IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

# Algorithm Visualizer With Pygame And Tkinter

[1]Dr. Deepa V B, [2]Suchin N H, [3]Shashank H, [4]Shreyas N Gatti, [5]Chandan B K
[1]Assistant Professor at JNNCE, [2,3,4,5]Student at JNNCE
[1]Information Science and Engineering,
[1]Jawaharlal Nehru New College Of Engineering, Shivamogga, India

***Abstract:*** The Algorithm Visualizer is an interactive educational tool developed to simplify the understanding of algorithms by providing dynamic, real-time visualizations. Developed using Python libraries Tkinter for GUI and Pygame for animations, the tool supports visualization of sorting, searching, and algorithms. It bridges the gap between theoretical learning and practical implementation, enabling users to observe each step of an algorithm in action. Features such as customization of input, control over animation speed, and comparative analysis of algorithm efficiency make this tool particularly valuable for students, educators, and programming enthusiasts. This paper discusses the architecture, features, and educational impact of the Algorithm Visualizer.

***Index Terms* -** Algorithm Visualization, Tkinter, Pygame, Educational Tool, Real Time Interaction.

## I. Introduction

Algorithms are the foundation of computational systems, yet they often pose a challenge to learners due to their abstract nature. Traditional teaching methods, such as static diagrams and textual explanations, fail to convey the dynamic behaviour of algorithms effectively. This project aims to overcome these challenges by providing a visual learning tool that allows users to interact with algorithms dynamically.

The Algorithm Visualizer supports multiple algorithms, including sorting (Bubble Sort, Merge Sort, Quicksort), searching (Linear Search, Binary Search), and graph traversal (Dijkstra’s, BFS, DFS). By offering step-by-step animations and interactive controls, the tool enhances comprehension, promotes critical thinking, and bridges the gap between theory and practice.

## II LITERATURE SURVEY

The papers reviewed highlight the transformative role of algorithm visualization in modern education, making complex computational concepts more accessible and engaging. [3] Tools like AlgoViz leverage interactive, web-based visualizations to simplify algorithms such as [1] Bubble Sort, Quick Sort, and [2] Dijkstra’s. These tools provide features like step-by-step animations, real-time control, and cross-platform compatibility, enabling learners to comprehend time and space complexities more effectively. Modern platforms also tackle the limitations of passive learning by promoting active engagement through [4] interactive inputs, dynamic feedback, and customizable dataset. Users find these tools that particularly beneficial in understanding the impact of dataset variations, obstacles, and constraints on algorithm efficiency. The integration of adjustable parameters, real-time performance metrics, and comparative analysis fosters critical thinking and bridges the gap between theoretical concepts and practical applications. By combining accessibility, adaptability, and dynamic feedback, algorithm visualizers enhance user engagement, comprehension, and retention. Future advancements, such as incorporating AR/VR and expanding algorithm libraries, hold the potential to further revolutionize computer science education, making it more interactive, immersive, and impactful. Algorithm visualization stands as a powerful tool in bridging theoretical knowledge with real-world applications, aligning with the broader educational goal of fostering computational thinking and problem-solving skills.

## III SYSTEM ARCHITECTURE

The system architecture of the Algorithm Visualizer is designed to deliver an interactive and engaging learning experience by integrating multiple components seamlessly. It combines functionality, performance, and interactivity to help users explore and comprehend algorithms effectively. The frontend, developed using Python's Tkinter library, serves as the graphical user interface where users can input datasets, select algorithms like Bubble Sort or Dijkstra's, and adjust parameters such as visualization speed. Intuitive controls like buttons and sliders make the interface simple and accessible for users of all levels. The backend, powered by Python's Pygame library, handles the execution of algorithms and generates dynamic visualizations. It depicts key operations such as comparisons, swaps, and node traversals, ensuring visually accurate and clear representations of computational processes.

