International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 05, IIRA 5.0 (2026)

# Route Exploration Tool: An Interactive Visualization Platform for Pathfinding Algorithm

Jayanti Gupta
Department of CSE
Shri Ramswaroop Memorial College of Engineering &Management
jayantigupta.489@gmail.com

Karan Sahani
Department of CSE
Shri Ramswaroop Memorial College of Engineering &Management
karansahani223@gmail.com

Er. Shivam Dixit
Shri Ramswaroop Memorial College of Engineering &Management
shivamdixit.cs@srmcem.ac.in

Er. Sarika Singh
Shri Ramswaroop Memorial College of Engineering &Management
sarikasingh2494@gmail.com

**Abstract— Pathfinding algorithms underpin many modern technologies-from network routing and autonomous navigation to web optimization and game development. The Route Exploration Tool is an interactive, web-based platform that visualizes and compares various pathfinding techniques (BFS, DFS, Dijkstra's, A*), bridging the gap between abstract theory and practical application. This paper presents a comprehensive account of the literature sourcing, development methodology, and evaluation process behind the tool. In addition, it explains the need that motivated the project, its current applications in educational and professional settings, and potential future use cases across diverse fields.**

**Keywords: Pathfinding, Algorithm Visualization, Research Methodology, Interactive Tool, Educational Technology, System Development, Future Applications**

## INTRODUCTION

Path-finding is the computational process of determining the best route between two points within a given space. In practical applications, these algorithms underpin systems such as GPS navigation, robotics, and network traffic management. Despite their significance, the abstract nature of these algorithms often makes them difficult to master through traditional lecture methods alone.

To address this challenge, our project presents an interactive visualization tool that offers real-time insights into popular path-finding algorithms. Built on ReactJS, the tool leverages modern web technologies to enable users to:

- Define a customizable 2D grid environment
- Place start and end nodes.
- Insert obstacles (walls) and adjust node weights.
- Select and animate the execution of different algorithms.

Feedback from initial user trials indicates that the intuitive interface and real-time updates significantly enhance comprehension, making abstract concepts more tangible

## LITERATURE SURVEY

**Borissova.D .,& Musktakerov,I.(2015**). In their paper on e-learning tools for shortest path algorithms, Borissova and Musktakerov emphasize the value of interactive learning systems in education. The authors propose an e-learning module that allows students to explore classical algorithms like Dijkstra's through visualization. Their work underlines the importance of visual interactivity for better understanding algorithm logic. [1]

**Patil,V., Patil, S., et al. (2022).** This study focuses on the visualization of sorting and pathfinding algorithms. By implementing visual simulation tools, the authors aim to enhance students' comprehension of algorithm efficiency and complexity. The visualizer demonstrates how different algorithms perform on various data sets, helping learners compare their performance.[2]

**Chauhan, M., et al. (2024).** The paper presents "Algo Graph," a visualization tool that uses interactive graphs to demonstrate pathfinding algorithms. Their approach enhances conceptual clarity through real-time feedback and parameter-based interaction. The tool is designed to bridge the gap between theoretical learning and practical understanding.[4]

**Halim, S., & Halim, F. (2011).** The VisuAlgo project offers an extensive suite of animated visualizations for numerous algorithms, including shortest path algorithms. Designed for both educators and students, it allows manipulation of data inputs and step-by-step tracing, supporting self-paced learning.[5]

**Nash, A., Koenig, S., et al. (2007).** In their work on Theta*, Nash and Koenig propose an any-angle pathfinding algorithm that improves on traditional grid-based techniques. The paper introduces a method that allows agents to find shorter and more realistic paths in navigational environments.[6]

**Oh, S., & Leong, H. W. (2017).** This paper presents Edge N-Level Sparse Visibility Graphs (ENLSVGs), which significantly enhance the performance of any-angle pathfinding. It emphasizes hierarchical taut paths to efficiently calculate optimal routes with fewer computations.[7]

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 05, IIRA 5.0 (2026)

**Sinyukov**, **D. A., & Padir, T. (2017).** CWave is introduced as a fast, any-angle pathfinding method on grids. It enables single-source routing with reduced computational overhead. The study is significant for real-time applications in robotics and AI navigation.[8]

**Liu, J., Dwyer, T., et al. (2020).** This work explores user-centric optimization systems. While not limited to pathfinding, the interactive visual features described in the study can be applied to algorithm visualization for decision-making tasks.[9]

