

DEPARTMENT OF COMPUTER SCIENCE, IT AND ANIMATION



Project Report

On

“ALGORITHM VISUALIZER”

Submitted by

Mr.Akshay Mule

Mr.Gaurav Sarode

BCA (Sci.) IV Semester

Academic Year-2021-2022

Guided by

Ms. Bhumika Chandkar

Submitted To

**Department of Computer Science, IT and Animation
Deogiri College, Aurangabad 431005.**

DEOGIRI COLLEGE, AURANGABAD

DEPARTMENT OF COMPUTER SCIENCE, IT AND ANIMATION



CERTIFICATE

*It is the certified work of candidates of **BCA(Sci.) II Year** who have satisfactorily completed the project entitled "**Algorithm Visualizer**" for the Partial fulfillment of the Bachelor Degree from Dr. Babasaheb Ambedkar Marathwada University, Aurangabad for the academic year **2021-22**.*

Submitted By

Mr. Akshay Mule

Mr. Gaurav Sarode

In-Charge

Dr. Yogita Patil

Project-Guide

Ms. Bumika Chandkar

HOD

Dr. S.N. Helambe

External Examiner

ACKNOWLEDGEMENT

Many people have contributed in some or the other forms to this project, although our name appears on the cover of this project. We could not have done this Case Study without the assistance or support of each of the following. We thank all of you.

We wish to place on record our deep sense of gratitude towards our project guide **Ms. Bhumika Chandkar Mam** for the constant motivation and valuable help through the project. We also want to express our gratitude towards our Head of the department **Dr. S.N.Helambe**, for his invaluable guidance, insightful advice and continuous encouragement. We also extend our thanks to other faculty members for their co-operation.

We also thank those who have directly as well as indirectly supported us for the completion of this Project.

Thank you,

Mr.Akshay Mule

Mr.Gaurav Sarode

INDEX

SR. NO.	CHAPTER NAME	PAGE NO.
1.	INTRODUCTION TO HTML	1
2.	INTRODUCTION TO CSS	4
3.	INTRODUCTION TO JAVASCRIPT	7
4.	INTRODUCTION TO PHP	9
5.	INTRODUCTION TO SQL	11
6.	INTRODUCTION TO PROJECT	13
7.	SYSTEM ANALYSIS STUDY REPORT	14
8.	SCOPE OF SYSTEM	15
9.	HARDWARE & SOFTWARE REQUIREMENT	16
10.	DFD (DATA FLOW DIAGRAM)	17
11.	ENTITY REALATIONSHIP (ER) DIAGRAM	19
12.	FEASIBILITY STUDY	21
13.	TESTING	22
14.	DATA DICTIONARY	23
15.	SOURCE CODE	25
16.	OUTPUT	94
17.	CONCLUSION	101
18.	BIBLIOGRAPHY	102
19.	WEB REFERENCES	103

1. INTRODUCTION TO HTML

HTML is a short form of Hyper Text Mark-up Language. HTML was discovered by Tim Berners-Lee in 1991. It is used to design web pages using mark-up language. HTML is the combination of Hypertext and Mark-up language. Hypertext defines the link between the web pages. Mark-up language is used to define the text document within tag which defines the structure of web pages. This language is used to annotate (make notes for the computer) text so that a machine can understand it and manipulate text accordingly. Most of mark-up (e.g. HTML) languages are human readable. Most of mark-up (e.g. HTML) languages are human readable. Language uses tags to define what manipulation has to be done on the text. HTML is used by the browser to manipulate text, images and other content to display it in required format. The first ever version of HTML was HTML 1.0 but the first standard version was HTML 2.0 which was published in 1999.

1.1 HTML page structure

It contains some elements like head, title, body, etc. These elements are used to build the blocks of web pages.

The Basic structure of HTML page is given below:

<DOCTYPE! Html>: This tag is used to tell the HTML version. This currently tells that the version is HTML 5.

<Html>: This is called HTML root element and used to wrap all the code.

<Head>: Head tag contains metadata, title, page CSS etc.

HTML elements that are used inside the <head> tag are:

<Style>

<Title>

<Base>

<No script>

<Script>

<Meta>

<Title>

<Body>:

Body tag is used to enclose all the data which a web page has from texts to links. All of the content that you see rendered in the browser is contained within this element. Example: HTML page can be created using any text editor (notepad). Then save that file using .html or html extension and open that file in browser. It will get the HTML page response.

1.2 Features of HTML are:

- It is easy to learn and easy to use.
- It is platform independent.
- Images, video and audio can be added to a web page.
- Hypertext can be added to text.
- It is a markup language.

1.3 Why learn HTML?

- It is a simple mark-up language. Its implementation is easy.
- It is used to create a website.
- Helps in developing fundamentals about web programming.
- Boost professional career.

1.4 Advantages:

- HTML is used to build a websites.
- It is supported by all browsers.
- It can be integrated with other languages like CSS, JavaScript etc.

1.5 Disadvantages:

- HTML can create only static webpages so for dynamic web page other languages have to be used.
- Large amount of code has to be written to create a simple web page.
Security feature is not good.

Syntax:

```
<Html>  
  <head>  
    <title>
```

```
        </title>
    </head>
    <body>

    </body>
</html>
```

2. INTRODUCTION TO CSS

Cascading Style Sheets is referred as CSS. It is a simple design language used for process of making web pages presentable. CSS handles the look and feel part of a web page. Using CSS, you can control the colour of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colours are used, layout designs, and variations in display for different devices and screen sizes as well as a variety of other effects. CSS is easy to learn and understand but it provides powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the mark-up languages HTML or XHTML.

2.1. What are Style sheets?

In the late '90s, HTML coders noticed that they were retyping the same old tags again and again on the same page, leading to bigger HTML files and above all, time consumption and frustration. You may have found yourself in the same situation, adding in mountains of `` tags, despite wanting them all the same; or using tricks like invisible gifs for spacing. Then, someone had a great idea: have one file that defines all the values that those piles of tags would have done, and then have all your pages checking this file and formatting your pages accordingly. You can therefore leave out most of the formatting tags in HTML and use only nice structural elements (like headings, paragraphs and links) — separating structure and presentation. In late 1996 CSS (Cascading Style Sheets) became a reality, forged by our good friends the » World Wide Web Consortium (W3C). Your style sheet acts as a partner to your HTML code; taking care of all the layout, fonts, colours and overall *look* of your site. If you ever decide to change the look of your site, you modify that one CSS file (your *style sheet*) and all the HTML pages reading from that file will display differently. This makes maintenance of your design much easier.

Another of CSS's boons is that you define things *once*, making it far more **efficient** than defining everything in HTML on every page.

2.2 Benefits:

- **Pages download faster**, sometimes by as much as 50%.
- You have to type less code, and your pages are shorter and neater.

2.3 Types of CSS:

1. Inline CSS
2. Internal CSS
3. External CSS

I. Inline CSS

In Inline CSS every style content is in HTML elements. It is used for a limited section. Whenever our requirements are very small we can use inline CSS. It will affect only single elements. In HTML we require that various HTML tags. Views are different so then we use inline Cascading Style Sheets.

There is a disadvantage of inline Cascading Style Sheets. It must be specified on every HTML tag. There is a lot of time consumed by that and it is not the best practice for a good programmer and the code will be quite large and very complex.

II. Internal CSS

In Internal CSS the style of CSS is specified in the <head> section. This is internal CSS, it affects all the elements in the body section. Internal CSS is used in the condition when we want a style to be used in the complete HTML body. For that, we can use the style in the head tag.

III. External CSS

In External CSS we create a .cuss file and use it in our HTML page as per our requirements. Generally, external Cascading Style Sheets are used whenever we have many HTML attributes and we can use them as required; there is no need to rewrite the CSS style again and again in a complete body of HTML that inherits the property of the CSS file.

Implementation:

CSS files are termed “cascading” style sheet because of two reasons: one style sheet can cascade, or have influence over, multiple pages. Similarly, many CSS files can define a single page.

There are 3 ways to implement css commands into your site:

- Use one CSS file for all your pages. This is the best way to do it.
- Integrate CSS commands into the head of each of your documents.
Use the style attribute to put CSS code directly into a HTML element.

Advantages of CSS:

- CSS saves time – you can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- Pages load faster – If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
- Easy maintenance – To make a global change, simply change the style, and all elements in all the web pages will be updated automatically

3. INTRODUCTION TO JAVASCRIPT

JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform.

3.1 Why to Learn JavaScript?

JavaScript is a MUST for students and working professionals to become a great Software Engineer especially when they are working in Web Development Domain. JavaScript is the most popular programming language in the world and that makes it a programmer’s great choice. Once you learnt JavaScript, it helps you developing great front-end as well as back-end software using different JavaScript based frameworks like query, Node.JS etc.

JavaScript is everywhere, it comes installed on every modern web browser and so to learn JavaScript you really do not need any special environment setup. For example Chrome, Mozilla Firefox, Safari and every browser you know as of today, supports JavaScript. JavaScript helps you create really beautiful and crazy fast websites. You can develop your website with a console like look and feel and give your users the best Graphical User Experience. JavaScript usage has now extended to mobile app development, desktop app development, and game development. This opens many opportunities for you as JavaScript Programmer.

Due to high demand, there is tons of job growth and high pay for those who know JavaScript. You can navigate over to different job sites to see what having JavaScript skills look like in the job market.

There are many useful JavaScript frameworks and libraries available:

1. Angular is
2. React
3. JQuery
4. Vue.js
5. Ext.js
6. Ember.js
7. Meteor
8. Mithril
9. Node.js
10. Polymer
11. Aurelia
12. Backbone.js

Applications of JavaScript Programming

JavaScript is one of the most widely used programming languages (Front-end as well as Backend). It has its presence in almost every area of software development.

There are some applications of JavaScript:

1. Clientside validation - This is really important to verify any user input before submitting it to the server and JavaScript plays an important role in validating those inputs at front-end itself.

2. Manipulating HTML Pages - JavaScript helps in manipulating HTML page on the fly. This helps in adding and deleting any HTML tag very easily using JavaScript and modify your HTML to change its look and feel based on different devices and requirements.

3. User Notifications - You can use JavaScript to raise dynamic pop-ups on the webpages to give different types of notifications to your website visitors.

4. Back-end Data Loading - JavaScript provides Ajax library which helps in loading backend data while you are doing some other processing. This really gives an amazing experience to your website visitors.

5. Presentations - JavaScript also provides the facility of creating presentations which gives website look and feel. JavaScript provides Reveal JS and Bespoke JS libraries to build a webbased slide Presentations.

6. Server Applications - Node JS is built on Chrome's JavaScript runtime for building fast and scalable network applications. This is an event Based library which helps in developing very sophisticated server applications including Web Servers.