The data layer allows users to upload custom datasets or choose from predefined examples, ensuring smooth integration with the visualization engine while showcasing how algorithm performance varies across different data types and sizes. The visualization engine lies at the core of the architecture, transforming computational logic into real-time visual animations. It highlights critical steps and dynamically adapts to user inputs, providing a cohesive and immersive learning experience. Together, these components ensure the system is scalable, adaptable, and effective in bridging the gap between theoretical learning and practical understanding, making algorithms more accessible and interactive.

## IV ARCHITECTURAL DESIGN

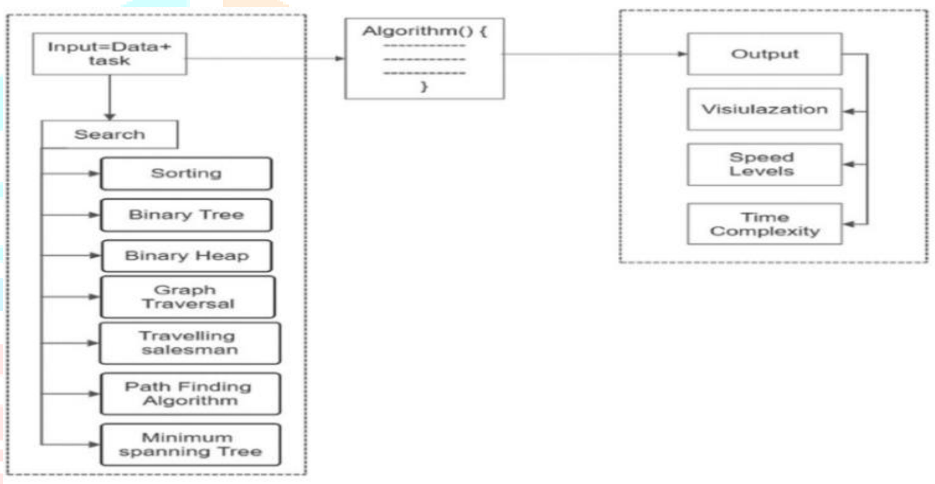

Fig.1 Architectural design

The architectural design of the Algorithm Visualizer integrates multiple components to create an efficient and interactive system for understanding computational processes. The Input Module serves as the starting point, allowing users to input datasets and select tasks such as sorting, searching, or graph traversal. This flexibility accommodates diverse scenarios and learning needs. The Search and Algorithm Selection Module enables users to choose from a comprehensive library of algorithms, including sorting techniques like Bubble Sort and QuickSort, and Dynamic problems like the sudoku, N-Queen, Knight Tour.

The Algorithm Execution Module processes the selected algorithm, breaking it down into step-by-step operations for visualization, ensuring accuracy and responsiveness even for large datasets. The Output Module transforms the algorithm's execution into real-time visualizations, featuring dynamic animations, adjustable speeds, and time complexity analysis. Together, these modules ensure a seamless workflow, where user inputs are processed, algorithms executed, and results visualized in an intuitive and engaging manner. This modular design not only enhances user engagement but also ensures scalability for future expansions.

### IV.I ALGORITHMS

Sorting algorithms like Bubble Sort, Merge Sort, and QuickSort offer unique approaches to data organization. Bubble Sort operates by comparing adjacent elements and swapping them if they are out of order, effectively "bubbling" the largest unsorted element to its correct position in each iteration. While simple and easy to understand, it is inefficient for large datasets due to its $O(n^2)$ time complexity, making it more suitable for educational purposes. Merge Sort, a divide and-conquer algorithm, recursively splits an array into smaller sub-arrays until each contains a single element, then merges them back in sorted order. It is efficient for handling large datasets with a consistent time complexity of $O(n \log n)$, though it requires additional space for temporary arrays. QuickSort partitions an array around a pivot element, placing smaller elements on one side and larger ones on the other. The process is repeated recursively, making it highly efficient with an average time complexity of O(n log n). However, it may degrade to O(n²) if the pivot selection is poor, such as always choosing the first or last element in already sorted data.

