INPUT AND OUTPUT VISUALIZATION

As part of my summer internship at the GISE Lab, IIT Bombay, I was assigned the task of creating the input and output visualizations for an Evacuation tool developed here.

As part of the input and output visualization, the following tasks can be carried out:

- Draw nodes and connect them using edges using mouse click and mouse drag coupled with pressing the shift key.
- The details can be stored in the database by clicking on the nodes and edges and filling out the details in the form.
- The user specified data can be edited if the user has the permission to do so.
- Scenarios can be created and details can be given.
- Scenarios can be edited if the privilege exists.

Once the details are collected and stored, the following visualizations can be seen:

- See the source nodes based on floor.
- When a source node is clicked, then the path to the destination can be seen.
- See the destination nodes on a floor.
- When a destination node is clicked, then a table with all the sources reaching this destination is generated. When clicked on a row in the table, the corresponding floor is loaded with the sources being highlighted.
- See the constraint edges floor wise
- The floor wise graph can be loaded and only those nodes and edges being used in the evacuation process are highlighted separately.

Draw nodes and edges:

Jsp files involved**:**

- ➔ OriginalInputGraph.jsp
- ➔ NodeForm.jsp
- ➔ NodeAddition.jsp
- ➔ EdgeForm.jsp
- ➔ EdgeAddition.jsp
- ➔ Check.jsp
- ➔ checkE.jsp

References of prototype specification and object creation have been taken from https://github.com/cjrd/directed-graph-creator

How to create a node?

Press shift Key and a right click would create a node which is a circle and an a default node size. The node can be translated and its position can be changed. Upon double click on a node, a form with pre assigned node id and the position of x and position of y is loaded. The user is required to specify the details and upon pressing save, the details are stored in the database.

How to create an edge?

One needs to create at least 2 nodes to draw an edge between them. Press shift Key and drag the mouse between two nodes in order to create an edge whose source node is the first pressed node and target node is the node on which mouse is released. The node can be translated and its position can be changed and all the attached edges also move accordingly. Upon double click on an edge, a form with pre assigned edge id and the node id of the source and the node id of the target node is loaded. The user is required to specify the details and upon pressing save, the details are stored in the database.

How to delete a node?

A node can be clicked and selected. Then upon pressing delete key or backspace key, control shifts to an intermediate jsp check.jsp which checks if the node is already in the database or not. The node and the edges attached to it are removed from the screen. If the node already exists in the database (outcome of check.jsp), then the node and the corresponding edges connected to the node are removed from the database.

How to delete an edge?

An edge can be clicked and selected. Then upon pressing delete key or backspace key, control shifts to an intermediate jsp checkE.jsp which checks if the edge is already in the database or not. The edge is removed from the screen. If the edge already exists in the database(outcome of checkE.jsp), then the edge is removed from the database.

A proper Database connection is required. This tool uses PostgreSQL.

Fig1: Image for reference purpose.

Retrieve data with edit/ delete:

Jsp files involved**:**

- ➔ Building-floordata.jsp
- ➔ NodeForm.jsp
- ➔ NodeAddition.jsp
- ➔ EdgeForm.jsp
- ➔ EdgeAddition.jsp
- ➔ NodeDetails.jsp
- ➔ EdgeDetails.jsp
- ➔ Check.jsp
- ➔ checkE.jsp

How to create a new node?

Press shift Key and a right click would create a node which is a circle and an a default node size. The node can be translated and its position can be changed. Upon double click on a node, a form with pre assigned node id and the position of x and position of y is loaded. The user is required to specify the details and upon pressing save, the details are stored in the database.

How to create a new edge?

One needs to create at least 2 nodes to draw an edge between them. Press shift Key and drag the mouse between two nodes in order to create an edge whose source node is the first pressed node and target node is the node on which mouse is released. The node can be translated and its position can be changed and all the attached edges also move accordingly. Upon double click on an edge, a form with pre assigned edge id and the node id of the source and the node id of the target node is loaded. The user is required to specify the details and upon pressing save, the details are stored in the database.

How to delete a node?

A node can be clicked and selected. Then upon pressing delete key or backspace key, control shifts to an intermediate jsp check.jsp which checks if the node is already in the database or not. The node and the edges attached to it are removed from the screen. If the node already exists in the database (outcome of check.jsp), then the node and the corresponding edges

connected to the node are removed from the database.

How to delete an edge?

An edge can be clicked and selected. Then upon pressing delete key or backspace key, control shifts to an intermediate jsp checkE.jsp which checks if the edge is already in the database or not. The edge is removed from the screen. If the edge already exists in the database(outcome of checkE.jsp), then the edge is removed from the database.

How to edit a node?