This list goes on, there are various areas where millions of software developers are happily using JavaScript to develop great websites and others software's. **Syntax**

```
<script type='text/JavaScript'>
```

4. INTRODUCTION TO PHP

The PHP Hypertext Pre-processor (PHP) is a programming language that allows web developers to create dynamic content that interacts with databases. PHP is basically used for developing web based software applications.

4.1 Why to Learn PHP?

PHP started out as a small open source project that evolved as more and more people found out how useful it was. Rasmus Lerdorf unleashed the first version of PHP way back in 1994.

- PHP is a MUST for students and working professionals to become a great Software Engineer especially when they are working in Web Development Domain.

- PHP is a recursive acronym for "PHP: Hypertext Pre-processor".
- PHP is a server side scripting language that is embedded in HTML. It is used to manage dynamic content, databases, session tracking, even build entire e-commerce sites.
- It is integrated with a number of popular databases, including MySQL, PostgreSQL, Oracle, Sybase, Informix, and Microsoft SQL Server.
- PHP is pleasingly zippy in its execution, especially when compiled as an Apache module on the UNIX side. The MySQL server, once started, executes even very complex queries with huge result sets in record-setting time.
- PHP supports a large number of major protocols such as POP3, IMAP, and LDAP. PHP4 added support for Java and distributed object architectures (COM and CORBA), making n-tier development a possibility for the first time.
- PHP is forgiving: PHP language tries to be as forgiving as possible.
- PHP Syntax is C-Like.

4.2 Common uses of PHP:

- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, and modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.
- It can encrypt data.

4.3 Characteristics of PHP:

Five important characteristics make PHP's practical nature possible –

- Simplicity
- Efficiency ➤ Security
- Flexibility

➤ Familiarity

Syntax

<? php//PHPcodegoeshere?>

Applications of PHP:

- PHP is one of the most widely used languages over the web. There are some applications of php.
- PHP performs system functions, i.e. from files on a system it can create, open, read, write, and close them.
- PHP can handle forms, i.e. gather data from files, save data to a file, through email you can send data, return data to the user.
- You add, delete, and modify elements within your database through PHP.
- Access cookies variables and set cookies.
- Using PHP, you can restrict users to access some pages of your website.

5. INTRODUCTION TO SQL

SQL is a database computer language designed for the retrieval and management of data in a relational database. SQL stands for Structured Query Language. SQL is the language used to create, edit and manipulate a database. In other words, SQL is used to manage data held within a relational database management system (RDBMS). Because this is a database design series, we will not be working with SQL directly, but will design our database to work with SQL in future (once it is completely designed and ready to be programmed).

SQL is the general language used to communicate with relational database management systems. This means that we use SQL to communicate to MySQL, Oracle, SQL Server, etc... So learning about SQL will help you with a lot of different things! A RDBMS takes SQL and uses it to do something with the database.

Why to Learn SQL?

SQL is Structured Query Language, which is a computer language for storing, manipulating and retrieving data stored in a relational database. SQL is the standard language for Relational Database System. All the Relational Database Management Systems (RDMS) like MySQL, MS Access, Oracle, Sybase, Informix, Posture's and SQL Server use SQL as their standard database language.

Also, they are using different dialects, such as –

- MS SQL Server using T-SQL,
- Oracle using PL/SQL,

MS Access version of SQL is called JET SQL (native format) etc.

What Can SQL do?

- SQL can execute queries against a database.
- SQL can retrieve data from a database.
- SQL can insert records in a database.
- SQL can update records in a database.
- SQL can delete records from a database.
- SQL can create new databases.
- SQL can create new tables in a database.
- SQL can create stored procedures in a database.
- SQL can create views in a database.
- SQL can set permissions on tables, procedures, and views.

Applications of SQL

SQL is one of the most widely used query language over the databases. There is some applications of SQL.

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.

- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

6. INTRODUCTION TO PROJECT

Our Project is on **Algorithm Visualizer**. As we all know visualization is the best thing to learn anything faster than learning from books of any text form of information. It is widely perceived that algorithm visualizations can provide a powerful alternative to static written presentations (from textbooks) or verbal descriptions supported by illustrations (from lectures). There has been some debate in the literature as to whether algorithm visualizations are effective in practice. Some studies have shown the classic dismissal that is the downfall of most technological interventions in education: “no significant difference”. Other studies have shown that algorithm visualizations can indeed improve understanding of the fundamental data structures and algorithms that are part of a traditional computer science curriculum. Certainly, many visualizations exist and are widely (and freely) available via the Internet. Unfortunately, the vast majority of those currently available serve no useful pedagogical purpose.

How do you get something done? You don't have to be extremely complex in solving the problem, for example, if your car's headlight is broken (although nowadays, manufacturers are trying the patience of the community with their increasingly abstract, space-age designs). The main issue is figuring out the best way to go about it. To locate step-by-step directions in your car's handbook, you conduct research, or do you use instinct to find someone who knows how to do it? In short, my instinct tells me that I am a visual learner and hence more suited to acquire topics by watching them than by reading about them. In this case, I found that seeing the data move to its rightful spot as the result of an algorithm is MUCH easier to follow than looking at the source code and trying to figure out where the data was supposed to go. My project was born out of my curiosity about sorting algorithms, which inspired the

idea for this paper, which details an online tool I built that explains how sorting algorithms transform and organize sets of data. It is possible to organize a list of people, for example, by their age in ascending order using different methods. To aid my visualization, I created a histogram of numerical data to represent four of the well-known examples. Each number is depicted as a bar, and the height of each bar represents the value of that number. It is being shifted by the algorithm from its original, unordered location to its final ordered place, making it distinct from the rest of the data. Selection Sort, Bubble Sort, Insertion Sort, and Merge Sort are the four sorting algorithms. 2 Let's imagine that you have printed each person's age on a separate index card. Bring the youngest card to the front and then sort the cards by age. To discover the next smallest item, identify the age that has already been ordered and position it behind the already ordered age. Index cards full of ages will be at the end of the pile. Selection Sort works in the same way as this. In this case, to sort a set of data, you select the smallest first, and then the next smallest, and so on until you've sorted all of the data. This technique is quite simple to explain to someone in conversation, but more advanced sorting algorithms, such as Quick Sort, which requires the data to be moved around a pivot point, are not easy to grasp using text alone. I wanted the animation to appeal to a wide spectrum of individuals utilizing various technology media, and so I had it made in a web-based format. Instead of requiring the user to install extra software or attempt to organize setups to use the tool, this helps to remove this source of anxiety.

7. SYSTEM ANALYSIS STUDY REPORT

I. User Interview

1. **how much Ram is needed for it** **Ans.** 500 Mb or more.

2. **which processors can support this processor**

Ans. Core 2 duo and all Android phone processor

3. **can this project can edit anybody**

Ans. Only developer can edit it by using their username and password

4. **This project has update regular.**

Ans. Yes .It require updating at some time.

5. **If any problem than any support will give you.**

Ans. We will provide all support need to you.

6. **How much data speed required for this project** **Ans.** minimum 100 kbps speed required.

7. **Are all internet browser support this site?**

Ans. Yes all browser support.

8. SCOPE OF SYSTEM

There are two principal applications of algorithm visualization: research and education. Potential benefits for researchers are based on expectations that algo-rithm visualization may help uncover some unknown features of algorithms. For example, one researcher used a visualization of the recursive Tower of Hanoi algo-rithm in which odd- and even-numbered disks were colored in two different colors. He noticed that two disks of the same color never came in direct contact during the algorithm's execution. This observation helped him in developing a better non-recursive version of the classic algorithm. To give another example, Bentley and McIlroy [Ben93] mentioned using an algorithm animation system in their work on improving a library implementation of a leading sorting algorithm.

The application of algorithm visualization to education seeks to help students learning algorithms. The available evidence of its effectiveness is decisively mixed. Although some experiments did register positive learning outcomes, others failed to do so. The increasing body of evidence indicates that creating sophisticated software systems is not going to be enough. In fact, it appears that the level of student involvement with visualization might be more important than specific features of visualization software. In some experiments, low-tech visualizations prepared by students were more effective than passive exposure to sophisticated software systems.

To summarize, although some successes in both research and education have been reported in the literature, they are not as impressive as one might expect. A deeper understanding of human

perception of images will be required before the true potential of algorithm visualization is fulfilled.

The scope for the system can be as follows:

- It can use anywhere in the world at any time.
- Easy interface to use the configuration and other settings.
- It helps to understand algorithms faster.
- It provide you a visual interface of any algorithms.

9. HARDWARE & SOFTWARE REQUIREMENT

Hardware Requirement:

- Ram : 512 MB.
- Hard disk : 60 GB.
- Browser : Any.
- Monitor : 15.5 inch color monitor.
- Keyboard : more than 100 keys.

Software Requirement:

- Operating System: Windows 7 or Above.
- Database: SQL server, PHP.
- Browser: Any.
- Visual Studio □ Notepad ++.

11. DFD (DATA FLOW DIAGRAM)

It's easy to understand the flow of data through systems with the right data flow diagram software. A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyse an existing system or model a new one.

Symbols and Notations Used in DFDs

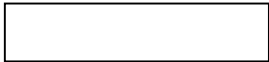
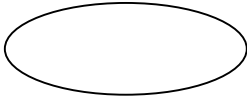

The symbols depict the four components of data flow diagrams:-

External entity: An outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might be an outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on the edges of the diagram.

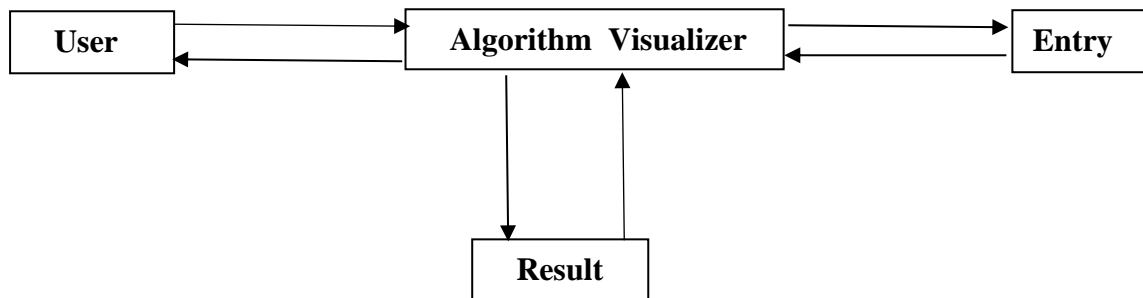
Data store: files or repositories that hold information for later use, such as a database table or only the register person can get appointment form.

