

# **Design and Implementation of Sorting Algorithm Visualizer**

**A PROJECT REPORT  
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE AWARD OF THE DEGREE OF**

**BACHELOR OF ENGINEERING  
IN  
COMPUTER SCIENCE AND ENGINEERING**

Submitted by

<b>Name</b>	<b>Reg. No.</b>
<b>Venkatesh A</b>	<b>– 211008072</b>
<b>Barathraj B</b>	<b>– 212008004</b>
<b>Chithukati Balaji</b>	<b>– 212008006</b>
<b>Mohamed Akram M N</b>	<b>– 212008015</b>

Project Guide

Dr.C.Anbuananth

**Associate Professor**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**



**ANNAMALAINAGAR – 608 002**

**May 2025**

**ANNAMALAI**



**UNIVERSITY**

**FACULTY OF ENGINEERING AND TECHNOLOGY  
DEPARTMENT OF COMPUTER SCIENCE AND  
ENGINEERING**

This is to certify that the Project report titled “**Design and Implementation of Sorting Algorithm Visualizer** ” is the bona fide record of the work done by

<b>Name</b>		<b>Reg. No.</b>
<b>Venkatesh A</b>	–	<b>211008072</b>
<b>Barathraj B</b>	–	<b>212008004</b>
<b>Chithukati Balaji</b>	–	<b>212008006</b>
<b>Mohamed Akram M N</b>	–	<b>212008015</b>

in partial fulfillment of the requirements for the Degree of Bachelor of Engineering in Computer Science and Engineering during the period 2024 - 2025.

**Dr.C.Anbuananth**  
Associate Professor  
Dept. of Computer Science & Engg.  
Project Guide

**Dr. R. Bhavani**  
Professor & Head  
Dept. of Computer Science & Engg.

Internal Examiner

External Examiner

Place: Annamalainagar  
Date:

## ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to **Dr. R. Bhavani, M.E., Ph.D., Professor & Head**, Department of Computer Science and Engineering, Faculty of Engineering and Technology, Annamalai University for giving us the opportunity to undertake this project.

We would like to convey our heartiest thanks to our project guide **Dr .C .Anbuananth, M.E., Ph.D., Associate Professor**, Department of Computer Science and Engineering, for all his help and support. He with his extreme patience has guided us in situation of need for which we are extremely grateful.

We would like to thank our project review committee members **Dr. G. Ramachandran M.E, Ph.D., Associate Professor, Dr. A. Kanthimathinathan M.E, Ph.D., Associate Professor, Dr. S. Saravanan M.E, Ph.D., Assistant Professor**, and **Dr. P. Anbalagan M.E., Ph.D., Assistant Professor**, Department of Computer Science and Engineering for their great support and encouragement during the course of our project.

We would like to thank all our friends for their support in time of need, encouragement and were always ready to help us without asking for it.

We wish to thank all the technical staff members, incharge of our department laboratories that fulfilled all our project needs and offered us timely help.

Above all, we are indebted to our beloved parents, whose blessings and best wishes have come a long way in making this final year project work a grand success.

<b>Venkatesh A</b>	–	<b>211008072</b>
<b>Barathraj B</b>	–	<b>212008004</b>
<b>Chithukati Balaji</b>	–	<b>212008006</b>
<b>Mohamed Akram M N</b>	–	<b>212008015</b>

## TABLE OF CONTENTS

CHAPTER NO.	CONTENTS		PAGE NO.
	Abstract (English)		i
	Abstract (Tamil)		ii
	List of Figures		iii
	List of Abbreviations		iv
<b>1</b>	<b>INTRODUCTION</b>		<b>1</b>
	1.1	INTRODUCTION	1
	1.2	PROBLEM STATEMENT	6
	1.3	OBJECTIVES	7
	1.4	ORGANIZATION OF THE THESIS	8
<b>2</b>	<b>LITERATURE SURVEY</b>		<b>9</b>
<b>3</b>	<b>METHODOLOGY</b>		<b>16</b>
	3.1	BLOCK DIAGRAM OF THE PROPOSED SYSTEM	16
	3.2	SYSTEM ARCHITECTURE	18
	3.3	SYSTEM REQUIREMENTS	20
	3.4	TOOLS AND TECHNOLOGIES	21
<b>4</b>	<b>IMPLEMENTATION</b>		<b>24</b>
	4.1	MODULAR DESIGN	24
	4.2	DATA INPUT HANDLING	26
	4.3	VISUALIZATION ENGINE	27
	4.4	CONTROL AND INTERACTIVITY	28

	4.5	PSEUDO-CODE HIGHLIGHTING MODULE	28
	4.6	EXPLANATION PANEL	30
	4.7	SORTING ALGORITHMS IMPLIMENTED	31
	4.8	USER EXPERIENCE ENHANCEMENTS	32
<b>5</b>	<b>CONCLUSION</b>		33
	5.1	FUTURE SCOPE	34
	<b>REFERENCES</b>		35

# ABSTRACT

The Sorting Algorithm Visualizer is a web-based interactive application designed to visually demonstrate the internal operations of popular sorting algorithms such as Bubble Sort, Selection Sort, and Insertion Sort. This tool aims to enhance the understanding of sorting techniques by displaying visual cues during element comparisons, swaps, and iterations. Users can select different algorithms, adjust array sizes, and modify sorting speeds to explore the behaviour of each sorting technique under varying conditions. The visualizer employs technologies like HTML, CSS, JavaScript, ReactJS, Material UI, and GSAP to create an intuitive and responsive user interface.

By transforming abstract algorithm concepts into animated, dynamic processes, this project provides learners with a clearer and more intuitive grasp of how sorting mechanisms function step-by-step. The educational value is further emphasized through real-time pseudocode highlighting and detailed explanations, making the learning experience engaging and effective. The visualizer offers flexibility and interactivity, allowing users to experiment with different sorting algorithms and observe their performance in real-time.

This project serves as a valuable resource for both learners and educators, fostering a stronger foundation in algorithmic thinking and problem-solving. It addresses the need for accessible, engaging, and informative learning tools in computer science education. By transforming abstract sorting concepts into interactive experiences, the Sorting Algorithm Visualizer stands as a significant educational tool that enhances the understanding of sorting algorithms through visual and interactive means.

## ABSTRACT IN TAMIL

Sorting Algorithm Visualizer என்பது இணையதள அடிப்படையிலான ஒரு இடைமுக செயலி. இது Bubble Sort, Selection Sort மற்றும் Insertion Sort போன்ற பிரபலமான வகைச் சேர்க்கை ஆல்கொரிதங்களின் செயல்பாடுகளை காட்சிப்படுத்தும் நோக்கத்துடன் உருவாக்கப்பட்டுள்ளது. ஒவ்வொரு ஒப்பீடு, மாற்றம் மற்றும் மீள்விளக்கம் ஆகியவற்றை நேரடி காட்சியாக காட்டுவதன் மூலம், மாணவர்கள் உள் செயல்பாடுகளை புரிந்துகொள்ள உதவுகிறது. பயனர்கள் ஆல்கொரிதங்களை தேர்ந்தெடுத்து, வரிசையின் அளவையும் வேகத்தையும் மாற்றி, பல்வேறு சூழ்நிலைகளில் அதன் செயல்பாட்டை ஆராய முடிகிறது. இந்த கருவி HTML, CSS, JavaScript, ReactJS, MaterialUI மற்றும் GSAP போன்ற தொழில்நுட்பங்களைப் பயன்படுத்துகிறது.

கற்பனை சார்ந்த ஆல்கொரிதக் கருத்துகளை இயக்கம் மற்றும் சுய இயக்கவியல் செயல்முறைகளாக மாற்றுவதன் மூலம், இந்த திட்டம் பயில்பவர்களுக்கு எளிமையான மற்றும் விளக்கமான புரிதலை வழங்குகிறது. நேரடி pseudocode-ஐ வலியுறுத்துவதும், விரிவான விளக்கங்களை வழங்குவதும் மூலம், கற்றல் அனுபவம் சுவாரஸ்யமாகவும் பயனுள்ளதாகவும் மாறுகிறது. இதில் உள்ள இடைமுகம் பயனர்களுக்கு நெகிழ்வான அனுபவத்தையும், நேரடியாக பல்வேறு ஆல்கொரிதங்களைச் சோதித்து விளைவுகளைப் பார்ப்பதற்கான வசதியையும் வழங்குகிறது.