Searching algorithms like Linear Search and Binary Search provide contrasting approaches for finding elements in a dataset. Linear Search sequentially checks each element of the array until the target is found or the array ends. It is simple, requires no sorting, and works well for small datasets, but its O(n) time complexity makes it impractical for larger datasets. In contrast, Binary Search is a much faster method that works only on sorted datasets. By repeatedly dividing the search range in half, it eliminates half the elements with each step. This logarithmic reduction makes it highly efficient, with a time complexity of O(log n). Binary Search exemplifies the importance of sorted data and is widely used in applications where quick lookups are essential.

## IV.II METHODOLOGY

The development of the Algorithm Visualizer involves a carefully structured methodology, divided into four critical phases: algorithm selection, framework development, customization, and deployment. This methodology ensures that the tool is both technically robust and educationally effective, catering to a wide range of users from beginners to advanced learners.

### *1) Algorithm Selection*

The first step in the development process is selecting a diverse set of algorithms that will be visualized. These algorithms are categorized into three primary types: sorting, searching, and pathfinding. Sorting algorithms such as Bubble Sort, QuickSort, and Merge Sort are chosen for their foundational importance in understanding data organization. Searching algorithms, including Linear Search and Binary Search, are included to demonstrate basic retrieval operations. Additionally, pathfinding algorithms like Dijkstra's Algorithm are selected for their application in graph theory and optimization problems.

### *2) Framework Development*

Framework development is a pivotal phase where the technical backbone of the Algorithm Visualizer is created. The framework uses Python's Tkinter library for the graphical user interface (GUI) development. Tkinter is widely used for building intuitive and interactive GUIs due to its rich set of widgets, such as buttons, sliders, and text fields. These interactive components provide users with the ability to manipulate the visualizer's behavior—whether adjusting the visualization speed via sliders or selecting different algorithms using buttons.

For rendering dynamic animations that illustrate algorithm execution, the Pygame library is employed. Pygame, a cross-platform multimedia library, enables the creation of interactive visualizations by handling user input and rendering graphics. In the case of the Algorithm Visualizer, Pygame is utilized to animate algorithm operations such as moving bars in sorting algorithms or exploring nodes in graph algorithms. This combination of Tkinter's user-friendly controls and Pygame's real-time animation capabilities ensures that users can observe each algorithm's step by-step execution clearly and smoothly.

Additionally, the framework is designed with modularity in mind. Each algorithm is implemented as a separate module, which makes the tool highly extensible. This modular approach allows for easy updates and additions of new algorithms and features, ensuring the visualizer remains relevant and adaptable to future educational needs. Furthermore, the integration of Pygame's animation engine with Tkinter's controls creates a seamless user experience, combining interactivity with visually compelling animations.

### *3) Customization*

Customization is an essential aspect of the Algorithm Visualizer, allowing users to tailor their experience to meet specific learning objectives and preferences. This feature enhances user engagement and supports active learning. The visualizer provides users with the ability to input their own datasets, such as arrays for sorting or custom graphs for pathfinding. This feature encourages experimentation, helping users understand how different data inputs affect the performance and behavior of algorithms. By testing various scenarios, users can observe how algorithms perform with datasets of varying sizes and complexities.

In addition to dataset input, the tool offers customizable parameters that allow users to adjust settings such as dataset size, input distribution (e.g., random, sorted, or reversed data for sorting algorithms), and animation speed. These customizable features enable users to experiment with different scenarios and observe how algorithm performance changes under various conditions. For example, users can test how an algorithm scales with large datasets or slow down the visualization to study each step in detail. Furthermore, the visualizer offers the option to compare multiple algorithms side-by-side, allowing users to directly contrast their performance. This feature fosters a deeper understanding of algorithm efficiency and aids users in recognizing the strengths and weaknesses of different approaches to problem-solving. Overall, these customization options provide learners with the flexibility to explore edge cases, challenge their understanding, and deepen their comprehension of algorithms.

*4) Deployment*

The final phase of the development process involves deploying the Algorithm Visualizer as a fully functional and accessible tool. Two primary deployment options are considered: web-based and standalone applications. The web-based deployment allows users to access the visualizer through any modern browser, making it highly accessible to a broader audience without the need for installation. The web version is hosted on a reliable platform, ensuring smooth performance and availability to users across different devices and locations. For users who prefer offline access or work in environments with limited internet connectivity, the standalone deployment option is available. This desktop version is compatible with major operating systems, including Windows, macOS, and Linux. It is designed to work seamlessly in offline settings while maintaining full functionality.