Data flow: the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typically

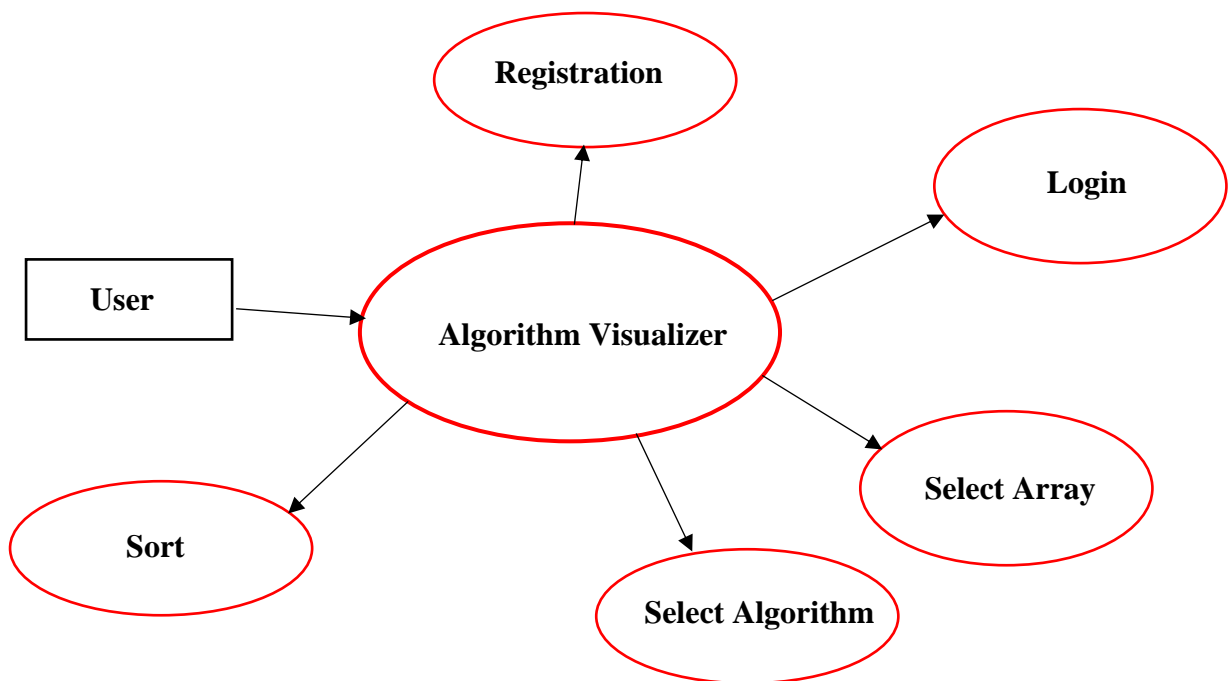
Labelled with a short data name, like "Appointment details."

Notation	Symbol
External Entity	
Process	
Data Flow	

DATA FLOW DIAGRAM (Zero Level)



DATA FLOW DIAGRAM (Level One)



11. ENTITY RELATIONSHIP DIAGRAM (E-R)

The ER or (Entity Relational Model) is a high-level conceptual data model diagram. EntityRelation model is based on the notion of real-world entities and the relationship between them. ER modelling helps you to analyse data requirements systematically to produce a well designed database. So, it is considered a best practice to complete ER modelling before implementing your database.

Components of the ER Diagram

This model is based on three basic concepts:

- Entities
- Attributes •
- Relationships

Entities:

A real-world thing either living or non-living that is easily recognizable and no recognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.

Attributes:

An attribute describes the property of an entity. An attribute is represented as Oval in an ER diagram. There are four types of attributes:

1. Key attribute.
2. Composite attribute.
3. Multivalued attribute.
4. Derived attribute.

Relationships:

Defines the numerical attributes of the relationship between two entities or entity sets.

Different types of cardinal relationships are:

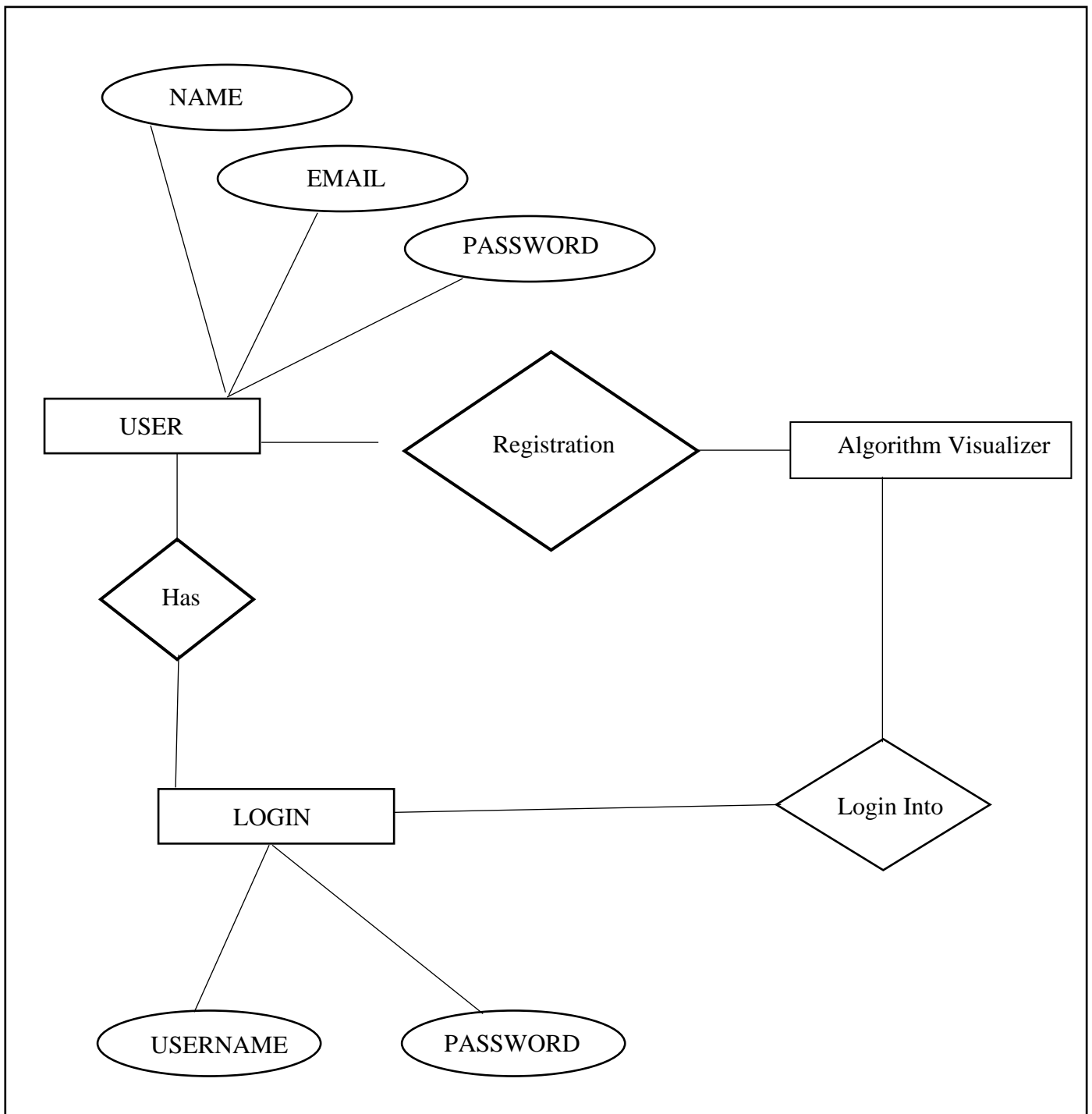
- One-to-One Relationships
- One-to-Many Relationships
- May to One Relationships
- Many-to-Many Relationships

ER- Diagram Notations

ER- Diagram is a visual representation of data that describe how data is related to each other.

- Rectangles: This symbol represent entity types
- Ellipses: Symbol represent attributes
- Diamonds: This symbol represents relationship type
- Lines: It links attributes to entity types and entity types with other relationship types.
- Primary key: attributes are underlined
- Double Ellipses: Represent Multi-valued attribute
- Dashed Ellipses: Derived Attributes
- Double Rectangles: Weak Entity Sets
- Double Lines: Total participation of an entity in a relationship set

ER- Diagram



12. FEASIBILITY STUDY

A feasibility study is carried out to select the best system that must performance requirement And it's working ability in an organization.

The feasibility of the system has been done in three Types:-

- Environmental Feasibility Study

- Economic Feasibility Study
- Technical Feasibility Study

Economic Feasibility:

In this we mainly calculate the cost of proposed system and the cost of customer and compare the cost to meet the users cost. The cost of hardware, facility cost, operating and supply costs are considered. The cost of feasibility is also considered on this such as wiring, lightning, A.C. etc.

Environmental Feasibility:

The environmental feasibility study considers both human and environmental health factors. The FS is a comparative process that looks at all potential solutions, then evaluates them against specific criteria to ultimately find the best choice.

Technical Feasibility:

Technically we have made feasibility in keeping mind about the hardware so that the designed software can work smoothly with maximum efficiency with the hardware.

13. TESTING

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also. **White Box Testing**

White Box Testing is defined as the testing of a software solution's internal structure, design, and coding. In this type of testing, the code is visible to the tester. It focuses primarily on

verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security.

This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors. I tested step wise every piece of code, taking care that every statement in the code is executed at least once. The white box testing is also called Glass Box Testing. **Black Box Testing**

Black box testing is defined as a testing technique in which functionality of the Application under Test (AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software.

This testing method considers a module as a single unit and checks the unit at interface and communication with other modules rather getting into details at statement level. Here the module will be treated as a block box that will take some input and generate output. Output for a given set of input combinations are forwarded to other modules. **Cross browser-testing**

Cross Browser Testing is a process to test web applications across multiple browsers. Cross browser testing involves checking compatibility of your application across multiple web browsers and ensures that your web application works correctly across different web browsers.