## LIST OF FIGURES

FIGURE NO.	TITLE	PAGE NO.
1.1	Web Technologies	2
1.1.1	Sorting Algorithms	3
3.1	Block Diagram of the Proposed System	16
3.2	MVC Architecture	18
4.1	Modular Design	25
4.2	Custom Input Module	26
4.3	GSAP	27
4.4	Controls Module	28
4.5	Pseudo Code Module	29
4.6	Explanation Module	30
4.7	Sorting Visualizer	31



## LIST OF ABBREVIATIONS

HTML	-	HyperText Markup Language
CSS	-	Cascading Style Sheets
SCSS	-	Sassy Cascading Style Sheets
JS	-	JavaScript
ReactJS	-	React JavaScript
UI	-	User Interface
GSAP	-	GreenSock Animation Platform
MUI	-	Material User Interface
MVC	-	Model-View-Controller
CPU	-	Central Processing Unit
RAM	-	Random Access Memory
HMR	-	Hot Module Reloading

# CHAPTER 1

## INTRODUCTION

### 1.1 INTRODUCTION

Sorting plays a vital role in the field of computer science, serving as a foundation for a wide range of applications, from search optimization and data analysis to graphics, machine learning, and database management. Sorting algorithms, such as Bubble Sort, Selection Sort, and Insertion Sort, are commonly introduced in the early stages of computer science education. However, despite their importance, students often struggle to comprehend how these algorithms work behind the scenes, especially when relying solely on theoretical explanations or textual code representations.

This challenge highlights the need for interactive learning tools that bridge the gap between abstract algorithm concepts and real-world understanding. One such solution is the Sorting Algorithm Visualizer—a web-based interactive application designed to visually demonstrate the internal operations of popular sorting algorithms. By transforming static logic into animated, dynamic processes, this project aims to provide learners with a clearer and more intuitive grasp of how sorting mechanisms function step-by-step.

The primary purpose of this project is to enhance the understanding of sorting techniques by displaying visual cues during element comparisons, swaps, and iterations. The visualizer offers the flexibility to select different algorithms, adjust array sizes, and modify sorting speeds, enabling users to explore the behavior of each sorting technique under varying conditions. Through color-coded bars and real-time animations, users can observe how data elements move, align, and eventually reach a sorted order.

Moreover, this project does not merely focus on visual appeal—it also emphasizes educational value. Each algorithm is implemented with its inherent time and space complexities in mind, allowing users to analyze and compare

performance. For example, learners can observe how Bubble Sort performs inefficiently on larger datasets, while Selection Sort maintains consistent behavior, and Insertion Sort excels with partially sorted arrays.

In today's digital era, learning through visualization has proven to be significantly more effective than traditional methods. The Sorting Algorithm Visualizer not only makes learning engaging but also supports deeper retention of algorithmic principles. The design and implementation of this tool are rooted in technologies such as HTML, CSS, and JavaScript, with an emphasis on intuitive UI, responsive layout, and efficient logic handling.

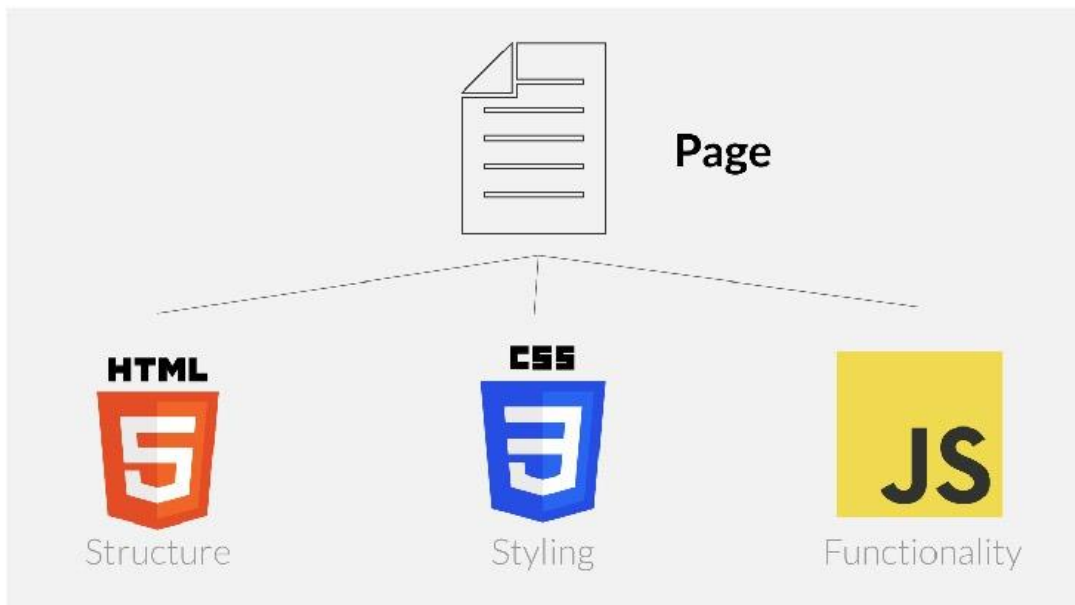


Fig 1.1 Web Technologies

Beyond its educational utility, the project also opens the door to further exploration of algorithm optimization, performance analysis, and interactive UI/UX development. It serves as a platform for aspiring developers to experiment with frontend technologies while simultaneously deepening their understanding of core computer science concepts.

In conclusion, this project addresses the need for accessible, engaging, and informative learning tools in computer science education. By transforming abstract sorting concepts into interactive experiences, the Sorting Algorithm Visualizer

stands as a valuable resource for both learners and educators, fostering a stronger foundation in algorithmic thinking and problem-solving.

### 1.1.1 OVERVIEW OF SORTING ALGORITHMS

Sorting is one of the most fundamental operations in computer science. A sorting algorithm arranges elements in a specific order, usually in numerical or lexicographical order. In computational theory, sorting algorithms are a category of algorithms that take a collection of elements (such as an array or list) and reorder them into a specified sequence, based on either ascending or descending order.

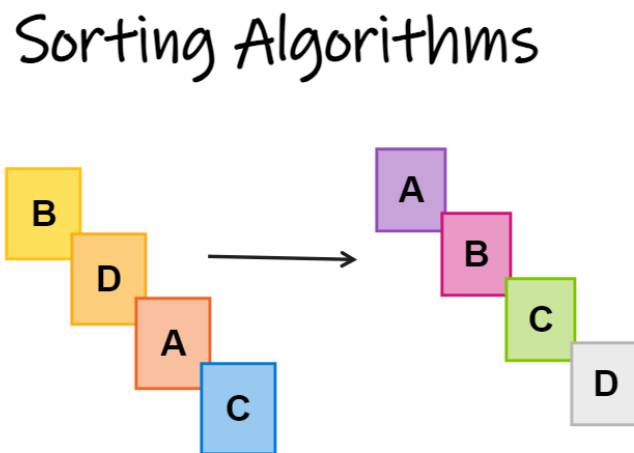


Fig 1.1.1 Sorting Algorithms

The process of sorting is critical in many applications, from organizing data in databases to optimizing search algorithms. Sorting algorithms are used in diverse fields, including data processing, machine learning, and information retrieval systems. The efficiency and behavior of sorting algorithms can have a significant impact on the overall performance of applications, especially when dealing with large datasets.

While sorting may appear as a simple task, the choice of algorithm can greatly affect performance. Different algorithms have varying complexities, from linear time algorithms to those with exponential time complexities. These algorithms can be classified into several categories, such as comparison-based sorting and non-

comparison-based sorting. The major types of sorting algorithms include Bubble Sort, Selection Sort, Quick Sort, Insertion Sort, and Merge Sort.

In the context of this project, visualizing sorting algorithms aims to provide an interactive and educational tool for understanding how these algorithms work. While learning sorting algorithms in theory is beneficial, visualizations help bridge the gap between abstract concepts and their real-world application.

When sorting algorithms are visualized, the steps of each algorithm are depicted through animations that show how data elements are compared, swapped, and rearranged. This helps learners understand key concepts such as selecting and swapping. A visualizer provides immediate feedback, making the algorithm's execution more intuitive. Users can observe how various algorithms behave under different conditions, compare their performance, and see the internal workings of each step. This project aims to make the understanding of sorting algorithms more accessible and engaging by providing an interactive interface.