Both deployment methods are accompanied by detailed documentation and tutorials, guiding users on how to install, use, and navigate the visualizer. To ensure the tool's compatibility and usability across different devices and operating systems, extensive testing is carried out, and user feedback from beta testers is incorporated. This iterative process helps identify and resolve any issues before the final release, ensuring that the visualizer meets the needs of its users and provides a smooth, intuitive experience. Through thoughtful consideration of accessibility and user experience, the Algorithm Visualizer aims to become a valuable resource for educators, students, and professionals. By offering both web-based and standalone deployment options, the tool ensures that users can access and benefit from the visualizer in a variety of settings, whether for personal learning, academic study, or professional development.

## V FEATURES

*1) Real-Time Visualization*

The core functionality of the Algorithm Visualizer is its ability to animate data changes as they occur during the execution of various algorithms. For sorting algorithms like Bubble Sort or QuickSort, the visualization shows each element comparison and swap as the algorithm progresses. These animations provide users with an intuitive understanding of the algorithm's behavior and logic, breaking down complex operations into digestible visual steps. Graph algorithms, such as Dijkstra's shortest path, demonstrate how nodes are explored, paths are built, and decisions are made in real-time. This feature makes abstract processes concrete, enhancing the learner's ability to connect theoretical concepts with practical execution. Additionally, the animations adapt dynamically to changes in the dataset, ensuring the visualizations remain relevant and accurate regardless of the input size or type.

*2) Metrics For Comparison Across Algorithms*

Understanding algorithm efficiency is critical, and the Algorithm Visualizer addresses this need by providing detailed metrics. These metrics include time complexity, space complexity, and the count of operations such as comparisons, swaps, or recursive calls. For example, when comparing Bubble Sort and Merge Sort, users can observe not only the visual progression but also quantitative data like the number of swaps performed or the recursive depth reached. The tool also displays performance differences in real-time, enabling users to analyze how algorithms scale with larger datasets.

*3) Interactive Controls For Speed Adjustment And Dataset Customization*

Interactivity is a cornerstone of the Algorithm Visualizer. Users have full control over the speed of the animations, which can be adjusted to suit their preferences. Beginners can slow down the visualizations to observe each step in detail, while advanced learners can increase the speed to focus on overall patterns and outcomes. The tool also allows users to input their own datasets, offering flexibility to experiment with various scenarios. For example, users can test sorting algorithms on custom arrays of numbers or analyze graph traversal on unique graph structures. This feature promotes a hands-on, exploratory learning experience, encouraging users to actively engage with algorithms rather than passively observe them. The ability to tweak parameters such as array size, input distribution, or graph connectivity further enhances the educational value of the visualizer.

*4) Educational Insights*

To supplement its visual and interactive elements, the Algorithm Visualizer provides comprehensive educational resources. Each algorithm comes with a detailed explanation of its logic, describing the purpose and strategy behind its steps. For instance, the tool explains how Merge Sort divides the dataset into smaller parts and then merges them in sorted order, or how Dijkstra's algorithm prioritizes paths based on cumulative weights. Alongside these explanations, the tool displays pseudocode to bridge the gap between visualization and implementation, helping users understand how the algorithm translates into actual code. Additionally, discussions of time complexity (e.g., O(n^2), O(n log n)) and space complexity provide valuable insights into the trade-offs associated with different algorithms. These insights are presented in a learner-friendly manner, making complex concepts accessible to a wide audience. By integrating theoretical knowledge with practical demonstration, the visualizer becomes an all-encompassing educational tool.