14. DATA DICTIONARY USERS

USER LOGIN:

COLUMN_NAME	DATA TYPE	CONSTRAINTS
USERNAME	VARCHAR	UNIQUE KEY
PASSWORD	VARCHAR	UNIQUE KEY

15. SOURCE CODE

Homepage:

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css" />
  <link rel="preconnect" href="https://fonts.googleapis.com" />
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
  <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400&display=swa
p" rel="stylesheet" />
  <style>
    #gridwormCanvas{
      position:absolute;      /* Take the element out of the normal document
flow */
      z-index:-1;              /* Place the element behind normal content */
      top:0;                   /* Start at top of viewport */
      left:0;                  /* Start at left edge of viewport */
    }
  </style>
  <link rel="shortcut icon" type="logo" href="images/ALGORITHM.png" />
  <link rel="stylesheet" href="./css/style.css" />
  <title>Sorting visualizer</title>
</head>

<body>
  <div class="nav-container">
    <div class="navinline">
      <h2 class="title"onclick="window.location.reload()">
        <ul id="logo">
          
          <!-- <button type="button" id="sign">Sign up</button> -->
        </ul>
      </h2>
      <h2 class="text , gradient-text"> Algorithm Visualizer</h2>
    </div>
    <div class="navbar" id="navbar">
      <a id="random" onclick="RenderScreen()">Generate array</a>
      <span class="options">
        <select name="select sort algorithm" id="menu" class="algo-menu">
          <option value="0">Choose algorithm</option>
          <option value="1">Bubble Sort</option>
          <option value="2">Selection Sort</option>
          <option value="3">Insertion Sort</option>
        </select>
      </span>
    </div>
  </div>
</body>
</html>
```

```

        <option value="4">Merge Sort</option>
        <option value="5">Quick Sort</option>
    </select>
</span>
<span class="options">
    <select name="select array size" id="menu" class="size-menu">
        <option value="0">Array size</option>
        <option value="5">5</option>
        <option value="10">10</option>
        <option value="15">15</option>
        <option value="20">20</option>
        <option value="30">30</option>
        <option value="40">40</option>
        <option value="50">50</option>
        <option value="60">60</option>
        <option value="70">70</option>
        <option value="80">80</option>
        <option value="90">90</option>
        <option value="100">100</option>
    </select>
</span>
<span class="options">
    <select name="speed of visualization" id="menu" class="speed-menu">
        <option value="0">Speed</option>
        <option value="0.5">0.50x</option>
        <option value="0.75">0.75x</option>
        <option value="1">1.00x</option>
        <option value="2">2.00x</option>
        <option value="4">4.00x</option>
    </select>
</span>
<a class="start">Sort</a>
<a class="icon"><i class="fa fa-bars"></i></a>
</div>
</div>
<div class="blackbox">

</div>
<!-- partial:index.partial.html -->
<canvas id='gridwormCanvas' width='1350' height='620' style='background-color:
white;' ></canvas>
<!-- partial -->
<script src="./script.js"></script>
<div class="center">
    <div class="array"></div>
</div>

<script src="./scripts/app.js"></script>
<script src="./scripts/helper.js"></script>

```

```

<script src="./scripts/sort-algorithms.js"></script>
<footer class="footer-distributed">

    <div class="footer-right">

        <a href="https://www.facebook.com/akshay.mule.7146/"><i class="fa fa-
facebook"></i></a>
        <a href="https://twitter.com/Aksxy_"><i class="fa fa-twitter"></i></a>
        <a href="https://www.linkedin.com/in/akshay-mule-287811218/"><i
class="fa fa-linkedin"></i></a>
        <a href="https://github.com/aksxy"><i class="fa fa-github"></i></a>

    </div>

</footer>

</body>

</html>

```

Register:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <script
        src="https://kit.fontawesome.com/64d58efce2.js"
        crossorigin="anonymous"
    ></script>
    <link rel="stylesheet" href="style.css" />
    <title>Sign in & Sign up Form</title>
</head>
<body>
    <div class="container">
        <div class="forms-container">
            <div class="signin-signup">
                <form action="#" class="sign-in-form">
                    <h2 class="title">Sign in</h2>
                    <div class="input-field">
                        <i class="fas fa-user"></i>
                        <input type="text" placeholder="Username" />
                    </div>
                    <div class="input-field">

```

```

        <i class="fas fa-lock"></i>
        <input type="password" placeholder="Password" />
    </div>
    <button type="button" class="btn btn-primary"><a href="../../sort-visualizer-master/index.html" style="text-decoration: none; color: white;">Login</a></button>

    <!-- <input type="submit" value="Login" class="btn solid" /> -->
    <p class="social-text">Or Sign in with social platforms</p>
    <div class="social-media">
        <a href="#" class="social-icon">
            <i class="fab fa-facebook-f"></i>
        </a>
        <a href="#" class="social-icon">
            <i class="fab fa-twitter"></i>
        </a>
        <a href="#" class="social-icon">
            <i class="fab fa-google"></i>
        </a>
        <a href="#" class="social-icon">
            <i class="fab fa-linkedin-in"></i>
        </a>
    </div>
</form>
<form action="#" class="sign-up-form">
    <h2 class="title">Sign up</h2>
    <div class="input-field">
        <i class="fas fa-user"></i>
        <input type="text" placeholder="Username" />
    </div>
    <div class="input-field">
        <i class="fas fa-envelope"></i>
        <input type="email" placeholder="Email" />
    </div>
    <div class="input-field">
        <i class="fas fa-lock"></i>
        <input type="password" placeholder="Password" />
    </div>
    <input type="submit" class="btn" value="Sign up" />
    <p class="social-text">Or Sign up with social platforms</p>
    <div class="social-media">
        <a href="#" class="social-icon">
            <i class="fab fa-facebook-f"></i>
        </a>
        <a href="#" class="social-icon">
            <i class="fab fa-twitter"></i>
        </a>
        <a href="#" class="social-icon">

```

```

        <i class="fab fa-google"></i>
      </a>
      <a href="#" class="social-icon">
        <i class="fab fa-linkedin-in"></i>
      </a>
    </div>
  </form>
</div>
</div>

<div class="panels-container">
  <div class="panel left-panel">
    <div class="content">
      <h2>Welcome!</h2>
      <p>
        To The Algorithm Visualizer, Sign-in and get access of your
visualize learning.
      </p>
      <button class="btn transparent" id="sign-up-btn">
        Sign up
      </button>
    </div>
    
  </div>
  <div class="panel right-panel">
    <div class="content">
      <h3>Register First!</h3>
      <p>
        Sign-Up Or Register if you want to get into word of Algorithm
Visualization!
      </p>
      <button class="btn transparent" id="sign-in-btn">
        Sign in
      </button>
    </div>
    
  </div>
</div>
</div>

<script src="app.js"></script>
</body>
</html>

```

DESIGN:

```
* {
```

```

/* Below is the standard CSS code one should add to get rid of default
margins and padding which most of tge HTML elements have */
margin: 0;
padding: 0;
box-sizing: border-box;
font-family: "Roboto", sans-serif;
user-select: none;
}
body {
position: relative;
min-height: 100vh;
text-align: center;
display: flex;
justify-content: space-between;
flex-direction: column;
align-items: stretch;
}

/* Title CSS */
.title {
/*background-color: black; var(--primary-color) */
background: rgb(9,110,190);
background: linear-gradient(90deg, rgba(9,110,190,1) 10%, rgba(224,34,146,1)
100%);
}

/* Navbar CSS */
.navbar {
display: flex;
justify-content: center;
align-items: center;
flex-wrap: wrap;
gap: 0.8em;
font-size: 16px;
min-height: 70px;
padding-block: 0.6em;
/* background-color: */
transition: all 0.5s cubic-bezier(0.645, 0.045, 0.355, 1);
position: relative;
bottom: 20px;
}
.text{
position: relative;
bottom: 36px;
color: white;
}
.navbar a {

```

```

all: unset;
cursor: pointer;
color: #fff;
font-weight: bold;
padding: 8px 10px;
border-radius: 6px;
transition: 0.3s;
background: rgb(9,110,190);
background: linear-gradient(90deg, rgba(9,110,190,1) 10%, rgba(224,34,146,1)
100%);
}
.navbar a:hover {
background-color: transparent;
}
.navbar #menu {
width: fit-content;
outline: none;
border: none;
border-radius: 4px;
padding: 6px 8px;
background-color: rgb(9,110,190);
color: white;
}
.navbar > .icon {
display: none;
}
#menu,
#random,
#start {
cursor: pointer;
}

/* Center css */
.center {
margin: 0 auto;
/*box-shadow: black; /* Added shadow to make page border free */
height: 510px;
width: 600px;
max-height: 731px;
position: relative;
bottom: 25px;
background-color: black ;
opacity: 100%;
border-radius: 0px 0px 15px 15px;
}
.array {
display: flex;
align-items: flex-start;

```



```

    min-height: 100%;
    height: 100%;
    padding: 1rem;
    flex-direction: row;
}
.cell {
    display: flex;
    align-items: flex-start;
    flex: 0.5;
    width: 0.010000%;
    margin: 1px;
    background-color: white;
    resize: horizontal;
    position: relative;
    transition: all 0.4s ease-in;
}
.cell.done {
    background-color: #9cec5b;
    border-color: #9cec5b;
    color: white;
    transition: all 0.4s ease-out;
}
.cell.visited {
    border-color: #6184d8;
    background-color: #6184d8;
    color: white;
    transition: 0.5s;
}
.cell.current {
    border-color: #50c5b7;
    background-color: #50c5b7;
    color: white;
    transition: all 0.4s ease-out;
}
.cell.min {
    background-color: #ff1493;
    border-color: #ff1493;
    color: white;
    transition: all 0.4s ease-out;
}

@media screen and (max-width: 600px) {
    .navbar {
        gap: 0.4em;
    }
    .title {
        font-size: 17px;
    }
}

```

```

}
.navbar *,
.navbar a {
    font-size: 14px;
}
.footer {
    font-size: 18px;
}
a#random {
    order: 4;
}
a.start {
    order: 5;
}
}
@media screen and (max-width: 550px) {
    .center {
        width: 95%;
    }
}
.blackbox{

    width: 200px;
    height: auto;
    background-color: black;
}
/* .blackbox{

    width: 1000px;
    height: 500px;
    padding: 50px;
    background-color: black;
    opacity: 50%;
    border-radius: 20px;

    }*/

.imsz{
    height: 30px;
    width: 30px;
    position: relative;
    right: 610px;
    top: 5px;
}
/*
#sign{
    position: relative;

```

```

        left: 600px;
        bottom: 6px;
        height: 30px;
        width: 70px;
        background: rgba(9,110,190,1);
        border: none;
        border-radius: 5px;
        color: white;
    } */

.footer-distributed {
    /*background-color: #292c2f;*/
    background: rgb(9,110,190);
    /*background: linear-gradient(90deg, rgba(9,110,190,1) 10%,
    rgba(224,34,146,1) 100%); */
    box-shadow: 0 1px 1px 0 rgba(0, 0, 0, 0.12);
    box-sizing: border-box;
    width: 100%;
    height: -5px;
    text-align: left;
    font: normal 16px sans-serif;
    padding: 25px 25px;
}

.footer-distributed .footer-right {
    float: right;
    margin-top: -45px;
    max-width: 180px;
    position: relative;
    bottom: -30px;
    right: 560px;
}

.footer-distributed .footer-right a {
    display: inline-block;
    width: 35px;
    height: 35px;
    background-color: black;
    border-radius: 2px;
    font-size: 20px;
    color: #ffffff;
    text-align: center;
    line-height: 35px;
    margin-left: 7px;
    transition: all .25s;
}

```

```

.footer-distributed .footer-right a:hover{transform:scale(1.1); -webkit-
transform:scale(1.1);}

.footer-distributed p.footer-links a:hover{text-decoration:underline;}

/* Media Queries */

@media (max-width: 600px) {
  .footer-distributed .footer-left, .footer-distributed .footer-right {
    text-align: center;
  }
  .footer-distributed .footer-right {
    float: none;
    margin: 0 auto 20px;
  }
  .footer-distributed .footer-left p.footer-links {
    line-height: 1.8;
  }
}

@import
url("https://fonts.googleapis.com/css2?family=Poppins:wght@200;300;400;500;600
;700;800&display=swap");

* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body,
input {
  font-family: "Poppins", sans-serif;
}

.container {
  position: relative;
  width: 100%;
  background-color: #fff;
  min-height: 100vh;
  overflow: hidden;
}

.forms-container {
  position: absolute;
  width: 100%;
  height: 100%;