### **1.1.2 MOTIVATION**

The motivation behind the Sorting Algorithm Visualizer project arises from the challenges that students, developers, and computer science enthusiasts face when learning about sorting algorithms. While theoretical explanations from textbooks and lectures provide the foundation, truly understanding how sorting algorithms operate in a dynamic and interactive manner is essential for mastering the concepts. Sorting algorithms are often explained through static, abstract examples, which can fail to convey the complexity of their execution in a real-world setting.

This project aims to bridge that gap by offering an interactive platform that allows users to visually observe how sorting algorithms work step by step. With the ability to control the input data, select from several common sorting algorithms (such as Bubble Sort, Selection Sort, and Insertion Sort), and view the real-time animation of each step in the sorting process, users can gain a deeper understanding of the underlying mechanics of these algorithms. The visualizer empowers users to

experiment with different scenarios, input sizes, and speeds, enhancing their learning experience and giving them better insight into algorithm efficiency and behavior.

In addition to being a valuable learning tool for students, the Sorting Algorithm Visualizer can serve as a reference for developers and computer science enthusiasts who want to explore or compare various sorting algorithms. By using the tool to visualize how algorithms perform under different conditions, users can make informed decisions about which algorithm is best suited for specific tasks, taking into account factors such as performance, scalability, and the characteristics of the input data.

### 1.1.3 KEY FEATURES

The Sorting Algorithm Visualizer is equipped with several interactive features designed to enhance the user experience and make the learning process engaging and informative. These features include:

**Multiple Algorithm Support:** The visualizer supports a range of sorting algorithms, such as Bubble Sort, Insertion Sort, Selection Sort, and others. Users can select any algorithm to observe its operation on a set of data.

**Real-time Animation:** Users can see the algorithm in action as it processes the data. The algorithm's steps, such as comparisons and swaps, are highlighted through dynamic animations, giving a clear understanding of the sorting process.

**Input Customization:** The user can customize the input data by generating random arrays of different sizes, or by manually entering a set of values. This allows users to experiment with various input sizes and types to observe how the algorithm performs under different conditions.

**Speed Control:** The visualizer allows users to control the speed of the animation. This is especially useful for learners who want to slow down the

process to follow each step carefully or speed it up to observe the algorithm's performance on larger datasets.

**Performance Metrics:** The visualizer displays key performance metrics such as the number of comparisons, swaps, and execution time for each algorithm. This data helps users assess the efficiency of each algorithm and understand how different algorithms perform in different scenarios.

The Sorting Algorithm Visualizer offers significant educational value by transforming the learning experience. Instead of relying solely on static diagrams or written explanations, users can interact with the algorithms and observe how their actions impact the data in real time. This hands-on approach improves retention and comprehension.

The visualizer is designed not only for students learning sorting algorithms but also for educators who can use it as a teaching aid in classrooms. By demonstrating sorting algorithms interactively, educators can facilitate a more engaging and effective learning environment. The visualizer also supports self-paced learning, where users can experiment and explore algorithms at their own convenience.

## 1.2 PROBLEM STATEMENT

Understanding the internal workings of sorting algorithms can be challenging for students and beginners in computer science, especially when relying solely on traditional learning resources such as textbooks or static diagrams. These methods often fail to effectively communicate the step-by-step logic and dynamic behaviour of sorting processes, making it difficult for learners to grasp key concepts such as comparisons, swaps, and algorithm efficiency.

To address this issue, there is a need for an interactive, visually engaging tool that demonstrates how various sorting algorithms function in real time. Such a tool should not only animate the sorting process but also allow users to experiment

with different input sizes, control the speed of visualization, and choose among multiple algorithms to compare their behaviour.

The goal of this project is to develop a web-based Sorting Algorithm Visualizer using modern web technologies (HTML, CSS, SCSS, ReactJS, Material UI, and GSAP) that can effectively bridge the gap between theory and practice by providing a hands-on, intuitive learning experience.

### 1.3 OBJECTIVES

The primary objective of the Sorting Algorithm Visualizer project is to create an interactive educational tool that helps users, particularly students and beginner developers, understand the fundamental concepts behind sorting algorithms. The visualizer aims to bridge the gap between theoretical learning and practical understanding by offering a hands-on, visual approach to algorithm education.

The key objectives of the project are as follows:

**Enhance Learning Through Visualization:** Enable users to visually observe how sorting algorithms operate on data step by step.

**Promote User Interaction:** Provide users with the ability to control input array size, adjust sorting speed, and choose among multiple sorting algorithms to better understand their differences.

**Offer a Responsive and Accessible Platform:** Ensure the visualizer is accessible across various devices and platforms through modern web technologies, requiring no additional installations.

**Support Self-paced Learning:** Empower users to explore sorting algorithms at their own pace, fostering deeper understanding through experimentation and interaction.

By achieving these objectives, the Sorting Algorithm Visualizer aims to become a valuable tool for computer science education and a practical reference for understanding sorting logic in real time.



## 1.4 ORGANIZATION OF THE THESIS

The thesis is organized into five chapters, each addressing different aspects of the Sorting Algorithm Visualizer project:

**Chapter 1: Introduction** This chapter provides a general introduction to the project, including the problem statement, objectives, and the organization of the report.

**Chapter 2: Literature Survey** This chapter reviews relevant literature, theories, and prior research related to sorting algorithms and visualization tools. It includes a summary and analysis of key findings from existing sources and identifies gaps or areas for further investigation.

**Chapter 3: Methodology** This chapter discusses the proposed system, including a general description of the Sorting Algorithm Visualizer with a block diagram. It outlines the system requirements (hardware and software) and provides general information about techniques or models used in the project.

**Chapter 4: Implementation** This chapter describes the design modules, including the original implementation of the Sorting Algorithm Visualizer, tools used, and dataset. It also presents the experiments and results, including performance measures, snapshots, charts, tables, and graphs.

**Chapter 5: Conclusion** This chapter summarizes the key findings and their significance. It discusses how the results align with the research objectives and suggests future work for further improvement.

**References** This section lists all the references cited throughout the thesis.

## **CHAPTER 2**

### **LITERATURE SURVEY**

**[1] Algorithm Visualization: A Report on the State of the Field**, Clifford A. Shaffer, Matthew Cooper, Stephen H. Edwards, (2010)

This report delves into the key challenges faced by researchers in the domain of algorithm visualization. It traces the evolution of algorithm visualization over time, highlighting significant milestones and developments. The authors emphasize the importance of visual tools in enhancing algorithm education, suggesting that these tools can bridge the gap between theoretical concepts and practical understanding

The study identifies several obstacles that have hindered progress in the field, such as the lack of standardized evaluation methods and the difficulty in creating universally effective visualizations. Despite these challenges, the report showcases various successful implementations and their impact on learning outcomes. The authors argue that addressing these issues is crucial for advancing the field and improving educational practices.

In conclusion, the report underscores the potential of algorithm visualization to transform computer science education. By providing clear and interactive representations of complex algorithms, visual tools can significantly enhance comprehension and retention. The authors call for continued research and development to overcome existing challenges and fully realize the benefits of algorithm visualization.

**[2] Effective Features of Algorithm Visualizations**, Purvi Saraiya, (2002)

This paper evaluates the interactive features of algorithm visualizations, focusing on elements that enhance user engagement and understanding. The author emphasizes the value of integrating pseudocode into visualizations, arguing that it helps users connect abstract concepts with concrete implementations. The study provides design recommendations for creating more effective visualizations.

The research highlights several key features that contribute to the effectiveness of algorithm visualizations, such as user interactivity, real-time feedback, and the ability to manipulate data inputs. These features are shown to improve comprehension and retention by allowing users to explore algorithms dynamically. The author also discusses the importance of visual clarity and intuitive design in making visualizations accessible to a wide audience.

In conclusion, the paper suggests that incorporating interactive elements and pseudocode into algorithm visualizations can significantly enhance their educational value. The author calls for further research to refine these features and develop best practices for designing effective visualizations. By doing so, educators can create tools that better support student learning and engagement.

**[3] A Literature Review on Algorithm Visualizers**, Sweeta Bansal, Karan Kohli, Krishna Kumar, Kush Gupta, (2022)

This literature review surveys a broad range of existing algorithm visualization tools, examining their strengths and weaknesses. The authors discuss how these tools enhance user engagement and clarity, making complex algorithms more accessible to learners. The review identifies several limitations and gaps in current visualization designs, suggesting areas for future research.