*5) Holistic Learning Experience*

Combining real-time visualizations, performance metrics, interactivity, and educational insights, the Algorithm Visualizer offers a holistic learning experience. It caters to a diverse audience, from students new to computer science to professionals seeking to deepen their understanding of algorithm efficiency. The tool not only aids in grasping algorithm fundamentals but also fosters critical thinking by allowing users to experiment and draw conclusions from observed behaviour. By bridging the gap between abstract theory and practical application, the Algorithm Visualizer empowers learners to confidently apply algorithms in academic, research, and real-world contexts.

## VI RESULTS AND ANALYSIS

This chapter presents the results obtained from the Algorithm Visualizer project and evaluates its performance in terms of usability, functionality, and educational effectiveness. The primary objective of the project was to provide an interactive platform for understanding and visualizing algorithms. This section includes detailed descriptions of the algorithms**.**

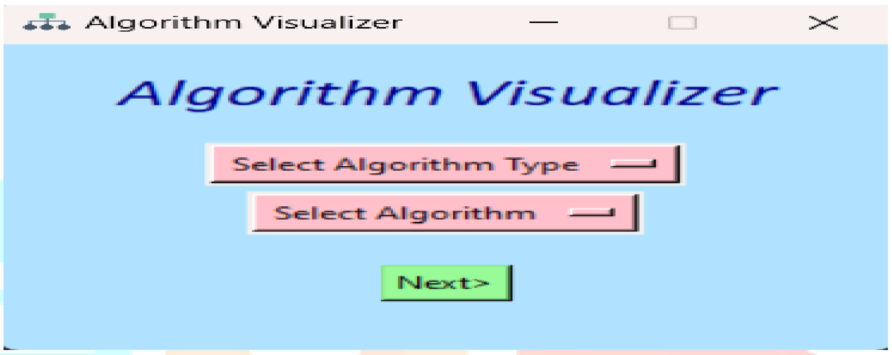

Fig.2 Home page of Algorithm Visualizer

Figure 2 Represents the home page of the Algorithm Visualizer features two main buttons. The first, "Select Algorithm Type," allows users to choose a category of algorithms. The second, "Select Algorithm," displays a list of algorithms within the chosen category, enabling users to visualize their execution.

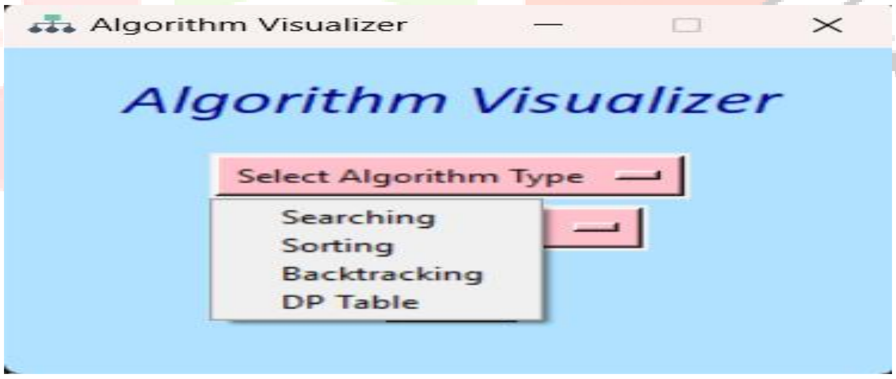

Fig.3 Selecting an Algorithm Type

Figure 3 Represents the "Select Algorithm Type", option allows users to choose a category of algorithms from a dropdown menu, including searching, sorting, backtracking, and DP table

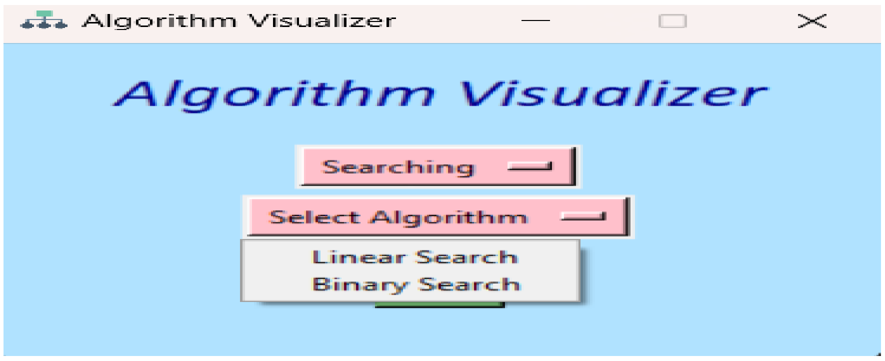

Fig.4 Selecting an Algorithm

Figure 4 Represents it allows users to choose a category of algorithms, such as "Searching." The second, "Select Algorithm," displays a list of algorithms within the chosen category, such as "Linear Search" and "Binary Search."