```

```

    top: 0;
    left: 0;
  }

.signin-signup {
  position: absolute;
  top: 50%;
  transform: translate(-50%, -50%);
  left: 75%;
  width: 50%;
  transition: 1s 0.7s ease-in-out;
  display: grid;
  grid-template-columns: 1fr;
  z-index: 5;
}

form {
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  padding: 0rem 5rem;
  transition: all 0.2s 0.7s;
  overflow: hidden;
  grid-column: 1 / 2;
  grid-row: 1 / 2;
}

form.sign-up-form {
  opacity: 0;
  z-index: 1;
}

form.sign-in-form {
  z-index: 2;
}

.title {
  font-size: 2.2rem;
  color: #444;
  margin-bottom: 10px;
}

.input-field {
  max-width: 380px;
  width: 100%;
  background-color: #f0f0f0;
  margin: 10px 0;
  height: 55px;
}

```

```

border-radius: 55px;
display: grid;
grid-template-columns: 15% 85%;
padding: 0 0.4rem;
position: relative;
}

.input-field i {
text-align: center;
line-height: 55px;
color: #acacac;
transition: 0.5s;
font-size: 1.1rem;
}

.input-field input {
background: none;
outline: none;
border: none;
line-height: 1;
font-weight: 600;
font-size: 1.1rem;
color: #333;
}

.input-field input::placeholder {
color: #aaa;
font-weight: 500;
}

.social-text {
padding: 0.7rem 0;
font-size: 1rem;
}

.social-media {
display: flex;
justify-content: center;
}

.social-icon {
height: 46px;
width: 46px;
display: flex;
justify-content: center;
align-items: center;
margin: 0 0.45rem;
color: #333;
border-radius: 50%;
}

```

```

border: 1px solid #333;
text-decoration: none;
font-size: 1.1rem;
transition: 0.3s;
}

.social-icon:hover {
  color: #4481eb;
  border-color: #4481eb;
}

.btn {
  width: 150px;
  background-color: #5995fd;
  border: none;
  outline: none;
  height: 49px;
  border-radius: 49px;
  color: #fff;
  text-transform: uppercase;
  font-weight: 600;
  margin: 10px 0;
  cursor: pointer;
  transition: 0.5s;
}

.btn:hover {
  background-color: #4d84e2;
  background: linear-gradient(90deg, rgba(9,110,190,1) 10%, rgba(224,34,146,1) 100%);
}

.panels-container {
  position: absolute;
  height: 100%;
  width: 100%;
  top: 0;
  left: 0;
  display: grid;
  grid-template-columns: repeat(2, 1fr);
}

.container:before {
  content: "";
  position: absolute;
  height: 2000px;
  width: 2000px;
  top: -10%;
  right: 48%;
  transform: translateY(-50%);
}

```

```

    background-image: linear-gradient(-45deg, #4481eb 0%, #04befe 100%);
    background: rgb(224,34,146);
    background: linear-gradient(90deg, rgba(224,34,146,1) 0%, rgba(9,110,190,1)
35%, rgba(9,110,190,1) 63%, rgba(224,34,146,1) 100%);
    transition: 1.8s ease-in-out;
    border-radius: 50%;
    z-index: 6;
}
#img1{
    height: 400px;
    width: auto;
    position: relative;
    left: -70px;
    bottom: -10px;
}
.image {
    width: 100%;
    transition: transform 1.1s ease-in-out;
    transition-delay: 0.4s;
}

.panel {
    display: flex;
    flex-direction: column;
    align-items: flex-end;
    justify-content: space-around;
    text-align: center;
    z-index: 6;
}

.left-panel {
    pointer-events: all;
    padding: 3rem 17% 2rem 12%;
}

.right-panel {
    pointer-events: none;
    padding: 3rem 12% 2rem 17%;
}

.panel .content {
    color: #fff;
    transition: transform 0.9s ease-in-out;
    transition-delay: 0.6s;
}

.panel h3 {
    font-weight: 600;
    line-height: 1;

```



```

    font-size: 1.5rem;
}

.panel p {
    font-size: 0.95rem;
    padding: 0.7rem 0;
}

.btn.transparent {
    margin: 0;
    background: none;
    border: 2px solid #fff;
    width: 130px;
    height: 41px;
    font-weight: 600;
    font-size: 0.8rem;
}

.right-panel .image,
.right-panel .content {
    transform: translateX(800px);
}

/* ANIMATION */

.container.sign-up-mode:before {
    transform: translate(100%, -50%);
    right: 52%;
}

.container.sign-up-mode .left-panel .image,
.container.sign-up-mode .left-panel .content {
    transform: translateX(-800px);
}

.container.sign-up-mode .signin-signup {
    left: 25%;
}

.container.sign-up-mode form.sign-up-form {
    opacity: 1;
    z-index: 2;
}

.container.sign-up-mode form.sign-in-form {
    opacity: 0;
    z-index: 1;
}

```

```

.container.sign-up-mode .right-panel .image,
.container.sign-up-mode .right-panel .content {
  transform: translateX(0%);
}

.container.sign-up-mode .left-panel {
  pointer-events: none;
}

.container.sign-up-mode .right-panel {
  pointer-events: all;
}

@media (max-width: 870px) {
  .container {
    min-height: 800px;
    height: 100vh;
  }
  .signin-signup {
    width: 100%;
    top: 95%;
    transform: translate(-50%, -100%);
    transition: 1s 0.8s ease-in-out;
  }

  .signin-signup,
  .container.sign-up-mode .signin-signup {
    left: 50%;
  }

  .panels-container {
    grid-template-columns: 1fr;
    grid-template-rows: 1fr 2fr 1fr;
  }

  .panel {
    flex-direction: row;
    justify-content: space-around;
    align-items: center;
    padding: 2.5rem 8%;
    grid-column: 1 / 2;
  }

  .right-panel {
    grid-row: 3 / 4;
  }

  .left-panel {
    grid-row: 1 / 2;
  }

```

```

}

.image {
  width: 200px;
  transition: transform 0.9s ease-in-out;
  transition-delay: 0.6s;
}

.panel .content {
  padding-right: 15%;
  transition: transform 0.9s ease-in-out;
  transition-delay: 0.8s;
}

.panel h3 {
  font-size: 1.2rem;
}

.panel p {
  font-size: 0.7rem;
  padding: 0.5rem 0;
}

.btn.transparent {
  width: 110px;
  height: 35px;
  font-size: 0.7rem;
}

.container:before {
  width: 1500px;
  height: 1500px;
  transform: translateX(-50%);
  left: 30%;
  bottom: 68%;
  right: initial;
  top: initial;
  transition: 2s ease-in-out;
}

.container.sign-up-mode:before {
  transform: translate(-50%, 100%);
  bottom: 32%;
  right: initial;
}

.container.sign-up-mode .left-panel .image,
.container.sign-up-mode .left-panel .content {
  transform: translateY(-300px);
}

```

```

}

.container.sign-up-mode .right-panel .image,
.container.sign-up-mode .right-panel .content {
  transform: translateY(0px);
}

.right-panel .image,
.right-panel .content {
  transform: translateY(300px);
}

.container.sign-up-mode .signin-signup {
  top: 5%;
  transform: translate(-50%, 0);
}
}

@media (max-width: 570px) {
  form {
    padding: 0 1.5rem;
  }

  .image {
    display: none;
  }

  .panel .content {
    padding: 0.5rem 1rem;
  }

  .container {
    padding: 1.5rem;
  }

  .container:before {
    bottom: 72%;
    left: 50%;
  }

  .container.sign-up-mode:before {
    bottom: 28%;
    left: 50%;
  }
}

```

JAVASCRIPT CODE:

```
//set up the gridworm
class GridWorm
{
  constructor(point,interval,pointsList,screenWidth,screenHeight)
  {
    this.radius = 2;
    this.xCoord = point.x;
    this.yCoord = point.y;
    this.interval= interval;
    this.color = this.getColor(1,true);//get random color object
    this.mainColor = this.color.color;//color of the head and body of the
    girdworm
    this.mainColorIndex = this.color.index;
    this.nColor = this.getColor(1,true);//get another random color object
    this.arrowHeadColor = this.nColor.color;//color of the arrow points
    at the head of the gridworm
    this.arrowHeadColorIndex = this.nColor.index;
    this.pointsList = pointsList;
    this.screenWidth = screenWidth;
    this.screenHeight= screenHeight;
    this.speed = 5;//the magnitude of the velocity
    this.velocity= this.getVelocity();
    this.junctionMemory = [{point:point,velocity:this.velocity}];//memory
    of each junction visited(helps to construct the worm)
    //the maximum number of junctions a gridworm can keep in memory(this
    determines how long the gridworm will be)
    this.junctionMemoryLength = 6;
  }
  getColor(opacity,isRandom = true,index = 0)
  {
    if(opacity < 0 || opacity > 1 || opacity === null ||
    isNaN(opacity))//if opacity is incorrect
    {
      opacity = 1;
    }
    var colors =
    [
      `rgba(0,0,0,${opacity})`,`rgba(192,192,192,${opacity})`/*silver*/,
      `rgba(128,128,128,${opacity})`/*gray*/,`rgba(128,0,0,${opacity})`/*maroon*/,
      `rgba(255,0,0,${opacity})`/*red*/,`rgba(0,255,0,${opacity})`/*lime
      */`,`rgba(0,0,255,${opacity})`/*blue*/,`rgba(255,0,255,${opacity})`/*fuchsia*/,
      `rgba(128,128,0,${opacity})`/*olive*/,`rgba(0,128,0,${opacity})`/*
      green*/,`rgba(128,0,128,${opacity})`/*purple*/,
      `rgba(0,128,128,${opacity})`/*teal*/,`rgba(0,0,128,${opacity})`/*n
      avy*/,`rgba(138,57,0,${opacity})`/*brown*/, `rgba(205,133,63,${opacity})`,
```

```

        `rgba(244,164,96,{opacity})`,`rgba(139,105,30,{opacity})`,`rgba(
165,42,42,{opacity})`,`rgba(178,34,34,{opacity})`,`
        `rgba(220,20,60,{opacity})`,`rgba(255,140,0,{opacity})`,`rgba(25
5,165,0,{opacity})`,`rgba(255,215,0,{opacity})`,`rgba(184,134,11,{opacity})`
    `,
        `rgba(218,165,32,{opacity})`,`rgba(218,165,32,{opacity})`,`rgba(
238,232,170,{opacity})`,`rgba(189,183,107,{opacity})`,`rgba(240,230,140,{op
acity})`,`
        `rgba(0,100,0,{opacity})`,`
`rgba(34,139,34,{opacity})`,`rgba(32,178,170,{opacity})`,`rgba(47,79,79,{op
acity})`,`
        `rgba(0,139,139,{opacity})`,`rgba(95,158,160,{opacity})`,`rgba(7
0,130,180,{opacity})`,`rgba(25,25,112,{opacity})`,`
        `rgba(0,0,128,{opacity})`,`rgba(0,0,139,{opacity})`,`rgba(72,61,
139,{opacity})`,`rgba(75,0,130,{opacity})`,`rgba(139,0,139,{opacity})`,`
        `rgba(0,0,0,{opacity})`,`rgba(105,105,105,{opacity})`,`
`rgba(169,169,169,{opacity})`
    ];
    if(isRandom)
    {
        let index = Math.floor(this.getRandomNumber(0,colors.length-1));
        let color = colors[index];
        return {color:color,index:index};
    }
    else//if specific
    {
        if(index >=0 && index < colors.length)
        {
            return colors[index];
        }
        return colors[0];
    }
}
getVelocity()
{
    let x,y;
    //flip a coin to decide if gridworm moves vertically or horizontally
    if( Math.random() > 0.5)//if gridworm moves vertically
    {
        x = 0;//no horizontal movement
        y = Math.random() > 0.5? -this.speed: this.speed;//flip a coin to
decide if gridworm moves upwards or downwards
    }
    else//if gridworm moves horizontally
    {
        x = Math.random() > 0.5? -this.speed: this.speed;//flip a coin to
decide if gridworm moves left or right
        y = 0;//no vertical movement
    }
}