The study highlights the importance of user interactivity and visual clarity in effective algorithm visualizers. Tools that allow users to manipulate data inputs and observe real-time feedback are shown to improve comprehension and retention. However, the authors note that many existing visualizers lack these features, limiting their educational impact. The review also points out the need for standardized evaluation methods to assess the effectiveness of visualizations.

In conclusion, the authors call for continued research to address the limitations identified in the review. By developing more interactive and visually clear tools, educators can enhance the learning experience for students. The review suggests that future studies should focus on refining visualization designs and establishing best practices for their implementation in educational settings.

**[4] Review of Algorithm Visualization Methodologies**, Jay Talekar, Jugal Suthar, Sanket Joshi, Prof. Jignesh Patel, (2022)

This review compares various innovative, interactive visualization approaches, emphasizing the advantages of web-based solutions. The authors highlight the central role of user interactivity in enhancing learning, suggesting that interactive visualizations can significantly improve comprehension and retention. The review provides a comprehensive overview of different methodologies and their impact on algorithm education.

The study discusses several successful implementations of web-based visualizations, showcasing their effectiveness in educational settings. These tools allow users to manipulate data inputs, observe real-time feedback, and explore algorithms dynamically. The authors argue that web-based solutions offer greater accessibility and flexibility compared to traditional methods, making them ideal for modern educational environments.

In conclusion, the review underscores the potential of interactive visualizations to transform algorithm education. By providing clear and dynamic representations of complex algorithms, these tools can enhance student engagement and understanding. The authors call for further research to refine web-based methodologies and develop best practices for their implementation in educational settings.

**[5] Designing Educationally Effective Algorithm Visualizations**, N. H. Narayanan, M. Hegarty, S. R. Hansen, (1998)

This paper introduces the HalVis framework for context-rich, interactive algorithm animations. The authors demonstrate how synchronized pseudocode and visual feedback boost comprehension, validating the importance of interactivity in transforming traditional algorithm teaching. The study highlights the educational benefits of using interactive visualizations in algorithm education.

The research showcases several successful implementations of the HalVis framework, emphasizing its effectiveness in enhancing learning outcomes. By

providing real-time feedback and allowing users to manipulate data inputs, HalVis helps students connect abstract concepts with concrete implementations. The authors argue that interactive visualizations can significantly improve comprehension and retention compared to traditional teaching methods.

In conclusion, the paper suggests that the HalVis framework offers a valuable tool for algorithm education. By incorporating interactive elements and synchronized pseudocode, educators can create visualizations that better support student learning and engagement. The authors call for further research to refine the framework and develop best practices for its implementation in educational settings.

**[6] On the Role of Animated Analogies in Algorithm Visualizations, S. R. Hansen, N. H. Narayanan, (2000)**

An extended version of the HalVis framework, this paper presents interactivity features and learning modules of HalVis. The authors highlight the central role of user interactivity in learning enhancement, discussing how animated analogies can improve the understanding of complex algorithms. The study provides evidence of the framework's effectiveness in educational settings.

The research demonstrates how animated analogies can make complex algorithms more accessible to learners. By providing real-time feedback and allowing users to manipulate data inputs, HalVis helps students connect abstract concepts with concrete implementations. The authors argue that interactive visualizations can significantly improve comprehension and retention compared to traditional teaching methods.

In conclusion, the paper suggests that the HalVis framework offers a valuable tool for algorithm education. By incorporating interactive elements and animated analogies, educators can create visualizations that better support student learning and engagement. The authors call for further research to refine the framework and develop best practices for its implementation in educational settings.

**[7] On the Role of Animated Analogies in Algorithm Visualizations**, C. Kann, R. W. Lindeman, R. Heller, (1997)

This study reports significant gains in student performance and engagement through the use of animated analogies in algorithm visualizations. The authors highlight design considerations for effective in-class and self-paced use, providing empirical evidence of the benefits of using animated analogies to teach algorithms. The study suggests that these visualizations can enhance comprehension and retention.

The research showcases several successful implementations of animated analogies, emphasizing their effectiveness in educational settings. By providing real-time feedback and allowing users to manipulate data inputs, these visualizations help students connect abstract concepts with concrete implementations. The authors argue that interactive visualizations can significantly improve comprehension and retention compared to traditional teaching methods.

In conclusion, the paper suggests that animated analogies offer a valuable tool for algorithm education. By incorporating interactive elements and real-time feedback, educators can create visualizations that better support student learning and engagement. The authors call for further research to refine these visualizations and develop best practices for their implementation in educational settings.

**[8] A Web-Based Algorithm Animation System for an Electronic Classroom**, Marc H. Brown, Marc A. Najork, (2000)

This paper explores web-based interactive textbooks integrating algorithm animations. The authors discuss synchronous collaboration features for group learning and emphasize user control over animation speed and data inputs. The study highlights the potential of web-based systems to enhance algorithm education through interactive and collaborative learning environments.

The research showcases several successful implementations of web-based algorithm animations, emphasizing their effectiveness in educational settings. By providing real-time feedback and allowing users to manipulate data inputs, these

tools help students connect abstract concepts with concrete implementations. The authors argue that web-based solutions offer greater accessibility and flexibility compared to traditional methods, making them ideal for modern educational environments.

In conclusion, the paper suggests that web-based algorithm animations offer a valuable tool for algorithm education. By incorporating interactive elements and synchronous collaboration features, educators can create visualizations that better support student learning and engagement. The authors call for further research to refine these tools and develop best practices for their implementation in educational settings.

**[9] Testing Effectiveness of Algorithm Visualization**, J. S. Gurka, W. Citrin, (1998)

This paper presents an empirical framework for evaluating learning gains from visual tools. The authors address the challenge of quantifying "effectiveness" in controlled experiments and propose guidelines for designing user studies and analyzing engagement metrics. The study provides insights into the factors that contribute to the effectiveness of algorithm visualizations.

The research highlights several key factors that influence the effectiveness of algorithm visualizations, such as user interactivity, visual clarity, and the ability to manipulate data inputs. By providing real-time feedback and allowing users to explore algorithms dynamically, these visualizations can significantly enhance comprehension and retention. The authors suggest that standardized evaluation methods are needed to assess the impact of visual tools on learning outcomes.

In conclusion, the paper suggests that an empirical framework for evaluating algorithm visualizations can help educators design more effective tools. By understanding the factors that contribute to their effectiveness, educators can create visualizations that better support student learning and engagement. The authors call for further research to refine evaluation methods and develop best practices for their implementation in educational settings.

**[10] Smooth Animation of Algorithms in a Declarative Framework,** C. Demetrescu, I. Finocchi, (1999)

This paper introduces a declarative model for generating algorithm animations, focusing on rendering smooth transitions to maintain user context. The authors offer a scalable framework suitable for both simple and complex algorithm visualizers, highlighting the benefits of using a declarative approach. The study emphasizes the importance of smooth animations in helping users follow the algorithm's flow without losing context.

The research demonstrates how the declarative model can be applied to various algorithms, providing examples of both simple and complex visualizations. By maintaining smooth transitions, the model ensures that users can easily track changes and understand the algorithm's behavior. The authors argue that this approach enhances the educational value of visualizations by making them more intuitive and accessible.

In conclusion, the paper suggests that a declarative framework for algorithm animations offers significant advantages in terms of scalability and user experience. By focusing on smooth transitions and maintaining user context, educators can create visualizations that better support student learning and engagement. The authors call for further research to refine the framework and explore its applications in different educational settings.



