**SRI SIDDHARTHA ACADEMY OF HIGHER EDUCATION**

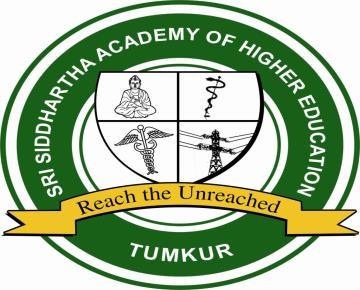
**(DEEMED TO BE UNIVERSITY)**

**Accredited A+ Grade by NAAC**

**SRI SIDDHARTHA INSTITUTE OF TECHNOLOGY**

(A Constituent College of SSAHE)

KUNIGAL ROAD, MARALURU, TUMAKURU-572105

****

**Mini Project-II (IS6MP2)**

**Synopsis on**

**“SORTIFY : VISUALISING ALGORITHMS”**

**BACHELOR OF ENGINEERING**

**Submitted by:**

|  |  |
| --- | --- |
| Chandana R | 22IS016 |
| Charmie J Jain | 22IS018 |

**Under the guidance of:**

**GUIDE NAME guide qualification**

Guide Designation, Dept. of ISE



**DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING**

**“Accredited by NBA, NEW DELHI”**

**2022-25**

**ABSTRACT**

Sorting algorithms are fundamental to computer science, facilitating the efficient organization of data. Understanding their intricacies and performance characteristics is crucial for students, educators, and developers. In this project, we present a Sorting Algorithm Visualizer, a web-based tool designed to provide an interactive platform for exploring various sorting algorithms.

The visualizer offers implementations of six popular sorting algorithms: Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, and Heap Sort. Through a user-friendly interface developed using HTML, CSS, and JavaScript, users can select algorithms, adjust parameters such as array size and sorting speed, and visualize the sorting process in real-time.

The visualizer dynamically updates the display as algorithms execute, providing step-by-step animations that aid in understanding their operations. Additionally, the system calculates and displays the space and time complexities of each algorithm, empowering users to make informed comparisons.

With its modular architecture and emphasis on clarity and interactivity, the Sorting Algorithm Visualizer serves as an educational resource for learners of all levels. By fostering a deeper understanding of sorting algorithms, this project aims to inspire curiosity and facilitate learning in the field of computer science.

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sl.No.** | **Contents** | **Page Number** |
| 1 | Introduction | i |
| 2 | Problem Statement | ii |
| 3 | Proposed Solution | iii |
| 4 | System Architecture | iv |

**I. Introduction**

The Sorting Visualizer is a comprehensive tool that not only demonstrates the functionality of various sorting algorithms but also provides insights into their space and time complexities. This visualizer offers users a streamlined interface to explore six prominent sorting algorithms: Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, and Heap Sort.

Each algorithm's implementation is accompanied by a clear visualization of its step-by-step execution, allowing users to observe its performance firsthand. Additionally, the visualizer provides detailed information on the space and time complexities of each algorithm, empowering users to make informed decisions about algorithm selection based on their specific requirements.

Whether you're a student learning about sorting algorithms, an educator teaching algorithmic concepts, or a developer optimizing algorithm performance, the Sorting Visualizer offers a valuable resource for understanding sorting algorithms and their complexities. Dive in and explore the world of sorting algorithms with ease and clarity.

**II. Problem Statement**

Design and develop a web-based Sorting Algorithm Visualizer that allows users to interactively explore the functionalities and complexities of various sorting algorithms. The visualizer should include implementations of the following sorting algorithms: Bubble Sort, Selection Sort, Insertion Sort, Merge Sort, Quick Sort, and Heap Sort.

Key Features:

1.Interactive Visualization: Provide a dynamic visualization of each sorting algorithm's step-by-step execution, allowing users to observe how elements are sorted in real-time.

2.Algorithm Selection: Allow users to select and visualize the execution of different sorting algorithms, providing options for comparison.

3.Adjustable Parameters: Enable users to adjust parameters such as array size and speed of execution to customize their visualization experience.

4. Space and Time Complexity: Display detailed information on the space and time complexities of each sorting algorithm, helping users understand their efficiency.

5.Documentation and Instructions: Provide clear documentation and instructions on how to use the visualizer effectively, including explanations of sorting algorithms and their complexities.

6.Performance Analysis: Include features for analyzing and comparing the performance of different sorting algorithms based on metrics such as runtime and memory usage.

By addressing these requirements, the Sorting Algorithm Visualizer aims to be a valuable educational tool for students, educators, and developers interested in understanding and exploring sorting algorithms.

**III. Proposed Solution**

For our Sorting Algorithm Visualizer, we'll leverage JavaScript and CSS to create a dynamic and interactive web application. Here's an outline of our approach:

1.Frontend Structure: We'll use HTML to structure the layout of our visualizer, dividing it into sections for algorithm selection, parameter adjustment, visualization display, and complexity information.

2.Styling with CSS: CSS will be used to style the visualizer, providing a clean and appealing interface. We'll use CSS animations and transitions to enhance the visualization experience, with color-coded elements to represent different states during sorting.

3.JavaScript Implementations: Each sorting algorithm will be implemented in JavaScript to enable real-time visualization. We'll create clear and efficient algorithm implementations, focusing on readability and performance.

4.Interactive Visualization: JavaScript will be responsible for animating the sorting process step by step. We'll update the DOM dynamically to reflect the changes in the array as the algorithm progresses.

5.Adjustable Parameters: Users will have options to adjust parameters such as array size and sorting speed using HTML input elements. JavaScript event listeners will capture these changes and update the visualization accordingly.

6.Space and Time Complexity Display: JavaScript will calculate and display the space and time complexities of each sorting algorithm based on their implementations. This information will be dynamically updated as users select different algorithms.

By combining the power of JavaScript for algorithm execution and DOM manipulation with CSS for styling and layout, our Sorting Algorithm Visualizer will provide an engaging and educational experience for users to explore and understand sorting algorithms.

**IV. System Architecture**



**Note:**

Font: Time new roman.

Font Size: Chapter names🡪16

Topics within chapter 🡪14

Subtopics with in topics 🡪12

Contents 🡪12

Figure names and Table names 🡪 12

Margin: Top: 1cm

Bottom: 0.25cm

Inside: 1cm

Outside: 1cm

Borders: Only for Front Sheet, Contents, Abstract.

Line Spacing: 1.5

Paragraph Spacing: 1.5