```

```

        return {x:x, y:y};
    }
    /**
     * Returns a random number between min (inclusive) and max (exclusive)
     * @param {number} min The lesser of the two numbers.
     * @param {number} max The greater of the two numbers.
     * @return {number} A random number between min (inclusive) and max
(exclusive)
     */
    getRandomNumber(min, max)
    {
        return Math.random() * (max - min) + min;
    }
    drawCircle(x,y,circleradius,ctx,colorIndex)
    {
        for(let i = 0; i < 3; i++)
        {
            let color    = '';
            let radius = 0;
            switch(i)//create three circles with same center
            {
                case 0:
                    radius =circleradius;//smallest circle
                    color  = this.getColor(1,false,colorIndex);
                    break;
                case 1:
                    radius =circleradius * 2;//bigger circle
                    color  = this.getColor(0.5,false,colorIndex);
                    break;
                case 2:
                    radius =circleradius * 6;//biggest circle
                    color  = this.getColor(0.2,false,colorIndex);
                    break;
            }
            //draw the node
            ctx.beginPath();
            ctx.arc(x,y,radius,0,2*Math.PI);
            ctx.fillStyle = color;
            ctx.fill();
            ctx.strokeStyle = color;
            ctx.stroke();
        }
    }
    drawArrowHead(x,y,circleradius,ctx,colorIndex)
    {
        let points = [];
        if(this.velocity.x === 0)//if gridworm is moving vertically
        {
            if(this.velocity.y > 0)//if gridworm is moving down

```

```

        {
            points.push({x:x+this.interval/3,y:y}); //point to the right
            points.push({x:x-this.interval/3,y:y}); //point to the left
            points.push({x:x,y:y+this.interval/3}); //point below
        }
        else //if gridworm is moving up
        {
            points.push({x:x+this.interval/3,y:y}); //point to the right
            points.push({x:x-this.interval/3,y:y}); //point to the left
            points.push({x:x,y:y-this.interval/3}); //point above
        }
    }
    else //if gridworm is moving horizontally
    {
        if(this.velocity.x > 0) //if gridworm is moving right
        {
            points.push({x:x+this.interval/3,y:y}); //point to the right
            points.push({x:x,y:y-this.interval/3}); //point above
            points.push({x:x,y:y+this.interval/3}); //point below
        }
        else //if gridworm is moving left
        {
            points.push({x:x-this.interval/3,y:y}); //point to the left
            points.push({x:x,y:y-this.interval/3}); //point above
            points.push({x:x,y:y+this.interval/3}); //point below
        }
    }
    //draw a circle about the points that make the arrow head
    for(let i = 0; i < points.length;i++)
    {
        let point = points[i];
        this.drawCircle(point.x,point.y,circleradius/2,ctx,colorIndex);
    }
    this.drawTriangle(points[0],points[1],points[2],ctx); //draw the arrow
head
    }
    drawTriangle(point1,point2,point3,ctx)
    {
        ctx.beginPath();
        ctx.moveTo(point1.x, point1.y);
        ctx.lineTo(point2.x, point2.y);
        ctx.lineTo(point3.x, point3.y);
        ctx.fillStyle = 'rgba(0,0,0,0.1)'; //transparent black
        ctx.fill();
    }
    draw(ctx)
    {
        //draw the head of the gridworm

```



```

        this.drawCircle(this.xCoord,this.yCoord,this.radius/2,ctx,this.mainColorIndex);
        this.drawArrowHead(this.xCoord,this.yCoord,this.radius/2,ctx,this.arrowHeadColorIndex);
        //draw circles and squares at every visited junctions in the gridworm's memory(not RAM)
        for(let i = 0; i < this.junctionMemory.length; i++)
        {
            let junction = this.junctionMemory[this.junctionMemory.length - (i+1)];
            //draw a circle at each junction point
            this.drawCircle(junction.point.x, junction.point.y,this.radius/2,ctx,this.mainColorIndex);
            //draw painted squares at every junction point
            ctx.fillStyle = this.getColor(0.1,false,this.mainColorIndex);
            ctx.fillRect(junction.point.x,junction.point.y,this.interval,this.interval);
        }
        //draw the line connecting head to body
        ctx.strokeStyle = 'black';
        ctx.lineWidth = this.radius;
        ctx.beginPath();
        ctx.moveTo(this.xCoord,this.yCoord);
        //draw a line to link all the visited junctions in the gridworm's memory(not RAM)
        for(let i = 0; i < this.junctionMemory.length; i++)
        {
            //starting from the most recent to the least recent(LIFO)[NB: more like a stack data structure]
            let junction = this.junctionMemory[this.junctionMemory.length - (i+1)];
            ctx.lineTo(junction.point.x, junction.point.y);
        }
        ctx.stroke();
        ctx.closePath();
    }
    update(deltaTime)
    {
        this.junctionMemoryLength = this.junctionMemoryLength < 1? 1: this.junctionMemoryLength;
        //keep the gridworm moving in its current direction
        this.xCoord += this.velocity.x;//if gridworm is going left or right, keep it going
        this.yCoord += this.velocity.y;//if gridworm is going up or down, keep it going
        if(this.xCoord <= this.interval)//if gridworm reaches the leftmost point
        {
            this.xCoord = this.interval;

```

```

        this.velocity.x = -this.velocity.x;//move right
        this.xCoord += this.velocity.x * 3;//nudge it a bit away from the
edge
    }
    if(this.xCoord >= this.screenWidth - this.interval)//if gridworm
reaches the rightmost point
    {
        this.xCoord = this.junctionMemory[this.junctionMemory.length-
1].point.x;
        this.velocity.x = -this.velocity.x;//move left
        this.xCoord += this.velocity.x * 3;//nudge it a bit away from the
edge
    }
    if(this.yCoord <= this.interval)//if gridworm reaches the topmost most
point
    {
        this.yCoord = this.interval;
        this.velocity.y = -this.velocity.y; //move down
        this.yCoord += this.velocity.y * 3;//nudge it a bit away from the
edge
    }
    if(this.yCoord >= this.screenHeight - this.interval)//if gridworm
reaches the lowest point)
    {
        this.yCoord = this.junctionMemory[this.junctionMemory.length-
1].point.y;
        this.velocity.y = -this.velocity.y;//move up
        this.yCoord += this.velocity.y * 4;//nudge it a bit away from the
edge
    }
    let currentCoord = {x:this.xCoord,y:this.yCoord};
    let latestJunction = this.getJunctionReached(currentCoord);
    if(latestJunction !== currentCoord)
    {
        let originalVelocity = this.velocity;
        let newVelocity = this.getVelocity();//flip a coin to decide to
move up and down or right and left
        if(originalVelocity.y === 0 )//if gridworm is moving horizontally
        {
            this.velocity = newVelocity;
            if(newVelocity.y === 0 && newVelocity.x === -
originalVelocity.x )//if it continues the horizontal movement in the opposite
direction
            {
                //don't add the new junction to the memory queue
            }
            else
            {

```

```

        let memory =
{point:latestJunction,velocity:this.velocity};
        if(!this.isInMemory(memory))
        {
            this.junctionMemory.push(memory);//add new memory to
the queue
        }
        //this.junctionMemory.push({point:latestJunction,velocity:
this.velocity});//add new memory to the queue
        }
        //nudge it a bit away from the junction
        this.xCoord += this.velocity.x * 3; //not complete yet. Don't
make it too much or too little.
    }
    else //if gridworm is moving vertically
    {
        this.velocity = newVelocity;
        if(newVelocity.x === 0 && newVelocity.y === -
originalVelocity.y )//if it continues the verticalal movement in the opposite
direction
        {
            //don't add the new junction to the memory queue
        }
        else
        {
            let memory =
{point:latestJunction,velocity:this.velocity};
            if(!this.isInMemory(memory))
            {
                this.junctionMemory.push(memory);//add new memory to
the queue
            }
        }
        //nudge it a bit away from the junction
        this.yCoord += this.velocity.y * 3; //not complete yet. Don't
make it too much or too little.
    }
}
if(this.junctionMemory.length > this.junctionMemoryLength)//if memory
is too long
{
    this.junctionMemory.shift();//remove the first memory
}
}
isInMemory(memory)//check if a junction is in memory
{
    this.junctionMemory.some(function(mem)
    {
        if(mem.point === memory.point)

```

```

        {
            return true;//junction is in memory
        }
        return mem.point === memory.point;
    });
    return false;//junction is NOT in memory
}
getJunctionReached(currentCoord)
{
    for(let i = 0; i < this.pointsList.length; i++)
    {
        let point = this.pointsList[i];
        //if point(junction) is too far away, ignore it
        if(Math.abs(currentCoord.x - point.x) > (2 * this.interval) ||
Math.abs(currentCoord.y - point.y) > (2 *this.interval) )
        {
            continue;
        }
        let distance = this.getDistance(currentCoord,point);
        if(distance <= (this.radius))//if gridworm head is close enough to
a junction
        {
            return point;
        }
    }
    return currentCoord;
}
getDistance(p1,p2)//the distance between two points, p1 and p2
{
    let dx = p1.x - p2.x;
    let dy = p1.y - p2.y;
    let distance = Math.sqrt(dx*dx + dy*dy);
    return distance;
}

/**
 * Let node correspond to window resizing.
 * @param {number} screenHeight The height of the screen.
 * @param {number} screenWidth The width of the screen.
 * @param {number} dy The percentage change in browser window
height
 * @param {number} dx The percentage change in browser window
width .
 */
refreshScreenSize(screenHeight,screenWidth,dx,dy,points)
{
}

