**Software Requirements Specification (SRS)**

**Project Title: Development of a JavaScript-based Graph Editor**

Prepared By: **Ajay Kumar**

SubmittedTo: **Utpal Sharma**

Roll No: **CSB22060**

**1. Introduction**

**1.1. Purpose**

1.1.1. This document specifies requirements for a JavaScript-based Graph Editor application.

**1.2. Scope**

1.2.1. The system will enable users to create, edit, and visualize graph structures.

1.2.2. The system will demonstrate graph algorithms: BFS, DFS, Prim's, Kruskal's, Dijkstra's, and Bellman-Ford.

1.2.3. The system will be a web-based application requiring no server-side processing.

**1.3. Definitions**

1.3.1. Graph: Structure with vertices (nodes) and edges (connections).

1.3.2. BFS/DFS: Breadth/Depth-First Search algorithms.

1.3.3. MST: Minimum Spanning Tree.

1.3.4. Shortest Path: Route between nodes with minimum total weight.

**1.4. Target Users**

1.4.1. Computer Science students learning graph concepts.

1.4.2. Educators teaching graph theory and algorithms.

1.4.3. Developers working with graph-based applications.

**2. Overall Description**

**2.1. Product Context**

2.1.1. Standalone web application running in modern browsers.

2.1.2. Utilizes local browser storage for data persistence.

**2.2. Constraints**

2.2.1. Supports Chrome, Firefox, Safari, Edge.

2.2.2. Performs well with graphs up to 100 nodes and 500 edges.

2.2.3. Uses standard web technologies (HTML5, CSS3, JavaScript).

**3. Functional Requirements**

**3.1. Graph Creation and Management**

**3.1.1. Graph Basics**

3.1.1.1. Create new directed/undirected graphs.

3.1.1.2. Name and describe graphs.

3.1.1.3. Switch between multiple saved graphs.

**3.1.2. Node Operations**

3.1.2.1. Add, select, move, and delete nodes.

3.1.2.2. Label nodes with custom text.

3.1.2.3. Customize node appearance (color, size, shape).

**3.1.3. Edge Operations**

3.1.3.1. Add and delete edges between nodes.

3.1.3.2. Set edge weights with numerical values.

3.1.3.3. Toggle edge direction (for directed graphs).

3.1.3.4. Customize edge appearance (color, thickness).

**3.1.4. Graph Layout**

3.1.4.1. Apply automatic layouts (force-directed, hierarchical, grid).

3.1.4.2. Manually adjust node positions.

**3.2. Algorithm Visualization**

**3.2.1. BFS Visualization**

3.2.1.1. Show level-by-level exploration from a user-selected start node.

3.2.1.2. Display traversal order and visited nodes.

**3.2.2. DFS Visualization**

3.2.2.1. Show recursive exploration from a user-selected start node.

3.2.2.2. Display discovery/finish times and edge classifications.

**3.2.3. Prim's Algorithm**

3.2.3.1. Visualize MST generation from a user-selected start node.

3.2.3.2. Show selected edges and display total MST weight.

**3.2.4. Kruskal's Algorithm**

3.2.4.1. Visualize MST generation by sorting edges and union-find process.

3.2.4.2. Display total MST weight and connected components.

**3.2.5. Dijkstra's Algorithm**

3.2.5.1. Show shortest path calculation from user-selected source node.

3.2.5.2. Display distance values and shortest paths.

**3.2.6. Bellman-Ford Algorithm**

3.2.6.1. Visualize shortest path with support for negative weights.

3.2.6.2. Detect and display negative cycles.

**3.2.7. Algorithm Controls**

3.2.7.1. Play, pause, step through, and reset animations.

3.2.7.2. Adjust animation speed.

3.2.7.3. Display algorithm status and current step explanation.

**3.3. Data Management**

**3.3.1. Storage Operations**

3.3.1.1. Save/load graphs from browser storage.

3.3.1.2. Auto-save to prevent data loss.

**3.3.2. Import/Export**

3.3.2.1. Export graphs as JSON and images.

3.3.2.2. Import graphs from JSON format.

**4. Non-Functional Requirements**

**4.1. Performance**

4.1.1. Render graphs up to 100 nodes without lag.

4.1.2. Respond to user interactions within 100ms.

4.1.3. Support smooth animations on standard hardware.

**4.2. Usability**

4.2.1. Provide intuitive interface with minimal training required.

4.2.2. Include tooltips for common operations.

4.2.3. Offer a simple getting-started guide for new users.

**4.3. Reliability**

4.3.1. Prevent data loss through auto-save mechanisms.

4.3.2. Validate user inputs to prevent errors.

4.3.3. Provide clear error messages when issues occur.

**4.4. Compatibility**

4.4.1. Function on major browsers across desktop devices.

4.4.2. Adapt to different screen resolutions.

**5. User Interface**

**5.1. Canvas**

5.1.1. Provide central interactive area for graph manipulation.

5.1.2. Support zoom, pan, and grid functionality.

5.1.3. Enable selection of multiple graph elements.

**5.2. Toolbars and Panels**

**5.2.1. Main Toolbar**

5.2.1.1. Include tools for node/edge creation, selection, and deletion.

5.2.1.2. Provide undo/redo controls.

**5.2.2. Properties Panel**

5.2.2.1. Edit properties of selected nodes and edges.

5.2.2.2. Customize visual and data attributes.

**5.2.3. Algorithm Panel**

5.2.3.1. Select algorithms and configure parameters.

5.2.3.2. Control algorithm execution and view results.

**5.3. Menus**

5.3.1. File Menu: Create, save, load, import, and export graphs.

5.3.2. Edit Menu: Cut, copy, paste, and select operations.

5.3.3. View Menu: Adjust zoom, grid, and panel visibility.

5.3.4. Algorithm Menu: Access and configure available algorithms.

**6. Data Structure**

**6.1. Node Data**

6.1.1. Unique ID, coordinates, label, and visual properties.

**6.2. Edge Data**

6.2.1. Unique ID, source/target nodes, weight, directedness, and visual properties.

**6.3. Graph Metadata**

6.3.1. Name, creation date, modification date, and graph type.

**6.4. Storage Format**

6.4.1. JSON structure containing graph, node, and edge data.

6.4.2. Support for common export formats (JSON, PNG).

**7. Appendix**

**7.1. Algorithm Specifications**

7.1.1. BFS: O(V+E) time complexity, level-order traversal.

7.1.2. DFS: O(V+E) time complexity, depth-priority traversal.

7.1.3. Prim's: O(E log V) time complexity, greedy MST algorithm.

7.1.4. Kruskal's: O(E log E) time complexity, union-find MST algorithm.

7.1.5. Dijkstra's: O(E log V) time complexity, positive-weight shortest paths.

7.1.6. Bellman-Ford: O(V×E) time complexity, supports negative weights.

**7.2. User Customization Reference**

7.2.1. Node customization options: colour, size, shape, label, border, icons.

7.2.2. Edge customization options: colour, thickness, pattern, arrow style, label.

7.2.3. Layout customization options: spacing, orientation, clustering.

7.2.4. Algorithm visualization options: colours, speed, detail level, annotations.