**Comparative Analysis**

The Algorithm Visualizer was compared to existing algorithm visualization tools to understand its strengths, unique features, and areas for improvement. This analysis highlights how the project performs relative to other tools in terms of usability, functionality, educational value, and performance.

*1) Comparison with Existing Tools*

The Algorithm Visualizer offers several advantages over existing tools, particularly in ease of use, customization, offline access, step-by-step visualization, and resource efficiency. Unlike existing tools, which are often complex for beginners due to advanced UI options, the Algorithm Visualizer provides a simple, stable, and beginner-friendly interface. It also allows greater customization by enabling users to input custom data and control the visualization speed dynamically, a feature often limited in other tools.

While most existing tools require an internet connection and are predominantly web-based, the Algorithm Visualizer operates as a fully offline, lightweight desktop application. Additionally, it enhances step-by-step visualization by allowing users to configure visualization speeds and limit dataset sizes, addressing the issue of overly fast or pre-configured visualizations in other tools. Finally, it is optimized for smooth performance with small to medium datasets, ensuring lower computational requirements compared to existing tools that demand high resources for rendering large data. This combination of features makes the Algorithm Visualizer a robust and accessible solution for algorithm visualization.

*2) User Feedback Comparison*

To evaluate the usability and effectiveness of the Algorithm Visualizer, user feedback was compared with comments on existing tools. Feedback was collected from a small group of students and instructors. The Algorithm Visualizer received praise for its simple, clean, and easy-to-understand user interface, unlike existing tools that were often criticized for being too cluttered and difficult to navigate. The learning experience was significantly enhanced, as the step-by-step animations provided by the Algorithm Visualizer were found to be very helpful, addressing the common issue of animations being too quick to follow in other tools. Customization options were also highlighted as a strong point, with features like custom input and speed adjustment offering greater flexibility compared to the limited controls in existing tools. Overall, users expressed higher satisfaction with the Algorithm Visualizer, describing it as perfect for beginners and learners, in contrast to existing tools, which were deemed more suitable for advanced users.

## VII CONCLUSION

The Algorithm Visualizer project successfully bridges the gap between theoretical concepts of algorithms and their practical understanding through an interactive and intuitive visualization tool. By offering step-by-step graphical representations, the project simplifies the complexities of algorithms, making them accessible to learners at all levels, from beginners to advanced students. It effectively visualizes key algorithms such as sorting algorithms like Bubble Sort, Merge Sort, and Quick Sort, searching algorithms like Linear Search and Binary Search, and backtracking algorithms like Sudoku.

Users can observe how data is processed in real-time, gaining insights into the inner workings of algorithms and analyzing their performance on varying dataset sizes. Dynamic features such as speed control, custom input handling, and step pausing enhance interactivity, fostering an engaging and personalized learning experience. The development process prioritized creating a functional and user-friendly interface, and rigorous testing ensured the delivery of a lightweight, offline application that is resource efficient and caters to the educational needs of students and educators. Performance evaluations highlighted the tool’s ability to handle small to medium datasets efficiently while maintaining smooth visualizations. The Algorithm Visualizer lays a strong foundation for further advancements, including the addition of complex algorithms, comparative analysis tools, and cross platform compatibility, such as web and mobile versions.

Future enhancements like real-time input integration, advanced animations, and live coding environments will further expand its usability and impact. In conclusion, the Algorithm Visualizer is a transformative educational resource that combines simplicity, interactivity, and clarity, empowering learners to understand algorithms and appreciate their behavior and performance across diverse scenarios. This project is a significant step toward making algorithm education more engaging, accessible, and effective for learners worldwide.