**Jolakoski, P., Deja, J. A., et al. (2024).** The authors present a novel teaching tool that combines robotics with overlaid projections to teach shortest path algorithms. This tangible approach promotes experiential learning and makes abstract algorithmic concepts more accessible.[10]

**Ferguson, D., & Stentz, A. (2005).** Field D is introduced as a real-time interpolation-based replanning algorithm. It ensures path continuity and smoothness, which are often lacking in discrete grid-based planners.[11]

**Nash, A., Koenig, S., & Likhachev, M. (2011).** Lazy Theta* improves upon Theta* by postponing certain path validation checks, thus reducing overhead. This modification preserves optimality while significantly boosting efficiency in complex terrains.[12]

**Baker, et al. (1996).** Mocha, a web-based algorithm animation system, allows real-time algorithm demonstration via the web. It uses multimedia interaction and user-controlled playback, helping users understand algorithm flow.[13]

**Institutional Archives: Tango, XTango, GAIGS, DynaLab, SWAN, JAWAA, FLAIR, POLKA, Samba** These historical tools represent foundational efforts in algorithm visualization. Each tool was designed for educational contexts, employing animations and simulations to make abstract data structures and algorithms comprehensible.[14]

**Pathfinding Visualizer by Ivan Sem** Ivan Sem's web-based tool allows visualization of algorithms like A* and Dijkstra. Users can set start/end nodes, adjust weights, and view the exploration process, helping learners understand real-time algorithm behavior.[15]

**"Pathfinding Visualizer: A Survey of the State-of-Art"** This comprehensive survey discusses implementation techniques and features of pathfinding visualizers. It covers challenges, UI considerations, and pedagogical implications for educational software.[16]

**"Pathfinding Visualizer Using Multiple Graph Algorithms"** This work compares the effectiveness of Dijkstra, DFS, BFS, and A* algorithms in visual form. It supports the use of animation as a teaching aid to highlight strengths and limitations of each approach.[17]

**"Pathfinding Visualizer" by Rohit Singh et al.** This project focuses on dynamic visual representation for real-time decision-making scenarios. It emphasizes algorithm suitability for applications in mapping, robotics, and transportation systems.[18]

**"Shortest Path Finding Visualizer"** This educational project uses a web-based GUI to simulate shortest path algorithms. It promotes algorithm literacy through interactive problem-solving.[19]

**"Visualising Path Finding Algorithms Application Development and Implementation"** This paper reviews the implementation of several pathfinding algorithms in a visual format, providing a holistic approach to algorithm education with use cases and design recommendations.[20]

## PROPOSED SYSTEM

The proposed system is a web-based application designed for dynamic visualization of path-finding algorithms. Its core functionalities are delivered through an intuitive, responsive interface that allows users to configure grid dimensions, place obstacles, and observe algorithmic processing in real-time.

### System Features

- **Grid Customization:** Users can define grid dimensions and adjust cell properties.
- **Interactive Obstacle Placement:** Walls can be drawn manually by clicking grid cells, altering their state, and blocking algorithm traversal.
- **Algorithm Selection:** The system supports Dijkstra's, A*, BFS, and DFS, allowing users to select and compare different approaches.
- **Real-Time Animation:** The tool animates algorithm execution step-by-step, highlighting node exploration and the final optimal path.
- **Reset and Clear Functions:** Users can easily reset the grid or clear the current path for new experiments.

## METHODOLOGY

The application is built using ReactJS for its component-based UI and JavaScript for implementing the core algorithms. This section outlines the system's operational flow and interactive features.

### Technology Stack

- **Frontend:** ReactJS, HTML5, CSS3, JavaScript.
- **Backend (Optional):** A lightweight Node.js server can be integrated for scalability.
- **Visualization:** Managed via React state updates for real-time, dynamic rendering.

### Flow of Operation

**Initialization:**

- The application loads a customizable 2D grid.
- Default nodes are set for the start and finish positions.

**User Interaction:**

- **Placing Walls:** Users click on grid cells to toggle them as walls. This updates the cell's state, changes its color, and prevents it from being traversed by the algorithm.
- **Algorithm Selection:** A control panel allows the user to choose an algorithm (Dijkstra's, A*, BFS, DFS), each linked to its corresponding function.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 05, IIRA 5.0 (2026)

- **Parameter Adjustment:** Users can modify animation speed, grid size, and other parameters through intuitive sliders and input fields.
- **User Feedback:** Preliminary user testing revealed that immediate visual feedback—such as color changes on cell toggling and dynamic instructions—greatly improves engagement and understanding.