# CHAPTER 3

## METHODOLOGY

### 3.1 BLOCK DIAGRAM OF THE PROPOSED SYSTEM

Sorting Algorithm Visualizer Flow Chart

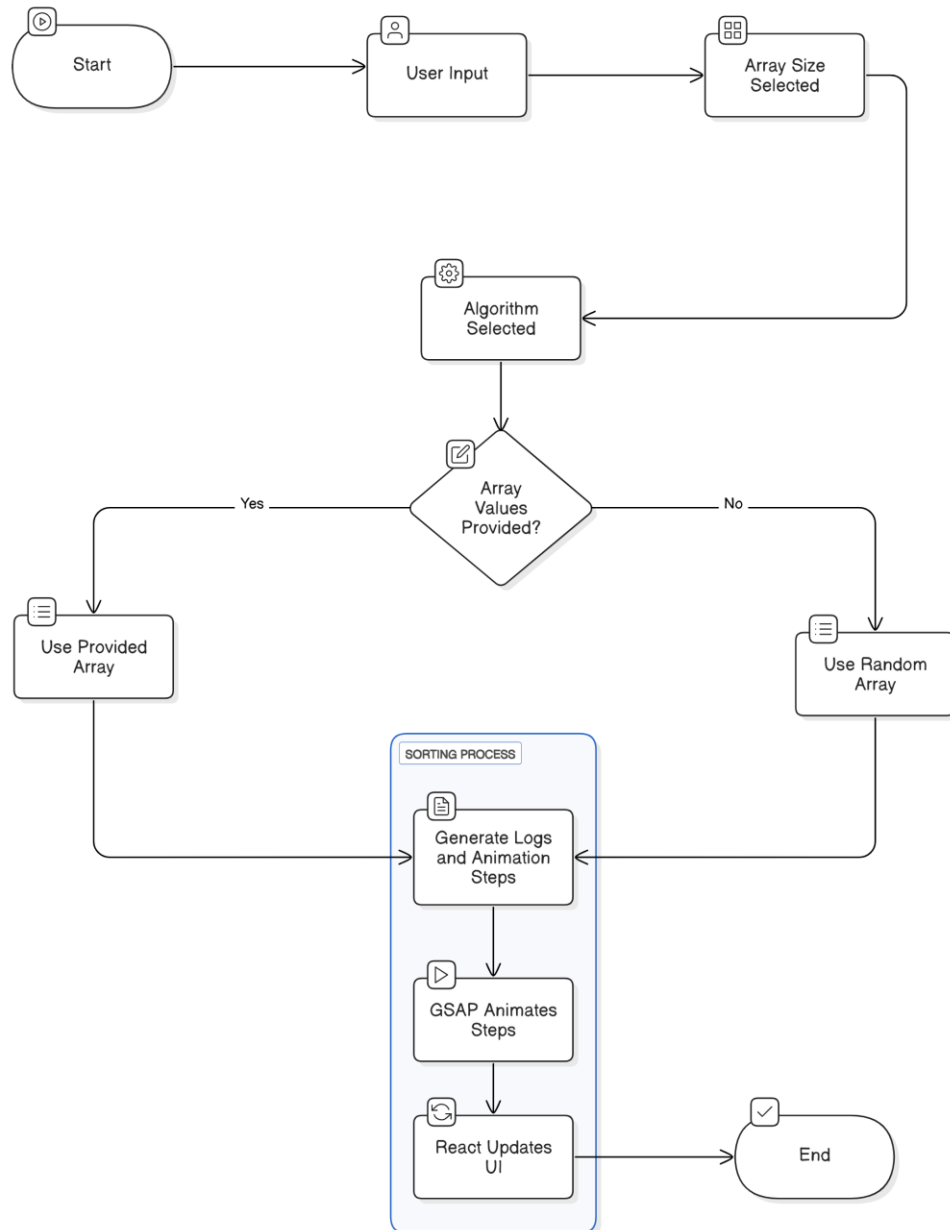


Fig. 3.1 Block Diagram of the Proposed System

Figure 3.1 shows The block diagram for the Sorting Algorithm Visualizer project outlines the flow of data and processes involved in visualizing sorting algorithms. The process begins with user input, where the user provides the array size, selects the sorting algorithm, and optionally inputs array values. The system then checks if specific array values are provided; if the user has provided array values, the system uses the provided array; otherwise, it generates a random array.

Once the array is determined, the system generates logs and animation steps based on the selected sorting algorithm and array values. These logs detail the steps taken during the sorting process, providing a comprehensive record of the algorithm's operations. The GreenSock Animation Platform (GSAP) then animates these steps, creating visual representations of the algorithm's operations. GSAP's powerful animation capabilities ensure that the sorting steps are displayed smoothly and dynamically, enhancing the user's understanding of the algorithm's behavior.

React plays a crucial role in updating the user interface reactively based on the animations generated by GSAP. This ensures that the visualizations are displayed in real-time and are responsive to user interactions. React's efficient rendering capabilities allow for seamless updates to the UI, providing an interactive and engaging experience for the user. The process concludes with the completion of the sorting visualization, marking the end of the sorting algorithm's execution and the visualization process.

This block diagram visually represents the sequence of operations in the Sorting Algorithm Visualizer project, from user input to the final animated visualization. It highlights the integration of user inputs, algorithm selection, log generation, animation, and UI updates, providing a clear and comprehensive overview of the project's workflow.

## 3.2 SYSTEM ARCHITECTURE

The Sorting Algorithm Visualizer is designed using the Model-View-Controller (MVC) architecture to ensure a clear separation of concerns, making the application more modular, maintainable, and scalable. The MVC architecture divides the application into three interconnected components: Model, View, and Controller.

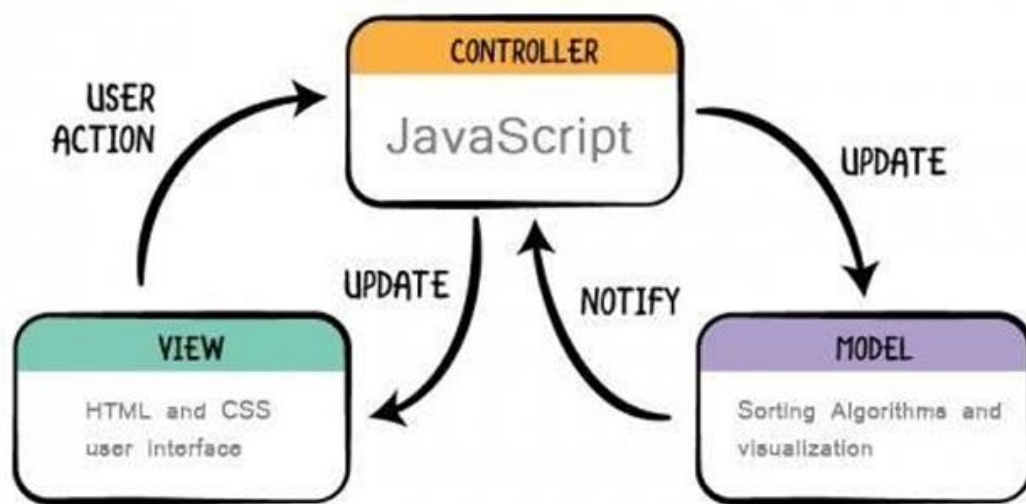


Fig 3.2 MVC Architecture

### 3.2.1 Model

The Model represents the data and the business logic of the application. In the context of the Sorting Algorithm Visualizer, the Model includes the sorting data, which is the array of elements to be sorted, and the visualization data, which encompasses the steps and states of the array during the sorting process. The Model is responsible for storing the array and its states, implementing the sorting algorithms (Bubble Sort, Selection Sort, Insertion Sort), and notifying the Controller of any changes in the data.

### 3.2.2 View

The View is responsible for the presentation layer of the application. It defines how the data is displayed to the user. In this project, the View includes HTML for the

structure of the user interface, CSS for the styling to make it visually appealing, and user interface components such as buttons, sliders, and display areas for the array and pseudocode. The View is responsible for rendering the array and its states visually, displaying the pseudocode and highlighting the current step, and providing a responsive and intuitive user interface.

### **3.2.3 Controller**

The Controller acts as an intermediary between the Model and the View. It handles user input and updates the Model and View accordingly. In this project, the Controller is implemented using JavaScript and includes event handlers to handle user actions such as starting, pausing, and resetting the sorting process. The Controller updates the visual representation of the array and pseudocode based on the Model's data and modifies the array and triggers sorting algorithms based on user input. The Controller ensures synchronization between the Model and the View by capturing user actions from the View, updating the Model with new data or commands, and updating the View with the latest data from the Model.

This MVC architecture ensures that the Sorting Algorithm Visualizer is well-organized, with each component handling its specific responsibilities, leading to a more efficient and maintainable application.

### 3.3 SYSTEM REQUIREMENTS

System requirements are the minimum specifications needed for hardware, software, and other resources to run a computer system or software application effectively.

