Rajang Digital Solutions Sdn. Bhd.
(1257741-X)
Lot 141, Jalan Bampfylde,
93200 Kuching, Sarawak, Malaysia.
https://www.rajang.com
SST REG NO : Y60-2209-32000010

Software Developer Challenge: Graph Network with D3.js and ReactJS

Challenge Duration: 2-5 Days

Objective:

Design a Graph Network using D3.js within a ReactJS application. The Graph Network should have the following functionalities:

1. Render all nodes and edges.
2. Add a node or edge.
3. Remove a node or edge.

Requirements:

- Visualization: Display nodes as circles and edges as lines connecting the nodes within a React component.
- Interactivity:
    - Add Node/Edge: Provide functionality to add a new node or an edge between existing nodes.
    - Remove Node/Edge: Allow users to remove a node or edge, ensuring connected edges are also removed when a node is deleted.
- Dynamic Graph: Ensure the graph updates and re-renders whenever a node or edge is added or removed.
- State Management: Manage the graph data (nodes and edges) using React's state management (e.g., `useState` or `useReducer`).

Instructions:

1. Use ReactJS as the framework and D3.js for graph visualization.

2. Implement a simple UI that:
    - Allows users to add a new node.
    - Allows users to add an edge between two nodes.
    - Allows users to remove a node or an edge.
3. Make sure the graph dynamically updates based on user interaction, maintaining the ReactJS state for nodes and edges.
4. Ensure the solution is well-structured and easy to maintain.

Bonus:

- Add features such as zoom and drag for the graph visualization.
- Implement node and edge highlighting on hover or selection.
- Use TypeScript to type the React components and D3.js interactions.
- In the solution provide use cases and test data on how graph networks can be used in the real world.

Submission:

- Submit your solution as a GitHub repository.
- Include a `README.md` that explains how to run the application, your design choices, and any additional libraries or tools used.