A popup of NodeDetails.jsp loaded with the details of the node opens up. Then the user can change the details of the nodes and save it. The corresponding changes are saved in the database.

How to edit an edge?

A popup of EdgeDetails.jsp loaded with the details of the edge opens up. Then the user can change the details of the edges and save it. The corresponding changes are saved in the database.

A proper Database connection is required. This tool uses PostgreSQL.

Retrieve data without edit/ delete:

Jsp files involved **:**

- ➔ Building-floordata.jsp
- ➔ NodeDetails.jsp
- ➔ EdgeDetails.jsp

Upon clicking an edge or a node, a popup opens with the details of the clicked item i.e node or an edge.

How to create a new node?
User cannot create a new node. This is due to the absence of privilege to edit a building not created by the user.

How to create a new edge?
User cannot create a new edge. This is due to the absence of privilege to edit a building not created by the user.

How to delete a node?
User cannot delete a node. This is due to the absence of privilege to edit a building not created by the user.

How to delete an edge?
User cannot delete an edge. This is due to the absence of privilege to edit a building not created by the user.

How to edit a node?
User cannot edit a node. This is due to the absence of privilege to edit a building not created by the user.

How to edit an edge?
User cannot edit a node. This is due to the absence of privilege to edit a building not created by the user.

A proper Database connection is required. This tool uses PostgreSQL.

Path from Source to Destination:

Jsp files involved**:**

- ➔ Click-GetPath.jsp
- ➔ ForPath.jsp
- ➔ StoD.jsp

How to see the path from a source to its destination?

The nodes and edges are loaded from the database and displayed. Only those nodes and edges that are involved in the evacuation plan are being shown in black color with the remaining ones being shown in gray. The sources, if any are shown in red color. These can only be clicked in order to see path visualization.

How does the visualization look like?

Upon clicking a source, ForPath.jsp which is an intermediate jsp is executed and the list of the node ids and edge ids are collected and then StoD.jsp pops up showing the path.

In the visualization of a node on one floor whose destination is in another floor, then the initial floor appears with the path being shown (red is the start node of the path of that floor and green is the end node of the path of that floor). Automatically after 3 seconds, the next floor appears with the path being highlighted. The final window appears showing the destination.

Note: One can change the time gap between adjacent windows.

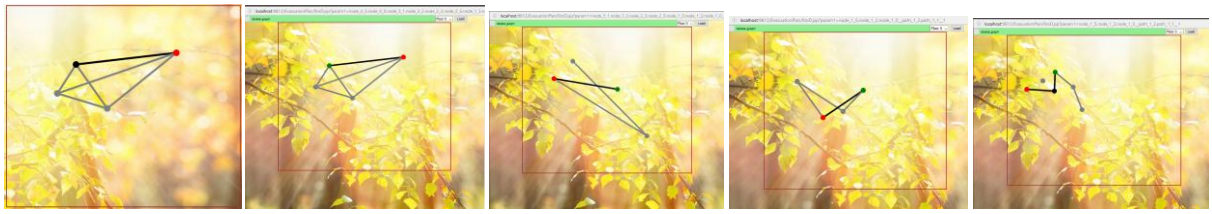

Fig1: Showing only the graphs generated for reference only.
When red node of the initial picture is clicked, the sub sequential screens appear with a lag of 3 seconds.

Sources reaching to a Destination:

Jsp files involved**:**

- ➔ ShowDestinations.jsp
- ➔ DestTable.jsp
- ➔ FloorSources.jsp

How does the visualization look like?

The nodes and edges are loaded from the database and displayed. Only those nodes and edges that are involved in the evacuation plan are being shown in black color with the remaining ones being shown in gray. The destinations, if any are shown in green color. These can only be clicked in order to see all sources ending up at this.

Upon clicking a destination, DestTable.jsp opens and the list of the floor and corresponding node ids which are sources and ending up at this destination are shown.

Upon clicking a row with floor number and source list, the screen changes to show that floor and the corresponding nodes and edges and ONLY those nodes which are the source nodes are highlighted in red.