#### 3.3.1 MINIMUM HARDWARE REQUIREMENTS

**Processor:** Dual-core CPU (e.g., Intel Core i3 or AMD Ryzen 3)

**RAM:** 2GB

**Storage:** 100MB of free disk space

**Graphics:** Integrated graphics (e.g., Intel HD Graphics or AMD Radeon Vega)

**Display:** Minimum resolution of 1280x800

**Internet Connection:** Stable internet connection

#### 3.3.2 RECOMMENDED HARDWARE REQUIREMENTS

**Processor:** Quad-core CPU (e.g., Intel Core i5 or AMD Ryzen 5)

**RAM:** 4GB or more

**Storage:** 500MB of free disk space

**Graphics:** Integrated graphics ((e.g., Intel UHD Graphics or AMD Radeon Vega)

**Display:** Full HD resolution (1920x1080)

**Internet Connection:** Stable internet connection

### 3.3.3 SOFTWARE REQUIREMENTS

**Operating System:** The application is compatible with major operating systems, including Windows, macOS, and Linux. Users can access the visualizer on any of these platforms via a modern web browser.

**Web Browser:** A modern web browser is required to run the Sorting Algorithm Visualizer:

- Google Chrome, Mozilla Firefox, Safari, or Microsoft Edge.
- Ensure the browser is updated to the latest version for best performance and compatibility with interactive elements and animations.

### 3.4 TOOLS AND TECHNOLOGIES

The following tools and technologies are used in this project:

- **ReactJS:** ReactJS is a widely-used JavaScript library for building user interfaces, especially for single-page applications (SPAs). It allows developers to create reusable UI components that can dynamically update as the application's state changes. In this project, ReactJS is used to create the interactive visualizations of the sorting algorithms, ensuring real-time updates and smooth transitions as the algorithms sort the data step by step. React's virtual DOM also contributes to improving performance by efficiently updating only the changed parts of the user interface.
- **Material UI:** Material UI is a React-based component library that follows Google's Material Design guidelines. It offers a wide range of pre-designed, customizable components such as buttons, sliders, dialog boxes, and more, that help developers quickly build aesthetically pleasing and functional user interfaces. In this project, Material UI is used to create a polished, user-friendly layout for the Sorting Algorithm Visualizer.

- **GSAP (GreenSock Animation Platform):** GSAP is an advanced JavaScript library that excels in creating high-performance animations. It is known for its smooth transitions and ease of use, even for complex animations. For this project, GSAP is used to animate the sorting algorithms, creating engaging, real-time visual effects as elements are rearranged during sorting. GSAP ensures that the animations are fluid and run without any lag, even with large datasets.
- **HTML & CSS/SCSS:** HTML is the foundational language used to structure the content of the Sorting Algorithm Visualizer, ensuring all elements are properly laid out in the browser. CSS, along with SCSS (a CSS preprocessor), is used to style the application. SCSS makes it easier to manage complex styles with features like variables, nested rules, and mixins, which help create a scalable and maintainable design. The visual design ensures the user interface is clean, modern, and intuitive.
- **JavaScript:** JavaScript is the primary language for the logic behind the Sorting Algorithm Visualizer. The project uses JavaScript to implement the core functionality of the sorting algorithms, such as Bubble Sort, Selection Sort, and Insertion Sort. These algorithms are executed step-by-step, and the JavaScript code updates the visual representation in real-time, allowing users to observe the sorting process visually. JavaScript also handles user interactions, such as starting, pausing, and resetting the animations.
- **Vite:** Vite is a modern, fast build tool and development server that significantly improves the developer experience. Unlike older tools such as Webpack, Vite leverages native ES modules in the browser for lightning-fast builds and hot module reloading (HMR). This means developers can make changes to the project and immediately see those changes in the browser without waiting for a full rebuild. Vite also optimizes the production build process, ensuring that the final application is fast and lightweight, which is crucial for performance when handling visualizations and animations in a web application.

- **Git & GitHub:** Git is a distributed version control system that allows multiple developers to work on the same project without overwriting each other's work. It tracks changes to the codebase and makes it easier to collaborate, roll back to previous versions, and manage project history. GitHub is an online platform that hosts Git repositories, enabling easy collaboration and version control for the project. It also allows developers to create branches, open pull requests, and track issues, making it an essential tool for managing the development and deployment of the Sorting Algorithm Visualizer.



## **CHAPTER 4**

### **IMPLEMENTATION**

The implementation of the Sorting Algorithm Visualizer project aims to provide an interactive and engaging way for users to understand and explore how different sorting algorithm's function. This web-based tool, built with modern technologies like ReactJS, GSAP, and Material UI, allows users to visualize the sorting process step by step, making it easier to comprehend the inner workings of sorting algorithms. The visualizer supports three popular sorting algorithms: Bubble Sort, Selection Sort, and Insertion Sort, which are demonstrated in real-time as users control the speed and execution steps of the sorting process. By offering this dynamic and visual approach, the project seeks to enhance the learning experience and make algorithm analysis more accessible, especially for beginners and students.

The tool provides various interactive features, such as the ability to customize inputs, control the speed of execution, and navigate through the sorting process step by step. This interactivity encourages users to experiment with different types of data, including random, or user-provided datasets. These features allow learners to see the impact of different inputs on the behavior of the sorting algorithms and understand how the performance of each algorithm varies. By combining visualization with real-time explanations and pseudo-code highlighting, users can directly correlate the visual actions with the algorithmic steps, deepening their understanding of how sorting algorithms operate under different conditions.

#### **4.1 MODULAR DESIGN**

The Sorting Algorithm Visualizer project employs a modular design to ensure flexibility, scalability, and ease of maintenance. The modular design is divided into several key components, each responsible for specific functionalities within the application. Here are the main modules:

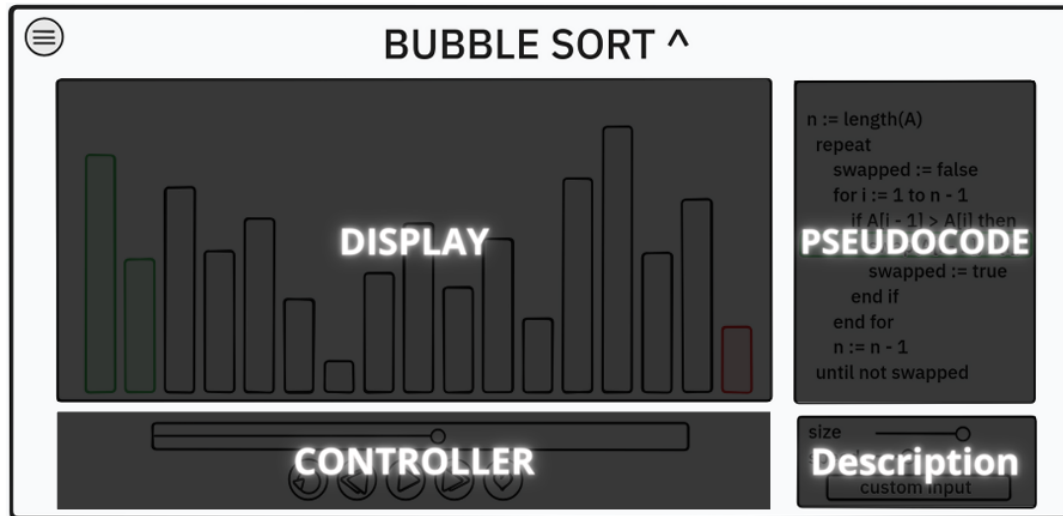


Fig 4.1 Modular Design

**Display Module:** This module is responsible for rendering the visual representation of the array being sorted. It uses bar graphs to display the array elements, with varying heights representing different values. The display is updated in real-time to reflect the changes made by the sorting algorithm.

**Pseudocode Module:** The Pseudocode Module is responsible for displaying and highlighting the pseudocode of the selected sorting algorithm. As the sorting process progresses, the corresponding lines of pseudocode are highlighted in real-time, helping users understand which part of the algorithm is being executed. This dynamic highlighting, synchronized with the animation steps, enhances the educational value of the visualizer by providing a clear and interactive representation of the algorithm's logic.

**Controller Module:** This module provides control buttons and settings for the sorting visualization. Users can start, pause, and reset the sorting process, as well as adjust the speed of the animations. The controller module ensures that users have interactive control over the visualization.

**Description Module:** This module provides detailed descriptions and explanations of the sorting process. It helps users understand the steps taken by the algorithm and the

changes occurring in the array. The descriptions are updated dynamically to match the current state of the visualization.