**Algorithm Execution:**

- **Dijkstra's Algorithm:** Uses a priority queue to select nodes with the smallest tentative distance.
- *An Algorithm**: Combines actual distance from the start node with a heuristic (Manhattan distance) to determine the most promising path.
- **BFS:** Explores nodes level-by-level, ensuring the shortest path in unweighted grids.
- **DFS:** Utilizes a stack-based approach to explore deep paths and backtracks when necessary.
- Each algorithm updates the grid in real-time, marking visited nodes and drawing the final path.

**Visualization and Feedback:**

- The grid dynamically updates, with distinct colors indicating unvisited nodes, visited nodes, obstacles, and the final path.
- **Real-Time Animation:** Transition effects and adjustable delays help illustrate the progression of each algorithm.
- **Error Handling:** If no path is found, the system alerts the user and prompts them to reset the grid.

**Model Description Diagram**

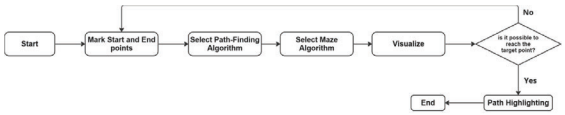



**Fig 1:** Working flow chart for user

**MODEL AND ANALYSIS**

The system has been implemented as an interactive, web-based application using contemporary front-end technologies. It offers a comprehensive interface for selecting algorithms, placing obstacles, and observing real-time visualizations. The following subsections describe the system's components with corresponding screenshots

**Login and Signup Interface**

The application starts with a secure authentication system that supports Login and Signup functionalities, allowing users to save preferences and revisit the tool. A Guest Login option is also available for users who wish to explore the system without creating an account.

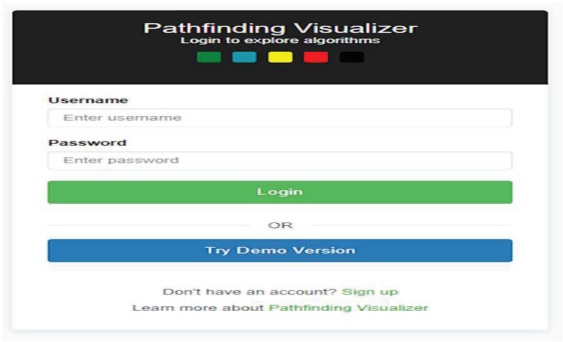



**Fig 2:** *Login and Signup Page with Guest Login Option*

**Home Screen with Instructions and User Guide**:
After logging in, users are directed to a home screen that provides an interactive layout with a step-by-step guide, tooltips, and brief instructions. This screen explains how to draw walls, select algorithms, and start visualizations. User feedback indicates that the guided interface significantly improves ease of use.

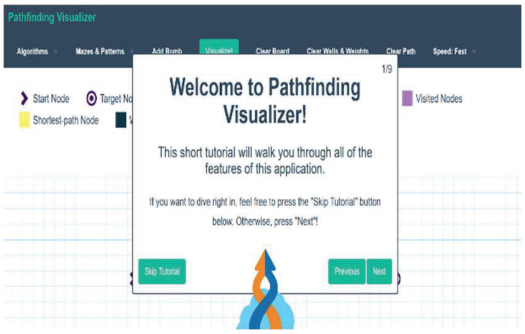



***Fig 3**: Home Screen with Instructions and User Guide*

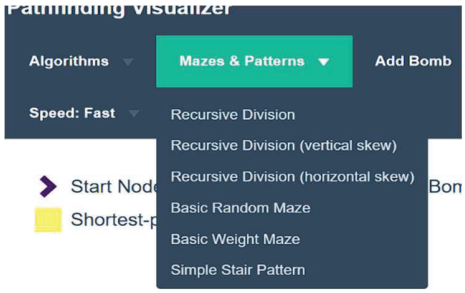
**Algorithm Selection and Grid Preparation**:
User can then choose from several path-finding algorithms:

- Dijkstra's Algorithm
- A* (A-star) Algorithm
- Breadth-First Search (BFS)
- Depth-First Search (DFS)

After selection, users interact with the grid to:

- Set start and end nodes.
- Draw walls (obstacles) manually.
- Reset or clear the grid as needed.

This setup allows users to simulate real-world routing challenges. Technical logs capture metrics such as the number of nodes explored and time taken, providing insights into algorithm performance.



