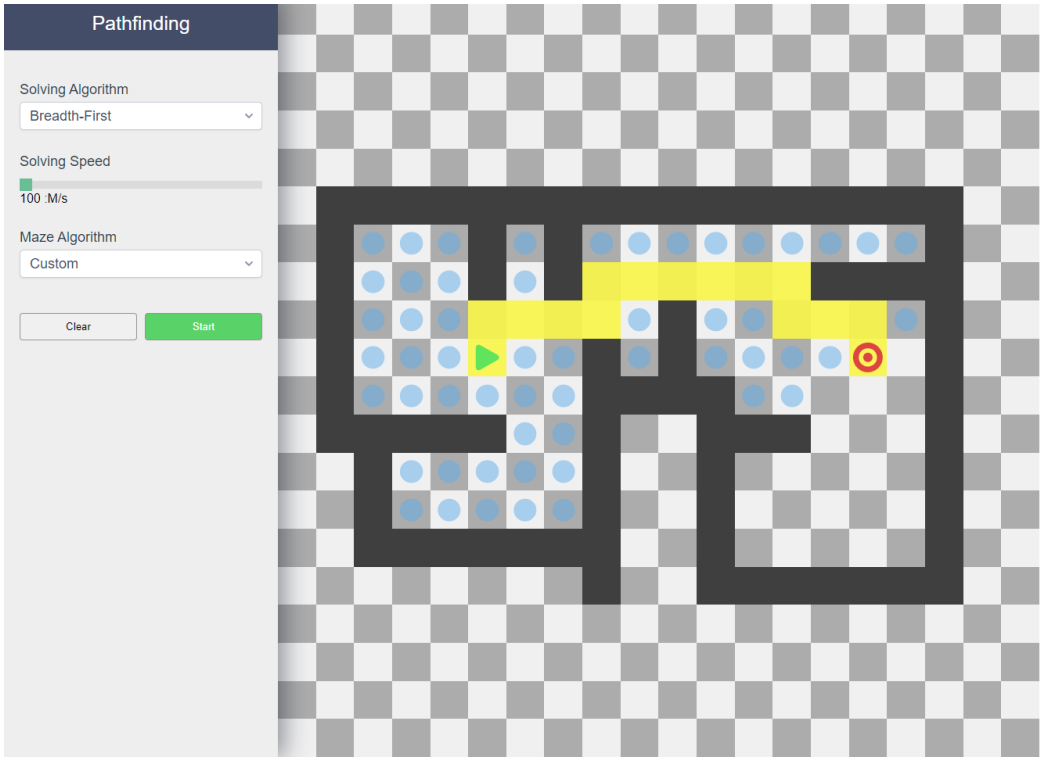
11.1 Screenshots

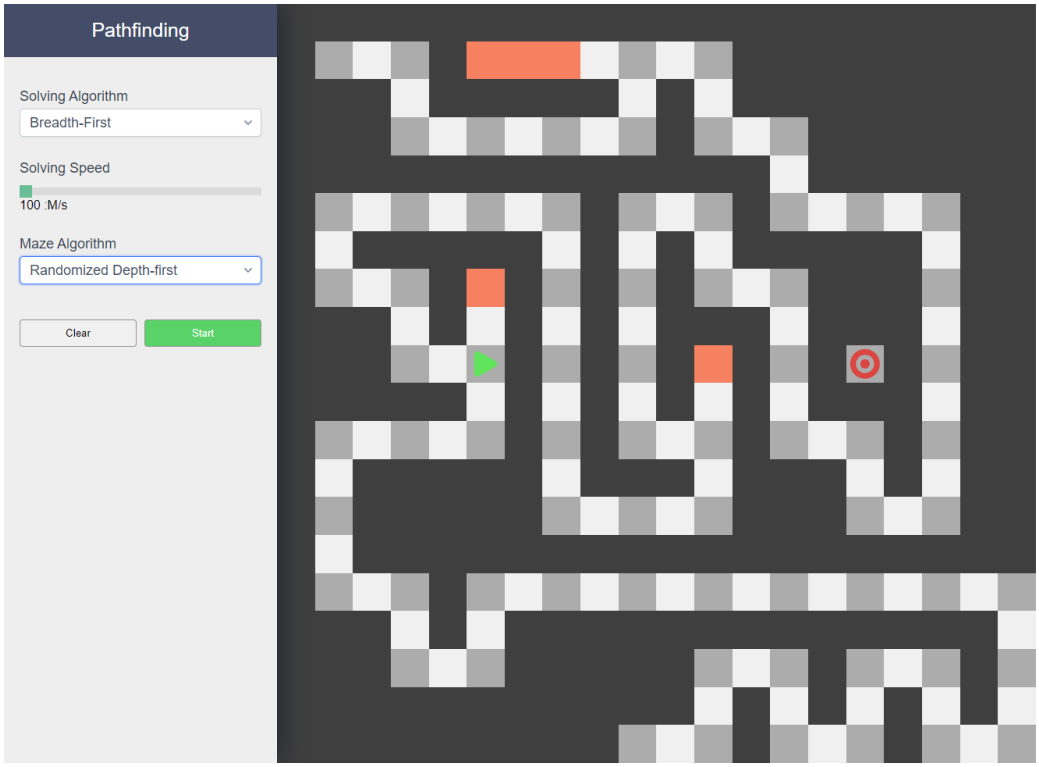





Pathfinding
Solving Algorithm
Breadth-First
Solving Speed
100 :M/s
Maze Algorithm
Custom
Clear
Start


Pathfinding
Solving Algorithm
Breadth-First
Solving Speed
100 :M/s
Maze Algorithm
Custom
Clear
Start


Pathfinding
Solving Algorithm
Breadth-First
Solving Speed
100 :M/s
Maze Algorithm
Custom
Clear
Start


Pathfinding
Solving Algorithm
Breadth-First
Solving Speed
100 :M/s
Maze Algorithm
Randomized Depth-first
Clear
Start

11.2 Gantt Chart

Gantt chart shows the time relationship between ‘events’ of the production program has been regarded as revolutionary in management. Gantt charts recognize the total program goals and it should be regarded as a series of interrelated supporting plans(or events) that people can comprehend and follow.

The following figure is the Gantt Chart of Smart Home Powered by Smart Phones. The plan explains the task versus the time they will take to complete.

Gantt Chart

Si. no	Task Name	Duration (days)	Start Design	End Design	july				august				september			
					w 1	w 2	w 3	w 4	w 1	w 2	w 3	w 4	w 1	w 2	w 3	w 4
1	Analysis	9	4/7/2022	12/7/2022												
2	System Design	12	13/7/2022	29/7/2022												
3	Code Design	23	2/8/2022	24/8/2022												
4	Testing	16	25/8/2022	9/9/2022												
5	Documentation	30	1/9/2022	30/9/2022												

12.REFERENCES

1. https://en.wikipedia.org/wiki/Pathfinding

2. https://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

3. A Comprehensive Study on Pathfinding Techniques for Robotics and Video Games https://www.hindawi.com/journals/ijcgt/2015/736138/

4. A* pathfinding algorithm https://www.growingwiththeweb.com/2012/06/a-pathfinding-algorithm.html

5. Pathfinding algorithms : the four Pillars. https://medium.com/@urna.hybesis/pathfinding-algorithms-the-four-pillars-1ebad85d4c6b