The modular design of the Sorting Algorithm Visualizer project allows each component to function independently while interacting seamlessly with other modules. This approach ensures that the application is easy to maintain and extend, as new sorting algorithms or features can be added without disrupting the existing functionality. The use of React and GSAP further enhances the modular design by providing efficient rendering and animation capabilities, making the visualizations smooth and engaging for users.

## 4.2 DATA INPUT HANDLING

The Data Input Handling Module allows users to input custom array values as comma-separated numbers. This module includes a user-friendly interface with a text box labeled "Custom Input," where users can enter their desired array values. Upon submission, the module parses the input string, validates the numbers, and converts them into an array format for the sorting algorithm. This feature provides flexibility and enhances user engagement by allowing personalized data input for visualization. allow users to analyze the effectiveness of each algorithm in different contexts.

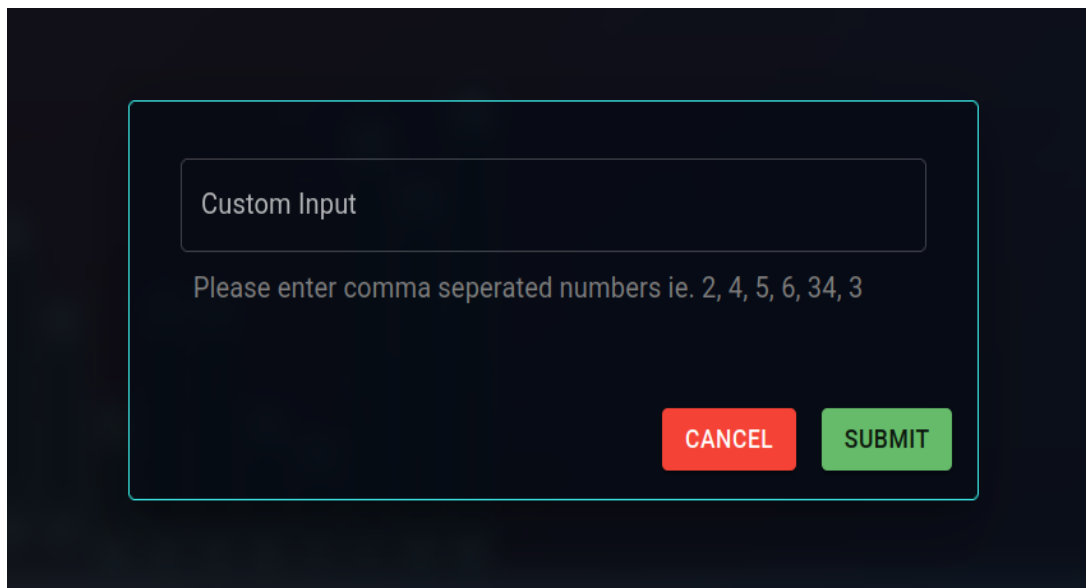
The image shows a dark-themed user interface for a 'Custom Input' module. It features a light blue rectangular box with rounded corners. Inside this box, at the top, is a text input field with the placeholder text 'Custom Input'. Below the input field, there is a line of instructional text: 'Please enter comma seperated numbers ie. 2, 4, 5, 6, 34, 3'. At the bottom right of the box, there are two buttons: a red button labeled 'CANCEL' and a green button labeled 'SUBMIT'.

Fig.4.2. Custom Input Module

### 4.3 VISUALIZATION ENGINE



Fig.4.3. GSAP

The core of the sorting algorithm visualizer lies in its advanced visualization engine, which translates the sorting process into animated, real-time visual representations. Each element in the data is represented by a vertical bar, with the height of the bar corresponding to the value of the element. As the algorithm proceeds, the bars are rearranged to show how the data is being sorted. The visualization is powered by GSAP (GreenSock Animation Platform), which ensures smooth and fast animations, even with larger datasets. The dynamic nature of the visualization helps users observe exactly how elements move, swap, or remain in place during the sorting process, making the algorithms easier to follow and understand.

Moreover, the visualization engine allows users to see both the data manipulation and the corresponding changes to the algorithm's internal state. Each step of the sorting process is displayed in real-time, so users can track how the algorithm progresses toward a fully sorted array. The smooth animations are synchronized with the algorithm's execution, providing users with a continuous flow

of information. This real-time feedback makes it easier to grasp abstract algorithm concepts such as comparisons, swaps, and data manipulation.

## 4.4 CONTROLS AND INTERACTIVITY

The sorting algorithm visualizer is designed with user interactivity in mind, providing various controls that allow users to customize their learning experience. The most prominent feature is the step-by-step navigation, which lets users manually control the execution of the sorting algorithm. This functionality mimics a media player interface, with "Next" and "Previous" buttons that allow users to move forward or backward through each step of the algorithm. This control is especially useful for users who wish to examine specific steps in greater detail or who need additional time to understand how the algorithm is processing the data.



Fig.4.4. Controls Module

In addition to step-by-step control, the visualizer includes a speed control slider, which allows users to adjust the speed of the sorting process. This feature enables users to slow down the sorting algorithm for a more thorough understanding of each operation or speed it up to observe the overall behavior of the algorithm. The ability to adjust the speed adds a layer of flexibility to the visualizer, catering to different learning paces. Slower speeds allow for careful observation of individual steps, while faster speeds allow users to quickly view how the algorithm handles larger datasets. By combining manual controls with adjustable speed, the visualizer gives users full control over their learning experience.

## 4.5 PSEUDO-CODE HIGHLIGHTING MODULE

A key educational feature of the sorting algorithm visualizer is the pseudo-code highlighting module. This feature enhances the user's understanding of the

algorithm by synchronizing the visual representation with the corresponding pseudo-code. As the sorting algorithm runs, the relevant line of pseudo-code is highlighted, allowing users to directly connect the visual actions they are observing with the underlying logic of the algorithm. This module is especially valuable for beginners, as it helps bridge the gap between visualizations and theoretical concepts, making it easier to understand the steps involved in sorting.

```
1 FUNCTION bubbleSort(arr):  
2   n ← length of arr  
3   FOR i FROM 0 TO n - 1:  
4     swapped ← false  
5     FOR j FROM 0 TO n - i - 1:  
6       IF arr[j].v > arr[j + 1].v:  
7         SWAP arr[j] WITH arr[j + 1]  
8         swapped ← true  
9     END FOR  
10    IF swapped = false:  
11      BREAK  
12  END FOR  
13 END FUNCTION
```

Fig.4.5. Pseudo Code Module

The synchronization between the visualized sorting process and the pseudo-code provides users with real-time feedback on what the algorithm is doing at each step. For example, when two elements are being compared or swapped, the corresponding lines in the pseudo-code are highlighted to indicate exactly what operation is taking place. This feature allows users to follow the algorithm's flow of execution while reinforcing the theoretical concepts behind each step. The real-time synchronization also encourages users to think critically about how algorithms work and to recognize patterns in the code, further enhancing their algorithmic knowledge.

#### 4.6 EXPLANATION PANEL

The explanation panel is another important feature that complements the visualizations and pseudo-code highlighting. It provides real-time, textual explanations of the algorithm's operations as they occur. For instance, when two elements are compared, the panel will display a message like "Comparing element 12 with 2," followed by an explanation if a swap is necessary. This textual feedback helps users understand the reasoning behind each action, such as why two elements are being swapped or what the algorithm is trying to achieve in each step. These explanations are essential for beginners who may not be familiar with the intricacies of sorting algorithms, as they provide context to the visual changes occurring on the screen.