## FUTURE SCOPE FOR IMPROVEMENT

The Algorithm Visualizer has demonstrated its effectiveness as an educational tool for understanding algorithms through step by-step visualization. However, there are several opportunities to enhance its functionality, interactivity, and scalability. Future improvements and extensions can further increase the tool's value.

*1) Addition of More Algorithms:*

The current tool supports basic algorithms like sorting, searching, and graph traversal. To broaden its scope, advanced algorithms can be included, such as Heap Sort, Tim Sort, and Radix Sort, alongside a comparative analysis of sorting algorithms to help users identify the most efficient techniques for various datasets. Dynamic programming algorithms, such as the Floyd-Warshall Algorithm for shortest path problems and the Knapsack Problem, can be added to simplify complex concepts through visualization of recursion and memoization. Greedy algorithms, like Dijkstra's Shortest Path, A* Search, and Tree Traversals, can also be incorporated to enhance understanding of graph and tree-based problems.

*2) Support for Real-Time Input:*

Interactivity can be enhanced by enabling real-time data input. Users can draw custom graphs or input adjacency matrices for graph algorithms, manually input dynamic arrays, or upload datasets (e.g., CSV files) for sorting algorithms. Additionally, integrating live datasets, such as stock prices or sensor data, can demonstrate practical applications of algorithms in real-world scenarios.

*3) Improved User Interface and Experience:*

Upgrading the visualizer’s user interface can improve engagement and accessibility. Features like theme customization (e.g., dark and light modes), enhanced animations with smoother transitions, and color-coded steps for comparisons, swaps, or traversals can provide a better explanation of algorithms. Progress bars and step counters can display execution progress, including the number of steps, comparisons, or iterations. Tooltips and inline explanations for algorithm steps can further aid understanding.

*4) Mobile and Web Integration:*

Expanding the project to support multiple platforms can increase its accessibility. A mobile-friendly version will allow users to learn on the go, while a web-based application ensures ease of access without the need for installation. Hosting on the cloud can facilitate visualization of larger datasets and provide options for saving progress. Cross-platform compatibility for Windows, Linux, macOS, and mobile devices will ensure seamless usage across different environments.

## VIII ACKNOWLEDGEMENT

We, the authors, express our heartfelt gratitude to Jawaharlal Nehru New College of Engineering and the Department of Information Science and Engineering for providing the resources and support required to carry out this work.

## REFERENCES

[1] Ashish Kumari, Manav Mittal, Vipul Jha, Abhishek Sahu, “Algorithm Visualization – Modern Web-Based Visualization of
Sorting and Searching Algorithms”, In Mili Publications, Vol. 21, No. 5, pp. 2721 – 2726, 2022.

[2] Rohit L. Ugile, Ritesh B. Barure, Gajanan S. Sawant, “Review Paper on Algorithm Visualizer ”, In International Research
Journal of Modernization in Engineering Technology and Science, Vol. 04, No. 11, pp. 2582 – 5208, 2022.

[3] Akash Athare, Omkar Gulave, Divvij Jaitly, Akshay Sharma, Prof. R. S. Waghole, “Visual Rendering of Path Finding
Algorithm”, In International Journal of Creative Research Thoughts (IJCRT), Vol. 11, No. 5, pp. 2320 – 2882, 2023.

[4] Leena I. Sakri, Deepa Bendigeri, Sachin Joshi, Arpita Hiremagadi, Pooja Gurumurti, “Visualizer for Algorithms and Path
Finding with User Interface Design ”, In International Journal for Research in Applied Science & Engineering Technology
(IJRASET), Vol. 11, No. 05, pp. 2321 – 9653, 2023.

[5] Sarvesh Arora, Dr. K. C. Tripathi, Dr. M. L. Sharma, “Graph Algorithm Visualizer”, In Iconic Research and Engineering
Journals, Vol. 06, No. 06, pp. 2456 – 8880, 2022.