Q2 [15 points] Force-directed graph layout

Goal	Create a network graph shows relationships between games in D3. Use interactive features like pinning nodes to give the viewer some control over the visualization.
Technology	D3 Version 5 (included in the lib folder) Chrome v131.0.0 (or higher): the browser for grading your code Python http server (for local testing)
Allowed Libraries	D3 library is provided to you in the **lib** folder. You must **NOT** use any D3 libraries (d3*.js) other than the ones provided. On Gradescope, these libraries are provided for you in the auto-grading environment.
Deliverables	[Gradescope] Q2**`.(html/js/css)`**: The HTML, JavaScript, CSS to render the graph. Do not include the D3 libraries or board_games.csv dataset.

You will experiment with many aspects of D3 for graph visualization. To help you get started, we have provided the Q2.html file (in the Q2 folder) and an undirected graph dataset of boardgames, board_games.csv file (in the Q2 folder). The dataset for this question was inspired by a Reddit post about visualizing boardgames as a network, where the author calculates the similarity between board games based on categories and game mechanics where the edge value between each board game (node) is the total weighted similarity index. This dataset has been modified and simplified for this question and does not fully represent actual data found from the post. The provided Q2.html file will display a graph (network) in a web browser. The goal of this question is for you to experiment with the visual styling of this graph to make a more meaningful representation of the data. Here is a helpful resource (about graph layout) for this question.

Note: You can submit a single Q2.html that contains all the css and js components; or you can split Q2.html into Q2.html, Q2.css, and Q2.js.

1. **[2 points] Adding node labels**: Modify Q2.html to show the node label (the node name, e.g., the source) at the **top right** of each node in **bold**. If a node is dragged, its label must move with it.

2. **[3 points] Styling edges**: Style the edges based on the "value" field in the links array:
   1. If the value of the edge is equal to 0 (similar), the edge should be gray, thick, and **solid** (The dashed line with zero gap is not considered as solid).
   2. If the value of the edge is equal to 1 (not similar), the edge should be green, thin, and **dashed**.

3. **[3 points] Scaling nodes:**

   a. **[1.5 points]** Scale the radius of each node in the graph based on the degree of the node (you may try linear or squared scale, but you are not limited to these choices).

   Note: Regardless of which scale you decide to use, you should avoid extreme node sizes, which will likely lead to low-quality visualization (e.g., nodes that are mere points, barely visible, or of huge sizes with overlaps).

   Note: D3 v5 does not support d.weight (which was the typical approach to obtain node degree in D3 v3). You may need to calculate node degrees yourself. Example relevant approach is here.

   b. **[1.5 points]** The degree of each node should be represented by varying colors. Pick a meaningful color scheme (hint: color gradients). There should be at least 3 color gradations and it must be visually evident that the nodes with a higher degree use darker/deeper colors and the nodes with lower degrees use lighter colors. You can find example color gradients at Color Brewer.

4. **[6 points] Pinning nodes**:
   a. **[2 points]** Modify the code so that dragging a node will fix (i.e., "pin") the node's position such that it will not be modified by the graph layout algorithm (Note: pinned nodes can be further dragged around by the user. Additionally, pinning a node should not affect the free movement of the other nodes). Node pinning is an effective interaction technique to help users spatially organize nodes during graph exploration. The D3 API for pinning nodes has evolved over time. We recommend reading this post when you work on this sub-question.
   b. **[1 points]** Mark pinned nodes to visually distinguish them from unpinned nodes, i.e., show pinned nodes in a different color.
   c. **[3 points]** Double clicking a pinned node should unpin (unfreeze) its position and unmark it. When a node is no longer pinned, it should move freely again.

IMPORTANT:

1. To pass autograder consistently for part 1 (which tests if a dragged node becomes pinned and retains its position), you may need to increase the radius of highly weighted nodes and reduce their label sizes, so that the nodes can be more easily detected by the autograder's webdriver mouse cursor.
2. To avoid timeout errors on Gradescope, complete the double click function in part 3 before submitting.
3. If you receive timeout messages for all parts and your code works locally on your computer, verify that you are indeed using the appropriate ids provided in the "add the nodes" section in the skeleton code.
4. D3 v5 does not support the d.fixed method (it was deprecated after D3 v3). For our purposes, it is used as a Boolean value to indicate whether a node has been pinned or not.

5. **[1 points] Add GT username:** Add your Georgia Tech username (usually includes a mix of letters and numbers, e.g., gburdell3) to the top right corner of the force-directed graph (see example image). The GT username must be a `<text>` element having the `id: "credit"`

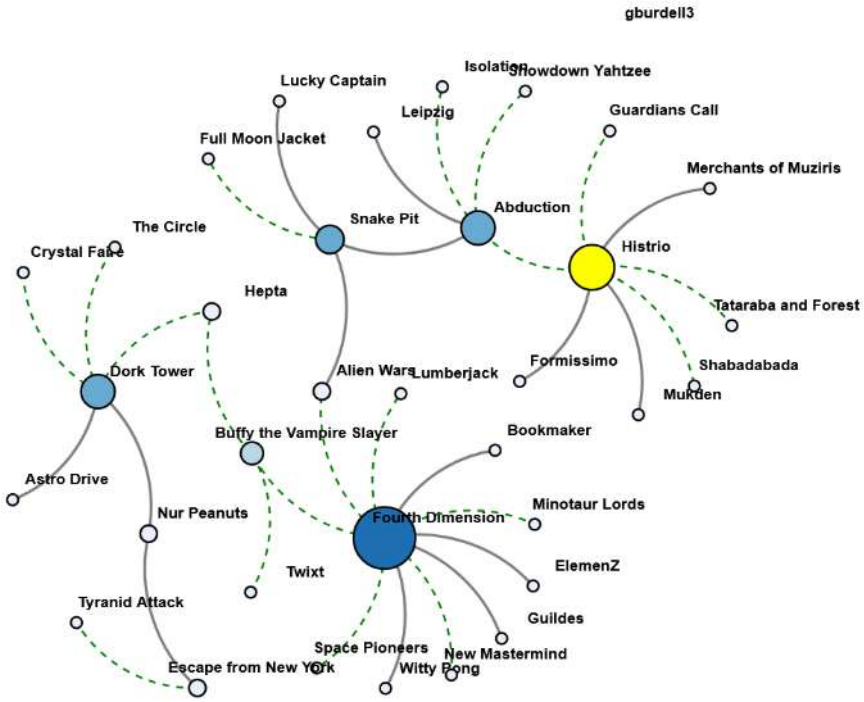

Figure 2: Example of Visualization with pinned node (yellow). Your chart may appear different and can earn full credit if it meets all the stated requirements.