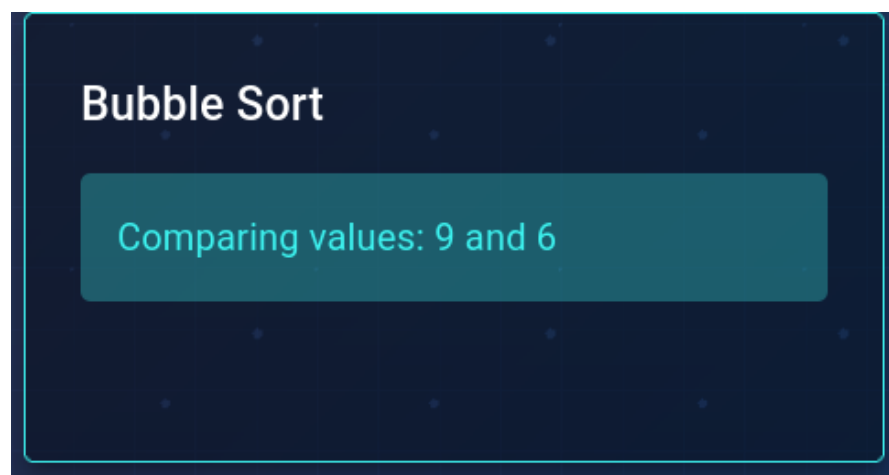


Fig.4.6. Explanation Module

The explanations are synchronized with the sorting process, updating in real-time as the algorithm progresses through its steps. This ensures that users always have a clear understanding of what is happening at each point in the execution. By combining visual feedback with detailed, step-by-step explanations, the panel enhances the educational value of the visualizer. It not only helps users grasp the mechanics of the sorting algorithms but also enables them to learn the underlying concepts, such as comparisons, swaps, and order of operations, in a structured and accessible way.

## 4.7 SORTING ALGORITHMS IMPLEMENTED

The sorting algorithm visualizer implements three fundamental sorting algorithms: Bubble Sort, Selection Sort, and Insertion Sort. These algorithms are commonly taught in introductory computer science courses, and each has its own unique approach to sorting. Bubble Sort is a simple comparison-based algorithm that repeatedly compares adjacent elements and swaps them if they are in the wrong order. This process continues until no more swaps are needed, indicating that the array is sorted. Although intuitive, Bubble Sort is inefficient for large datasets due to its high time complexity.

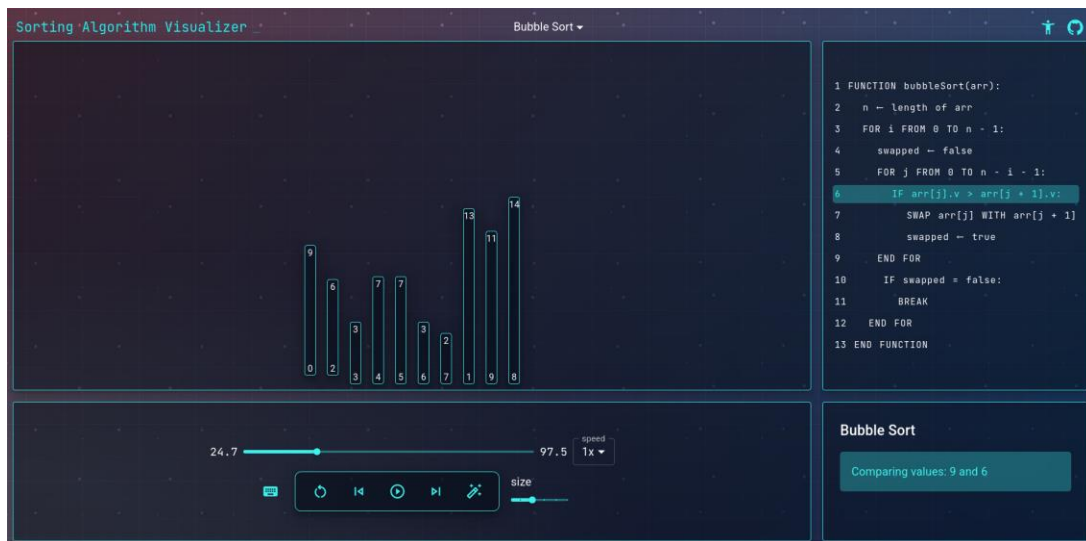


Fig.4.7. Sorting Visualizer



Selection Sort, on the other hand, works by selecting the smallest (or largest) element from the unsorted portion of the array and swapping it with the first unsorted element. This process continues until all elements are sorted. Like Bubble Sort, Selection Sort is inefficient for large datasets, but it provides a clearer understanding of the process of selection and swapping. Insertion Sort works similarly to Selection Sort but differs in that it inserts elements into their correct position within the sorted portion of the array, rather than swapping them. The visualizer allows users to explore the strengths and weaknesses of each algorithm, comparing their performance on different types of data, and learning how to choose the best algorithm for a given task.

## **4.8 USER EXPERIENCE ENHANCEMENTS**

The overall user experience is a key focus of the Sorting Algorithm Visualizer, with the goal of providing an intuitive, engaging, and educational interface. The visualizer's user interface is designed to be simple and easy to navigate, ensuring that users can focus on learning without distractions. All essential controls, such as speed adjustments, input selection, and step navigation, are clearly labeled and easily accessible. The clean layout is complemented by smooth animations, which ensure that users can clearly see the sorting process without any visual distractions or performance issues. This attention to detail in design is crucial for maintaining a positive and effective learning environment.

Additionally, the visualizer is built to be responsive, meaning it can be used on various devices, from desktop computers to mobile phones, without any loss of functionality. This flexibility ensures that users can access the visualizer from different devices and continue their learning at any time. The tool also includes user-friendly features like tooltips and guides that help users understand how to use the controls effectively. These enhancements contribute to a seamless and enjoyable learning experience, making the visualizer not only a powerful educational tool but also an accessible and user-friendly platform for exploring sorting algorithms.

## CHAPTER 5

### CONCLUSION

The implementation of a Sorting Algorithm Visualizer using interactive technologies such as ReactJS, Material UI, and GSAP provides an intuitive platform for users to explore and understand the behavior of sorting algorithms in real time. By integrating visualization techniques with educational features like pseudo-code highlighting, explanation panels, and speed control, the application offers a comprehensive learning experience tailored to different user preferences and paces. The inclusion of step-by-step execution, custom input handling, and animated transitions transforms complex algorithmic logic into accessible visual feedback that supports active learning and experimentation.

Furthermore, the visualization of core algorithms such as Bubble Sort, Selection Sort, and Insertion Sort allows for effective comparison of their characteristics and behaviors on various datasets. Through real-time manipulation and user interactivity, the tool addresses common challenges learners face in grasping algorithmic concepts, especially when it comes to understanding control flow, data comparisons, and element swapping. The responsive and user-friendly nature of the interface further enhances its educational value by ensuring accessibility across devices and platforms.

In conclusion, the integration of sorting algorithms with interactive visualization and intuitive UI design significantly aids in demystifying the internal workings of algorithms. This approach not only improves conceptual understanding but also serves as a valuable educational tool for students, educators, and enthusiasts seeking to strengthen their foundations in data structures and algorithms through hands-on learning.

## 5.1 FUTURE SCOPE

In future developments, the functionality of the Sorting Algorithm Visualizer can be significantly expanded to elevate both its educational and interactive value. One potential enhancement includes the integration of additional sorting algorithms such as Merge Sort, Quick Sort, Heap Sort, and Radix Sort, thereby offering a broader comparative perspective on sorting techniques. These advanced algorithms, when visualized, can provide learners with deeper insights into divide-and-conquer strategies, recursion, and time-space complexities. Furthermore, enabling side-by-side comparison of algorithms on the same input can highlight efficiency differences more clearly, especially when combined with performance metrics like the number of comparisons, swaps, and execution time.

Another promising area for future improvement lies in interactive elements like gamified challenges, quizzes, or algorithmic puzzles can be incorporated to make the learning experience more engaging and competitive. Moreover, providing support for mobile and tablet devices, along with accessibility features like keyboard navigation and screen reader compatibility, will help reach a wider audience. Collectively, these future enhancements can transform the visualizer from a static educational tool into a dynamic, personalized learning ecosystem that supports a wide range of learners.

## REFERENCES

1. Narayanan, N. H., Hegarty, M., & Hansen, S. R. Designing Educationally Effective Algorithm Visualizations. 1998.
2. Shaffer, C. A., Cooper, M., & Edwards, S. H. Algorithm Visualization: A Report on the State of the Field. August 2010.
3. GreenSock, GSAP Documentation, 2025.
4. Meta Platforms, Inc., React Documentation, 2025.
5. Material UI, Material UI Documentation, 2025.
6. Framer, Framer Motion Documentation, 2025.
7. The Odin Project, The Odin Project Documentation, 2025.
8. Marijn Haverbeke, Eloquent JavaScript: A Modern Introduction to Programming, 3rd Edition, 2018. Available at: Eloquent JavaScript
9. Douglas Crockford, JavaScript: The Good Parts, O'Reilly Media, 2008.