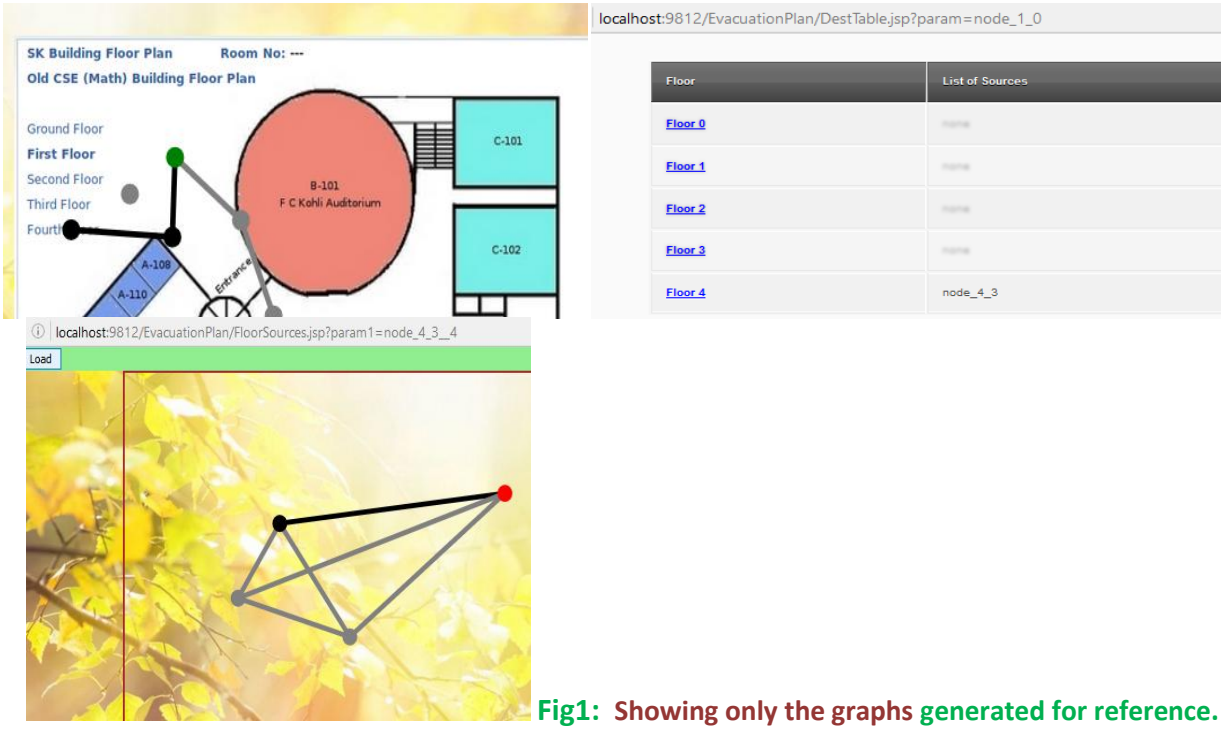

Fig1: Showing only the graphs generated for reference.

When green node of the initial picture is clicked, the sub sequential screens appear with a table containing all the source nodes of different floors. When a particular floor is clicked, the screen with that floor is loaded along with the sources reaching to the original destination.

See the Constraint Edges:

Jsp files involved**:**

➔ ShowConstraintEdges.jsp

How does the visualization look like?

The nodes and edges corresponding to that floor are loaded from the database and displayed. Only those nodes and edges that are involved in the evacuation plan are being shown in black color with the remaining ones being shown in gray. The constraint edges, if any are shown in red color.

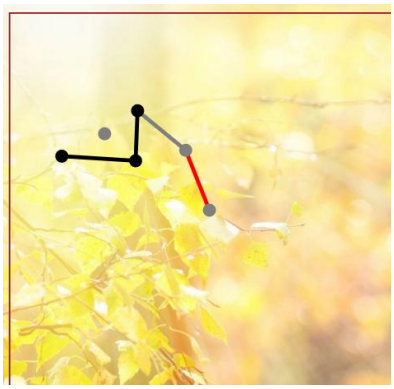

Fig1: Showing only the graph generated for reference.
When a particular floor is chosen, then all the nodes and edges are loaded. Only those edges and nodes which are involved in evacuation process are colored black and the remaining are gray. Those edges which are constraint edges are colored red.

D3 LIBRARY:

The entire visualization is built using d3 library and JavaScript

D3.js is a JavaScript library for manipulating documents based on data. **D3** helps you bring data to life using HTML, SVG, and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

D3.js is written in JavaScript and uses a functional style which means you can reuse code and add specific functions to your heart's content. Which means it is as powerful as you want to make it. How you chose to style, manipulate, and make interactive the data is up to you.

D3.js is a JavaScript library added to the front-end of your web application. Your back-end (the server) will generate the necessary data. The part of the application the users interact with (the front-end) will use D3.js.

You should use D3.js when your webpage is interacting with data. D3 stands for **D**ata **D**riven **D**ocuments. We will explore D3.js for it's graphing capabilities. We will not explore the myriad ways you can use data to in your web pages in non-graphical ways.

For further reading, please refer the specified link.
https://www.dashingd3js.com/why-build-with-d3js

YARA MANISHA
Second Year Undergraduate
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Summer Intern GISE Lab
Department of Computer Science and Engineering
Indian Institute of Technology, Bombay