```

```

}

//sets up and controls all points and gridworms on the canvas
class Painter
{
  constructor(screenWidth,screenHeight)
  {
    this.screenWidth    = screenWidth;
    this.screenHeight   = screenHeight;
    this.interval       = 40;//interval from one point to the next
    this.points         = this.createPoints(); //coordinates of the
vertices of all squares when the canvas is partitioned
    this.gridWorms      = this.createGridWorms();
    this.color          = this.getRandomColor(0.1);
    document.addEventListener('click',(event)=>//when user clicks on the
canvas
    {
      this.points      = this.createPoints();
      this.gridWorms   = this.createGridWorms();//spawn new gridworms
      this.color       = this.getRandomColor(0.1);
    });
  }
  createGridWorms()
  {
    let gridworms = [],
        numOfGridWorms = 30;
    for(var i = 0; i < numOfGridWorms; i++)
    {
      let point =
this.points[Math.floor(this.getRandomNumber(0,this.points.length-
1))]; //randomly select a point
      gridworms.push(new
GridWorm(point,this.interval,this.points,this.screenWidth,this.screenHeight));
    }
    return gridworms;
  }
  createPoints()//divide the canvas into squares
  {
    let points = [],
        interval = this.interval;//interval from one point to the next
    for(var y = interval; y < this.screenHeight; y+=interval)//get all
points in the grid, starting from the top to the bottom
    {
      if(y+interval > this.screenHeight)//if the next point is beyond
the right edge of the canvas
      {
        continue; //skip
      }
    }
  }
}

```

```

        for(var x = interval; x < this.screenWidth; x+=interval)//all the
while, getting all the horizontal points at each level
    {
        if(x+interval > this.screenWidth)//if the next point is beyond
the bottom edge of the canvas
        {
            continue; //skip
        }
        points.push({x:x,y:y});
    }
    }
    return points;
}
getRandomColor(opacity)
{
    var colors = [
        `rgba(255,0,0,      ${opacity})`,//red
        `rgba(255, 242,0,   ${opacity})`,//yellow,
        `rgba(0,0,255,      ${opacity})`,//blue
        `rgba(255,255,0,    ${opacity})`,//yellow
        `rgba(0,255,255,    ${opacity})`,//cyan
        `rgba(255,0,255,    ${opacity})`,//magenta/fuchsia
        `rgba(192,192,192,  ${opacity})`,//silver
        `rgba(128,128,128,  ${opacity})`,//gray
        `rgba(128,0,0,      ${opacity})`,//maroon
        `rgba(128,128,0,    ${opacity})`,//olive
        `rgba(0,128,0,      ${opacity})`,//green
        `rgba(128,0,128,    ${opacity})`,//purple
        `rgba(0,128,128,    ${opacity})`,//teal
        `rgba(0,0,128,      ${opacity})`,//navy
        `rgba(0, 255, 0,     ${opacity})`,//green
        `rgba(77, 0, 255,    ${opacity})`,//blue
        `rgba(255, 0, 140,   ${opacity})`,//purple
        `rgba(0,255,0,      ${opacity})`,//lime
    ];
    return colors[parseInt(this.getRandomNumber(0, colors.length))];
}
/**
 * Returns a random number between min (inclusive) and max (exclusive)
 * @param {number} min The lesser of the two numbers.
 * @param {number} max The greater of the two numbers.
 * @return {number} A random number between min (inclusive) and max
(exclusive)
 */
getRandomNumber(min, max)
{
    return Math.random() * (max - min) + min;
}
/**

```

```

* Let canvas respond to window resizing.
* @param {number} screenHeight The height of the screen.
* @param {number} screenWidth  The width of the screen.
*/
refreshScreenSize(screenHeight,screenWidth)
{
    if(this.screenHeight !== screenHeight || this.screenWidth !==
screenWidth)//if the screen size has changed
    {
        this.screenHeight = screenHeight;
        this.screenWidth  = screenWidth;
        this.points        = this.createPoints(); //coordinates of the
vertices of all squares when the canvas is partitioned
        this.gridWorms     = this.createGridWorms();
    }
}
update(deltaTime)
{
    this.gridWorms.forEach(function(gridworm)
    {
        gridworm.update(deltaTime);
    });
}
draw(ctx)
{
    /*
    for(var i = 0; i < this.points.length; i++)
    {
        let point = this.points[i];
        ctx.fillStyle = Math.random() > 0.5?
this.color:'white';//creates a disco effect
        ctx.fillRect(point.x,point.y,this.interval,this.interval);
    }
    */
    this.gridWorms.forEach(function(gridworm)
    {
        gridworm.draw(ctx);
    });
}
}

//set everything up
function getBrowserWindowSize()
{
    let win = window,
    doc     = document,
    offset  = 20,//
    docElem = doc.documentElement,
    body    = doc.getElementsByTagName('body')[0],

```

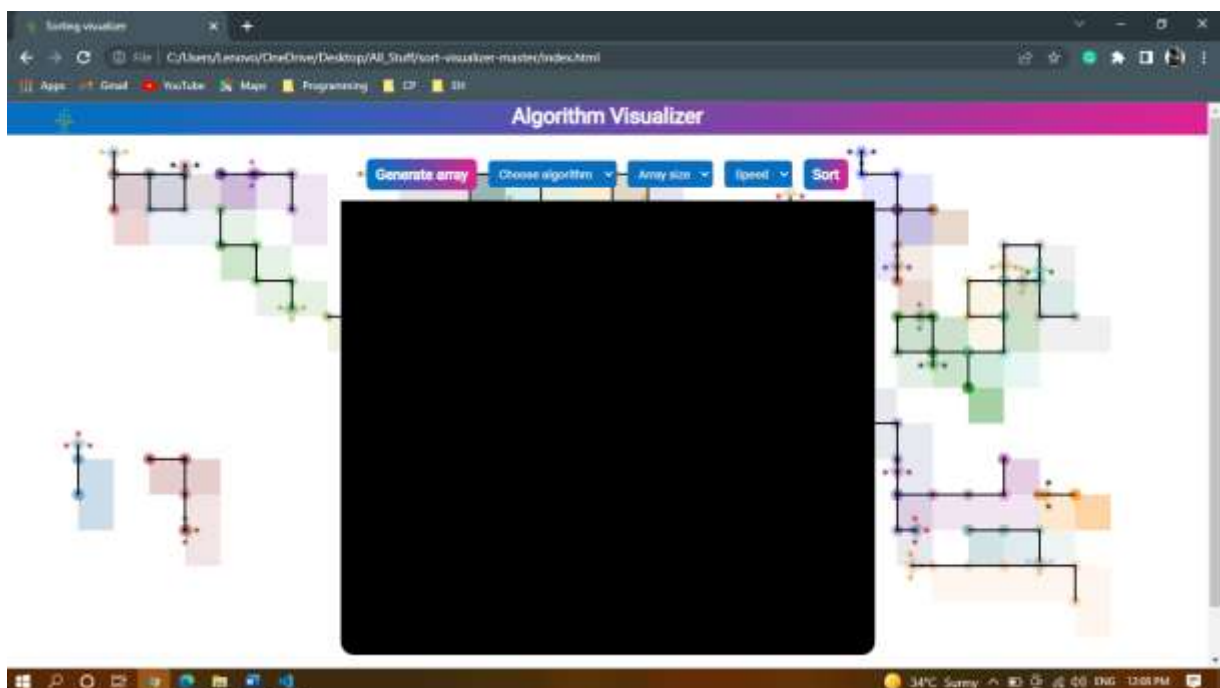
```

    browserWindowWidth = win.innerWidth || docElem.clientWidth ||
body.clientWidth,
    browserWindowHeight = win.innerHeight || docElem.clientHeight ||
body.clientHeight;
    return {x:browserWindowWidth-offset,y:browserWindowHeight-offset};
}
let browserWindowSize = getBrowserWindowSize(),
c = document.getElementById("gridwormCanvas"),
ctx = c.getContext("2d");
//set size of canvas
c.width = browserWindowSize.x;
c.height = browserWindowSize.y;
let SCREEN_WIDTH = browserWindowSize.x,
    SCREEN_HEIGHT= browserWindowSize.y,
    painter = new Painter(SCREEN_WIDTH,SCREEN_HEIGHT),
    lastTime = 100,
    windowSize;
function onWindowResize()//called every time the window gets resized.
{
    windowSize = getBrowserWindowSize();
    c.width = windowSize.x;
    c.height = windowSize.y;
    SCREEN_WIDTH = windowSize.x;
    SCREEN_HEIGHT = windowSize.y;
}
window.addEventListener('resize',onWindowResize);
function updateCanvas()
{
    ctx.clearRect(0,0,SCREEN_WIDTH,SCREEN_HEIGHT);
    ctx.fillStyle = 'white';
    ctx.fillRect(0,0,SCREEN_WIDTH,SCREEN_HEIGHT);
}
function doAnimationLoop(timestamp)
{
    updateCanvas();
    painter.refreshScreenSize(SCREEN_HEIGHT,SCREEN_WIDTH);//let canvas respond
to window resizing
    let deltaTime = timestamp - lastTime;
    lastTime = timestamp;
    painter.update(deltaTime);
    painter.draw(ctx);
    requestAnimationFrame(doAnimationLoop);
}
requestAnimationFrame(doAnimationLoop);

```


16. OUTPUT

Homepage:



Sign-Up:

The screenshot shows a web browser window with the URL `C:\Users\Lenovo\OneDrive\Desktop\All_Stuff\sort-visualizer-master\login%20page\index.html`. The page is titled "Sign up" and features a registration form with the following elements:

- Register First!** header with the text "Sign-Up Or Register if you want to get into word of Algorithm Visualizations!" and a "SIGN IN" button.
- Form fields for "Username", "Email", and "Password".
- A "SIGN UP" button.
- A link "Or Sign up with social platforms" with icons for Facebook, Twitter, Google, and LinkedIn.

The background of the page features a large illustration of a person sitting at a desk with a computer monitor displaying a play button icon.