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 05, IIRA 5.0 (2026)

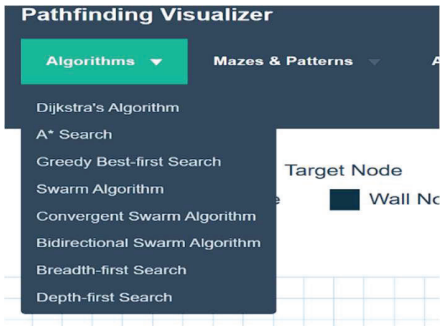



**Fig 4:** *Algorithm Selection and Interactive Grid with Walls (Barriers)*

**Visualization in Progress**

Upon clicking the "Visualize" button, the selected algorithm begins processing:

- The algorithm explores nodes and updates their states.
- The optimal path is drawn between the start and finish nodes.
- Color-coded feedback distinguishes visited nodes from the final path.

Real-time logs and performance metrics are available in the background, offering additional technical depth for analysis.

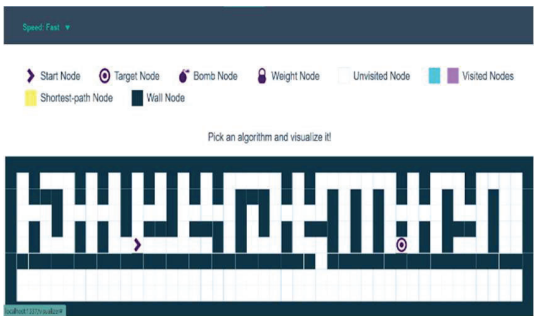



*Fig 5: Real-Time Visualization of the Algorithm in Action*

**Complete Interface Overview**

The final interface presents all system features in a cohesive layout. It includes:

- A control panel with start/stop/reset buttons.
- An algorithm dropdown menu.
- Speed controls and adjustable parameters.
- A color-coded grid and status indicators (start node, end node, visited nodes, optimal path).

User surveys have shown that this comprehensive interface enhances both the learning experience and the overall usability of the tool.

## FUTURE SCOPE

The Path-Finding Visualizer is designed as a foundational educational tool, and its potential can be extended significantly. Future enhancements may include:

- **3D Visualization:** Implementing 3D grids to simulate real-world navigation and spatial complexity, will provide more realistic training scenarios for robotics and autonomous systems.
- **Advanced Algorithm Support:** Integrating additional algorithms such as Bellman-Ford, Floyd-Warshall, and even hybrid approaches. Comparative studies on performance metrics (e.g., execution time, node exploration) could be incorporated.
- **Mobile and IoT Integration:** Adapting the tool for mobile devices and leveraging IoT sensors for real-time data feeds. This would enable dynamic updates in changing environments, making the visualizer useful in field robotics or smart city applications.
- **Maze Generation and Adaptive Learning:** Developing algorithms for automated maze creation (using recursive division or randomized methods) and adaptive learning modules that tailor difficulty levels based on user performance.
- **Backend and Collaborative Features:** Adding a server component to store user configurations, support multi-user sessions, and facilitate collaborative learning. This would enable features like user analytics, session sharing, and integration into academic curricula.
- **Integration of Real-Time Analytics:** Embedding tools to capture and display real-time metrics (such as average node processing time, path length comparisons, etc.) to provide deeper technical insights and feedback for both students and researchers..

## SPECIFICATIONS

**Advantages**

- **Interactive Learning:** Real-time visualizations enhance understanding and retention of complex algorithms.
- **User-Friendly Interface:** A clean, accessible design that requires no installations.
- **Modular and Scalable:** The tool is built to be easily extendable for future enhancements and additional algorithm integrations.
- **Educational Impact:** Bridges theoretical concepts with practical applications, validated by positive user feedback

**Disadvantages**

- **Performance Limitations:** Real-time rendering may face challenges with very large grids or highly complex scenarios.
- **2D Scope:** Currently limited to 2D grid simulations, which may not fully capture all real-world path-finding challenges.
- **Resource Intensive:** Intensive animations and real-time processing can be demanding on older hardware.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14, Issue 05, IIRA 5.0 (2026)

- **Challenges & Optimization:** Balancing smooth, real-time animations with performance.
- **Scalability:** Ensuring the application can handle larger datasets and more complex algorithmic processes.
- **User Interface Consistency:** Maintaining ease of use while integrating advanced accuracy and adaptability.

