Lab Activity 3

Web-based Interactive Algorithm Visualizer

Laboratory Activity: Web-Based Interactive Pathfinding Visualizer

Course: Web Engineering

Objective:

- *Develop a **JavaScript-based pathfinding visualizer** using Dijkstra's Algorithm or A Search*.
- Apply **graph traversal algorithms** to simulate how a user moves from a starting point to an endpoint on a grid.
- Create a web application with an interactive UI where users can **set obstacles, define paths, and visualize the algorithm's execution**.
- Understand **algorithmic efficiency, UI interactivity, and asynchronous JavaScript**.

Constraints on Using Generative AI

✅ **Allowed:**

- Autocompletion (e.g., GitHub Copilot, ChatGPT for code suggestions).
- Generating code scaffolds or templates (e.g., setting up the HTML structure).
- Debugging assistance.

Project Requirements

1. UI Requirements

- A **10x10 grid** where each cell represents a possible position.
- Users can:
    - **Click to place obstacles (walls).**
    - **Set a start and end point.**
    - **Click a "Find Path" button to visualize the algorithm in action.**

- The grid updates dynamically to **show the algorithm's pathfinding process**.

2. Algorithm Requirements

- Implement **Dijkstra's Algorithm** or *A Search** in JavaScript.
- Visually represent the **exploration of nodes**, the **final shortest path**, and **obstacles**.
- Ensure an **optimized** and **efficient** implementation.

3. JavaScript Implementation

- **Object-Oriented Programming (OOP)** encouraged (create `Node`, `Grid`, `Algorithm` classes).
- Use **asynchronous execution** (`setTimeout()` or `requestAnimationFrame()`) to animate the algorithm.
- Optimize for **performance** (avoid redundant calculations).

4. Additional Challenges

- Allow users to **resize the grid dynamically**.
- Implement **speed control** (slow, medium, fast visualization).
- Add **weighted nodes** (e.g., some paths cost more to traverse than others).
- Store and **load previously defined grids** using local storage.