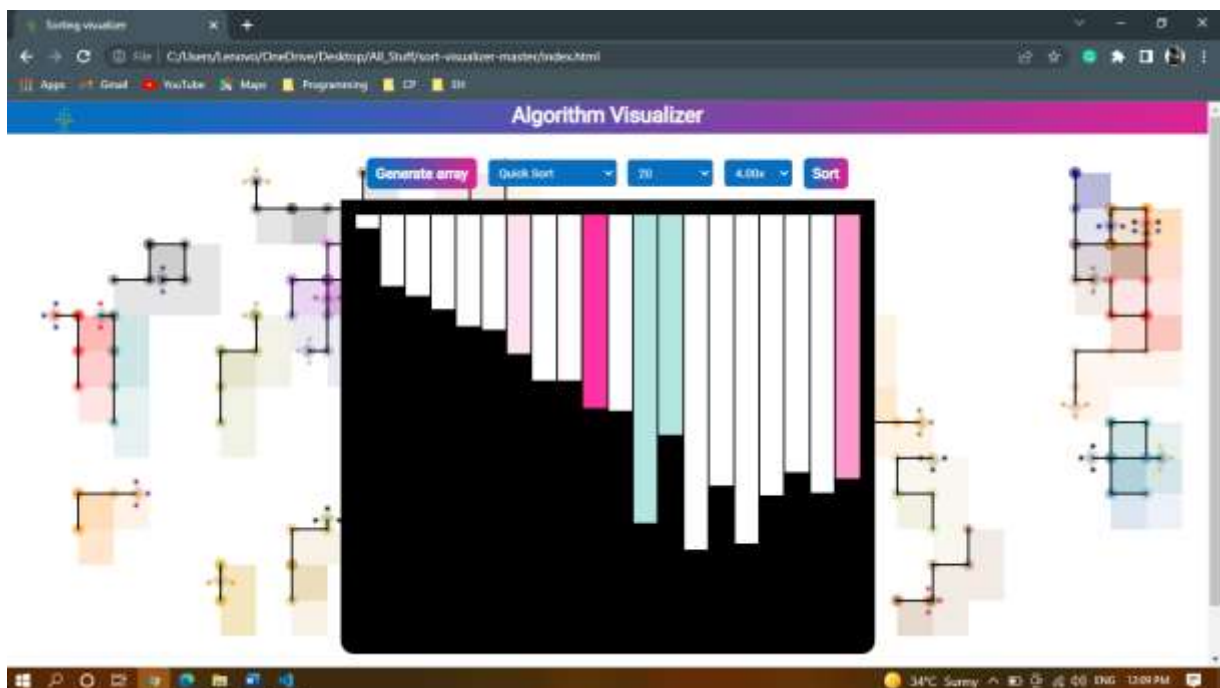
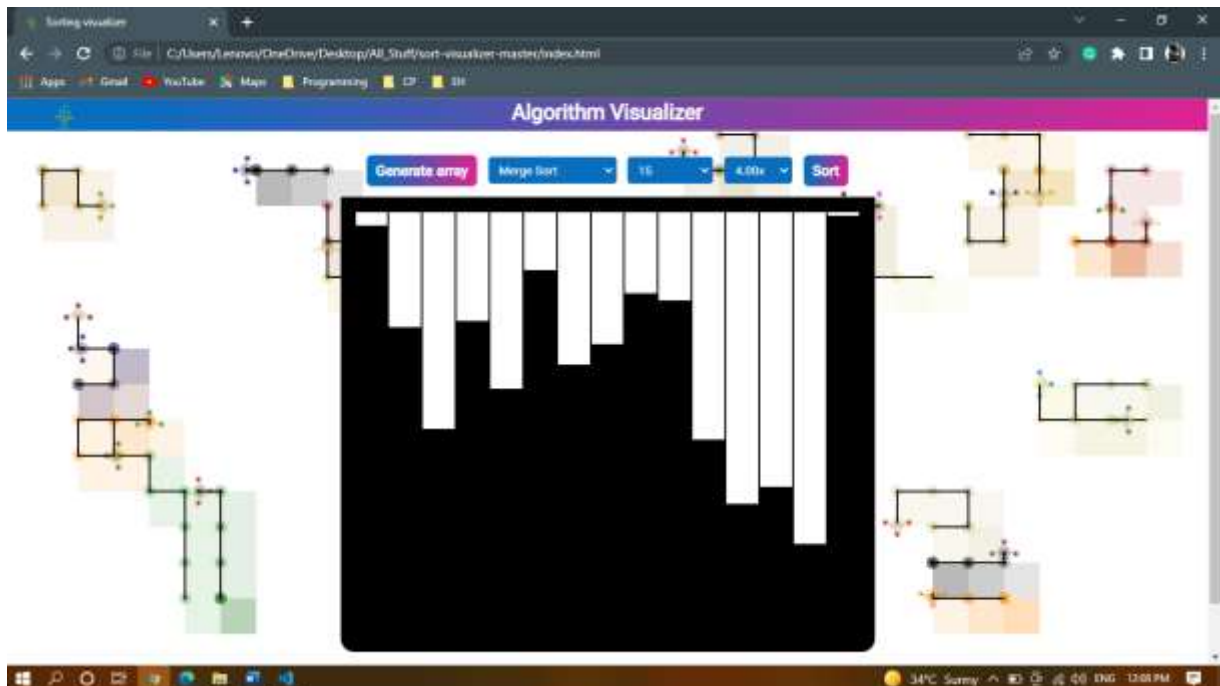
Sign-In:

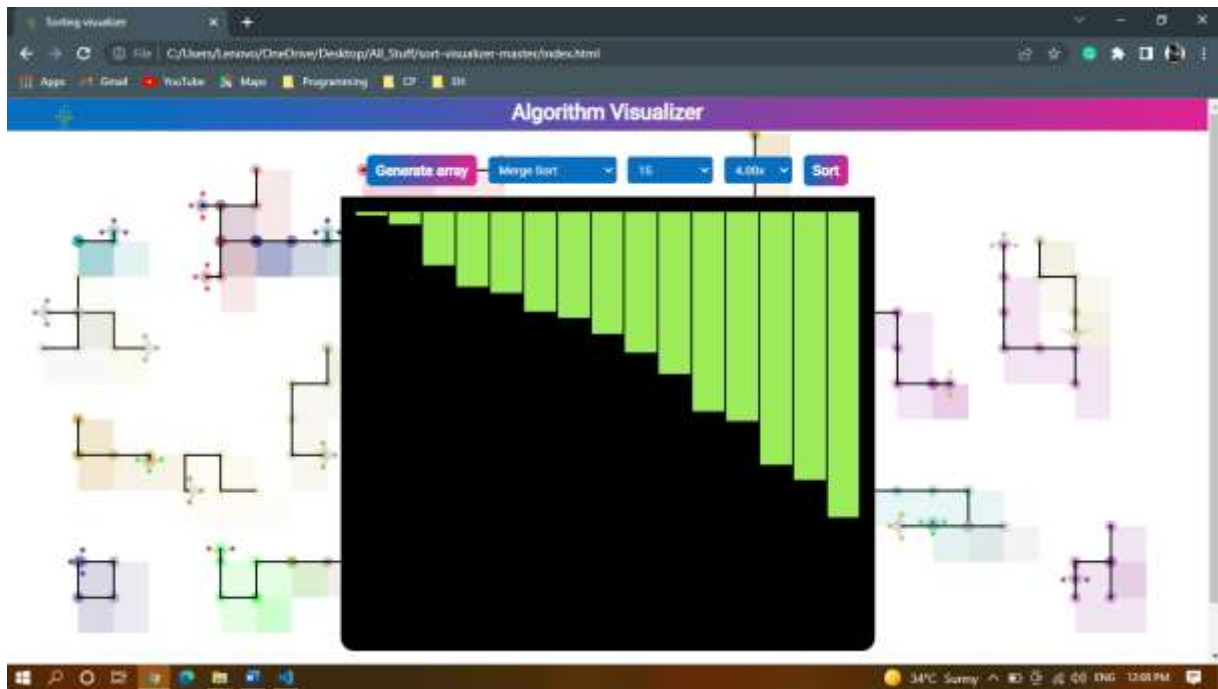
The screenshot shows the same web browser window as the Sign-Up page, but the content is for the login section. The page is titled "Sign in" and features a login form with the following elements:

- Welcome!** header with the text "To The Algorithm Visualizer, Sign-in and get access of your visualize learning" and a "SIGN UP" button.
- Form fields for "Username" and "Password".
- A "LOGIN" button.
- A link "Or Sign in with social platforms" with icons for Facebook, Twitter, Google, and LinkedIn.

The background of the page features a large illustration of a computer screen displaying a flowchart with a yellow decision diamond and a green "YES" box.

Working:





17. CONCLUSION

According to our findings, algorithm visualization can be seen as a valuable supporting tool, used in addition to standard ways of education in the field of computer science. Within the paper we provided an overview of the “Algorithm Visualizer” algorithm visualization platform as well as our practical experiences with the system. We believe (and the results of questionnaire support our belief) it helps to improve the quality of education in the field and contribute to the solution for some of the problems in higher education mentioned at the beginning of the paper. There are still open issues with using algorithm visualizations. Algorithm visualizations can help understanding the principles, but do not replace the need to implement algorithms by students in a chosen programming language. Another drawback of using algorithm visualizations within our subject is the lack of the tool offering required visualizations in a single package with the unified interface. The “Algorithm Visualizer” platform can also be considered as a step in this direction. Generally, more systematic evaluation of algorithm visualization tools is required, as there is rather informal evidence available that applications of algorithm visualizations are useful. We summarized results of the questionnaire filled in by students in order to support our decisions on further

development of the platform, too. Our intentions here include development of new plugin modules from the area of sorting algorithms and more complex data structures. Some of proposed core-related features are on the list too (like graphically better visualizations, optional changing of algorithm properties), but some of them will probably not be implemented in a near future (like undo/step back in running visualization), as they would require more fundamental changes. Except the extensions mentioned within the questionnaire, we also consider some other interesting features: dynamic changes in algorithm pseudocode reflected in visualization, different visual views on running algorithm or simultaneous comparison of different algorithm visualizations.

18. BIBLIOGRAPHY

- **HTML and CSS: Design and Build Websites**-By Jon Docket
- **The Essential Guide to CSS and HTML Web Design (Essentials)**-By Craig Grannel
- **Smashing CSS: Professional Techniques for Modern Layout**-By Eric Meyer
- **The Joy of PHP Programming: A Beginner's Guide**-By Alan Forbes
- **The Complete Beginner's Guide to Learn PHP Programming**-By Bruce Berke
- **New Features And Good Practices**-By Josh Lockhart
- **PHP Advanced and Object-Oriented Programming: Visual QuickPro Guide**-By Larry Ulman
- **SQL Queries for Mere Mortals**-By John L. Viescas and Michael J. Hernandez
- **SQL Performance Tuning**-y peter gulutzan
- **The Art of Sql**-By Stephane Faroult
- **Learn JavaScript VISUALLY**-By Ivelin Demirov

19. WEB REFERENCES

- <https://www.w3schools.com>
- <https://www.geeksforgeeks.org>
- <https://www.tutorialspoint.com>
- <https://www.youtube.com>
- <https://www.html-css-js.com>
- <https://www.js-fiddle.net>
- www.wikipedia.org
- www.youtube.com/webdevelopment
- <https://www.udemy.com>