## CONCLUSION

The Path-Finding Visualizer represents a significant leap in educational tools for computer science. By transforming abstract algorithmic concepts into interactive, visual experiences, the tool facilitates a deeper understanding of path-finding techniques. With user-friendly controls, real-time feedback, and comprehensive analysis metrics, it serves as a valuable resource for both novice and advanced learners. Future enhancements, including 3D visualization and advanced algorithm integrations, promise to extend its applicability across a broader range of fields, from robotics to smart city navigation.

## REFERENCES

[1] *Borissova, D., & Mustakerov, I. (2015). E-learning tools for shortest path algorithms. International Journal of Computer Applications.*

[2] *Patil, V., Patil, S., et al. (2022). Visualization of sorting and path-finding algorithms. International Research Journal of Engineering and Technology.*

[3] *Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms. MIT Press.*

[4] *Chauhan, M., et al. (2024). Algo Graph: Exploring Path Finding Algorithms through Interactive Visualization. International Journal of Science, Engineering and Technology.*

[5] *Halim, S., & Halim, F. (2011). VisuAlgo: Visualising Data Structures and Algorithms through Animation.*

[6] *Nash, A., Koenig, S., et al. (2007). Theta: Any-Angle Path Planning on Grids. Proceedings of the AAAI Conference on Artificial Intelligence.*

[7] *Oh, S., & Leong, H. W. (2017). Edge N-Level Sparse Visibility Graphs: Fast Optimal Any-Angle Pathfinding Using Hierarchical Taut Paths. Proceedings of the International Symposium on Combinatorial Search.*

[8] *Sinyukov, D. A., & Padir, T. (2017). CWave: Fast Single-source Any-Angle Path Planning on a Grid. IEEE.*

[9] *Liu, J., Dwyer, T., et al. (2020). Supporting the Problem-Solving Loop: Designing Highly Interactive Optimisation Systems. arXiv:2009.03163.*

[10] *Jolakoski, P., Deja, J. A., et al. (2024). Teaching Shortest Path Algorithms With a Robot and Overlaid Projections. arXiv:2411.15535.*

[11] *Ferguson, D., & Stentz, A. (2005). Field D: An Interpolation-Based Path Planner and Replanner. IEEE.*

[12] *Nash, A., Koenig, S., & Likhachev, M. (2011). Lazy Theta: Any-Angle Path Planning and Path Length Analysis in 3D. Proceedings of the AAAI Conference on Artificial Intelligence.*

[13] *Baker, et al. (1996). Mocha: A Web-Based Animation Algorithm System. [Referenced in Mocha system documentation]*

[14] *Additional references on Tango, XTango, GAIGS, DynaLab, SWAN, JAWAA, FLAIR, POLKA, and Samba are derived from institutional archives and historical studies on algorithm visualization.*

[15] *Pathfinding Visualizer by Ivan Sem This tool allows users to select algorithms, set start/end points, and visualize the pathfinding process, including handling weighted nodes. It supports algorithms like Dijkstra's and A*, and provides controls to navigate through the execution history.*

[16] *Pathfinding Visualizer: A Survey of the State-of-Art This paper provides a comprehensive guide to building a pathfinding algorithm visualizer using A* and Dijkstra's algorithms. It discusses the background, implementation guidance with customization options, and a brief analysis of complexities. The paper also covers features, applications, drawbacks, and future work of the visualizer, serving as a valuable resource for anyone interested in developing such tools.*

[17] *Pathfinding Visualizer Using Multiple Graph Algorithms*
*This study implements various pathfinding algorithms—Dijkstra, DFS, BFS, and A*—to provide visual aids for researchers, educators, and students. It demonstrates the theoretical usability of an e-learning application designed to help students construct mental models for understanding shortest path algorithms.*

[18] *Pathfinding Visualizer by Rohit Singh et al.*
*This project focuses on creating captivating visual representations of various algorithms that find the most efficient route from a starting point to a destination, considering factors like time and cost. The research explores the optimal applications of different pathfinding algorithms, with practical implications in traffic management, digital mapping, robotics, and game development.*

[19] *Shortest Path Finding Visualizer*
*This project aims to create a web-based e-learning tool to visualize shortest path algorithms. It illustrates conceptual applications through implementations of algorithms such as Dijkstra's and DFS, providing an interactive platform for users to understand how these algorithms work.*

[20] *Visualising Path Finding Algorithms Application Development and Implementation*
*This study provides an overview of pathfinding algorithms and their implementations, including Dijkstra's algorithm, A* Search, Greedy Best-first Search, Swarm Search, Breadth-first, and Depth-first Search. It aims to help users understand different algorithms and programming concepts, offering a fundamental understanding of creating navigational tools.*