NIRF-2024
Engineering Rank Band (151-200)
Pharmacy Rank - 77
Innovation Rank Band (11-50)

www.kiet.edu
Delhi-NCR, Ghaziabad

KIET GROUP OF INSTITUTIONS
Connecting Life with Learning

Accredited by
NAAC with Grade "A+"

Mini Project-2 (ID201B)
Even Semester
Session 2024-25

Algorithm Visualizer

**Kriti Anand (202410116100104), Khushi Kumari(202410116100101),
Kiran Yadav(202410116100102), Kumar Kartik(202410116100105)**

Project Supervisor:
Ms. Divya Singhal

Content

- Introduction (1 slide)
- Literature Review (1 slides)
- Objective of the Project (1 slide)
- Technology
  - Hardware Requirements
  - Software Requirements
- Modules (2-3 slides)
- Workflow (1 slide)
- Reports
- References (1 slide)

Algorithm Visualizer

A **web-based tool** to visualize **sorting and graph algorithms** step by step.

Helps users **understand the execution process** of different algorithms.

Understanding algorithms can be challenging, especially without a clear visualization. Our **Algorithm Visualizer** provides an **interactive and engaging** way to **learn sorting and graph algorithms** step by step. Unlike existing tools, we integrate a **backend with MongoDB** to store **user inputs, execution history, and preferences**. Using **React.js for UI and Node.js with WebSockets**, we ensure **smooth real-time visualization**. This project is designed to help students and developers grasp complex algorithms **efficiently and intuitively**

Literature Review

Existing Solutions

Algorithm visualizers exist but **they mostly focus on single type of algorithm only**.

Most tools focus only on **visual representation** without data storage.

Challenges in Existing Systems

No feature to **save user executions or replay past runs**.

Performance issues with **large datasets** in frontend-based solutions.

Proposed Solution

Backend Integration for execution tracking.

User-defined inputs & execution history storage.

Technology Stack

Frontend

- **React.js** (for UI)
- **CSS** (for styling)

Backend

- **Node.js + Express.js** (API handling)
- **MongoDB** (for storing user execution data)

Authentication

- **JWT Authentication**

Hardware Requirements

Development Environment

- **Laptop/Desktop** (Minimum: **4GB RAM, i3 processor** | Recommended: **8GB RAM, i5/i7**)

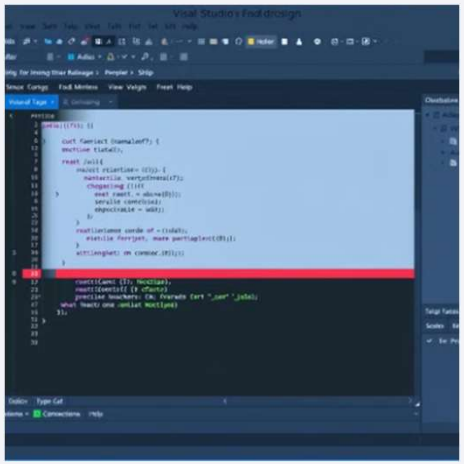
Server Requirements

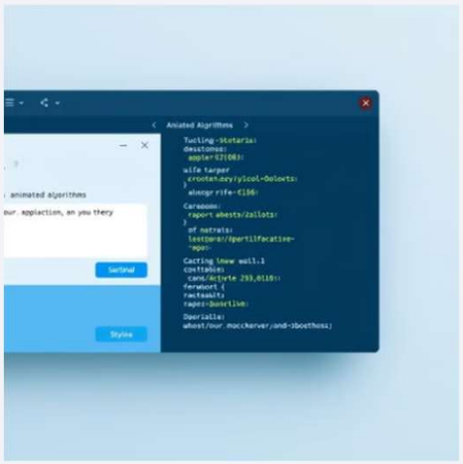
- Node.js server with **MongoDB**

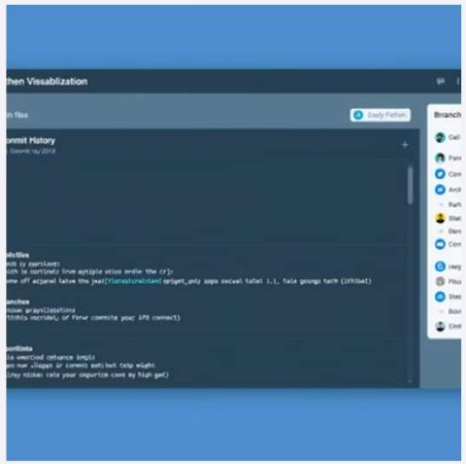
Client Requirements

- Any **modern browser** (Chrome, Firefox, Edge)
- Works on PCs, tablets, and mobile devices

Software Requirements







1 Programming Languages & Frameworks

Frontend: React.js, Tailwind CSS

Backend: Node.js, Express.js

Database: MongoDB

2 Development Tools

VS Code (Code Editor)

Postman (API Testing)

MongoDB Compass (Database Management)

Git & GitHub (Version Control)

Modules

1

Module: Algorithm Visualization

- **Module -1: Sorting Algorithms:** Bubble Sort, Merge Sort, Quick Sort, Insertion Sort
- **Module 2: Find path in a maze Problem**
- **Module 3: N Queen Problem**

2

Module 4: User Inputs & Controls

- Dropdown for selecting **sorting or graph algorithms**
- Custom **array input & execution speed selection**
- Buttons to **Start, Pause & Reset**

3

Module 5: Backend Integration

- Store **user-generated inputs** and execution history in MongoDB
- Retrieve **predefined algorithm descriptions**
- **JWT Authentication**

Workflow

Step 1

1

User selects **algorithm type, inputs array & speed**, clicks "Start".

Step 2

2

Frontend **sends request to backend** for execution.

Step 3

3

Frontend updates visualization dynamically based on received steps.

Step 4

4

Execution completes → User can **view history & replay past executions**.

Objective of the Project

1 Build an interactive algorithm visualizer

Supporting multiple algorithms for comprehensive learning experience.

2 Store user execution history

In **MongoDB** for future reference and analysis.

3 Allow custom features

Input selection, step speed adjustment, and smooth animations.

4 Educational Tool

Helps students learn algorithms in a more interactive and visual way, making abstract concepts easier to understand.

Reports & Data Storage

Project Summary

Simplifies **algorithm learning** with graphical representation

Visualization

Users can **visualize step-by-step execution** of algorithms

Learning Experience

Provides **an interactive & engaging learning experience**

Challenges & Solutions

Algorithm Efficiency with Large Datasets

- **Challenge:** Performance issues with large datasets.
- **Solution:** Backend optimized using **async/await** and **dataset size limits**.

Future Work

- **Support for More Algorithms** → A*, Dijkstra, Dynamic Programming
- **Improved Performance** → Optimize rendering & backend processing
- **User Customization** → Adjustable themes, colors, and speed
- **Interactive Learning Mode** → Hints, step explanations, and quizzes

References

React.js

https://react.dev/

Node.js & Express

https://expressjs.com/

MongoDB

https://www.mongodb.com/docs/

Sorting Algorithms

https://en.wikipedia.org/wiki/Sorting_algorithm

Graph Algorithms

https://en.wikipedia.org/wiki/Graph_traversal