### **1. Functional Test Cases**

#### **1.1. Load JSON File**

* **Objective**: Verify the graph viewer can load and parse a JSON file with nodes and edges.
* **Steps**:
  + Provide a valid JSON file containing nodes and edges.
  + Ensure the graph is rendered in the #mynetwork container.
* **Expected Result**: Nodes and edges appear correctly as per the JSON file structure.

#### **1.2. Render Nodes**

* **Objective**: Check that nodes are displayed with the correct labels and group attributes.
* **Steps**:
  + Load a JSON file with various nodes having different labels.
  + Verify that node labels match the label property.
* **Expected Result**: Nodes are displayed with accurate labels and visual attributes.

#### **1.3. Render Edges**

* **Objective**: Verify that edges are rendered with proper connections between nodes.
* **Steps**:
  + Load a JSON file with predefined edges.
  + Verify that each edge connects the correct from and to nodes.
* **Expected Result**: All edges are correctly connected.

#### **1.4. Cluster by Domain**

* **Objective**: Test clustering functionality based on the domain attribute.
* **Steps**:
  + Click the "Cluster by Domain" button.
  + Verify that nodes with the same domain are grouped into clusters.
* **Expected Result**: Nodes sharing the same domain are grouped into clearly labeled clusters.

#### **1.5. Recluster All**

* **Objective**: Check the reclustering functionality for existing clusters.
* **Steps**:
  + Perform clustering by domain.
  + Click the "Re-cluster All" button.
* **Expected Result**: Clusters are recreated as per their original domain grouping.

#### **1.6. Clear Clusters**

* **Objective**: Verify the ability to clear all clusters and return to the default node layout.
* **Steps**:
  + Perform clustering.
  + Click the "Clear Clusters" button.
* **Expected Result**: All clusters are removed, and individual nodes are restored.

#### **1.7. Double-Click to Expand Clusters**

* **Objective**: Check that double-clicking on a cluster expands it into individual nodes.
* **Steps**:
  + Perform clustering by domain.
  + Double-click on any cluster node.
* **Expected Result**: The selected cluster expands, showing its constituent nodes.

### **2. Validation Test Cases**

#### **2.1. Invalid JSON Structure**

* **Objective**: Verify the behavior when an invalid JSON file is loaded.
* **Steps**:
  + Load a JSON file missing nodes or edges.
  + Observe the application’s response.
* **Expected Result**: An error message is displayed, or the graph viewer does not render.

#### **2.2. Empty JSON File**

* **Objective**: Test behavior when an empty JSON file is loaded.
* **Steps**:
  + Load a JSON file with no nodes or edges.
  + Observe the application’s response.
* **Expected Result**: The viewer remains blank without errors.

#### **2.3. Large Graphs**

* **Objective**: Validate performance and rendering for large graphs.
* **Steps**:
  + Load a JSON file with 1,000+ nodes and edges.
  + Observe rendering performance and responsiveness.
* **Expected Result**: The viewer loads the graph without lag or crashes.

#### **2.4. Node and Edge Attributes**

* **Objective**: Ensure additional attributes (e.g., domain, color) are handled correctly.
* **Steps**:
  + Load a JSON file with nodes having additional attributes like domain and color.
  + Verify that clustering and visual attributes are applied correctly.
* **Expected Result**: Attributes like domain influence clustering, and colors reflect the cluster.

### **3. UI/UX Test Cases**

#### **3.1. Button Functionality**

* **Objective**: Ensure that buttons ("Cluster by Domain," "Re-cluster All," "Clear Clusters") trigger their respective actions.
* **Steps**:
  + Click each button and observe the graph’s behavior.
* **Expected Result**: Each button performs the corresponding function.

#### **3.2. Layout Consistency**

* **Objective**: Verify that the sidebar and graph viewer maintain a consistent layout.
* **Steps**:
  + Resize the browser window and reload the graph.
* **Expected Result**: The layout remains intact and readable.

### **4. Non-Functional Test Cases**

#### **4.1. Performance**

* **Objective**: Ensure the graph viewer handles large graphs without performance degradation.
* **Steps**:
  + Load increasingly large JSON files (500, 1,000, 5,000 nodes).
* **Expected Result**: Smooth interaction and rendering.

#### **4.2. Cross-Browser Compatibility**

* **Objective**: Validate compatibility across different browsers.
* **Steps**:
  + Load the graph viewer in Chrome, Firefox, Edge, and Safari.
* **Expected Result**: The viewer functions identically in all browsers.

#### **4.3. Accessibility**

* **Objective**: Test for compliance with accessibility standards (e.g., WCAG).
* **Steps**:
  + Use screen readers and keyboard navigation.
* **Expected Result**: All actions are accessible via keyboard, and labels are screen-reader friendly.

### **5. Test Environment**

* **Browsers**: Chrome, Firefox, Edge, Safari.
* **OS**: Windows 10/11, macOS, Linux.
* **Device**: Desktop, Laptop.