**I. Dependencies**

- NodeJS (<https://nodejs.org/en/>)

- ExpressJS - Web Framework (<http://expressjs.com/>)

- MySQL - Database (<https://www.mysql.com/>)

- SequelizeJS - ORM (<http://docs.sequelizejs.com/en/latest/>)

**II. Setup and install**

A. Files Setup

1. Clone the repository:

git clone <https://github.com/madca03/RESE2NSE-Graph-Visualization.git>

2. Inside the folder, run “npm install” to install the node modules.

B. Database Setup

- Execute in terminal

mysql -u root -e "CREATE USER 'rese2nse'@'localhost' IDENTIFIED BY 'rese2nse';GRANT ALL PRIVILEGES ON graph.\* TO 'rese2nse'@'localhost';CREATE DATABASE graph;" -p

C. Run Application

npm start

**III. File Structure**

A. config - This contains the configuration settings for the database.

B. migrations - The files in here are used to create the database tables used for the application.

C. models - Files here are used to define associations among the database tables.

D. node\_modules - auto-generated files from “npm install” containing Node dependencies used

in the application

E. public - This is the assets folder. It contains the CSS stylesheets, images, fonts, and front-

end javascript files.

- Main front-end javascript file -> public/javascripts/graph.js

F. routes - This contains the server-side javascript files. The accessible routes on the browser

and the routes used by the AJAX calls are defined here.

G. seeders - Files here are used to populate the database with random test data.

H. view - HTML files

**IV. Alter Database Schema and Data**

A. Run migration files to create DB tables

node\_modules/.bin/sequelize db:migrate

(to undo migration, run node\_modules/.bin/sequelize db:migrate:undo:all)

B. Run seed files to generate random data in the DB.

node\_modules/.bin/sequelize db:seed:all

(to undo seed, run node\_modules/.bin/sequelize db:seed:undo:all)

**V. Application Features**

A. Separate graph display for guest and admin users

1. Guest user graph display

- When a guest user accesses the graph visualization page (currently it’s in “localhost:3000/graph”), the user will only see the nodes that are already fixed in position. Furthermore, a link between two nodes is only displayed if both nodes are fixed in position. (see the query in “routes/index.js” under the “/nodes\_for\_display” route).

2. Admin user graph display

- This graph display is used for editing the nodes’ position and thus all nodes (fixed and not fixed) are displayed on the graph. (query for this is in “routes/index.js” under the “/nodes/:floor\_number” route).

B. Edit and Save node position

- An admin user can edit the nodes’ position via drag and drop. After editing the nodes’ position and clicking the save button, the front-end javascript (public/javascripts/graph.js) will get all of the nodes with a fixed property set to “true” then it will issue an AJAX POST request to the “/nodes/update” route in “routes/index.js”. This route will then parse the JSON data containing the fixed nodes and it will send an update query to the DB for each fixed node.

C. Real Time Update of links

- The main front-end javascript file (public/javascripts/graph.js) is set to update the graph display for every T seconds via “setInterval” function. For each update, the front-end javascript gets the updated links by making an AJAX call to “/nodes\_for\_display” route. After obtaining the updated links, it will remove all of the SVG path elements related to the previous set of links and these old SVG path elements will be replaced by a new set of SVG path elements that now correspond to the set of updated links.

- To test this feature, go to the graph visualization page at “localhost:3000/graph”. Edit some of the nodes’ position so that there will be nodes and links displayed on the guest user graph display. In the terminal, run the seed file for the edges:

node\_modules/.bin/sequelize db:seed --seed 20160202214947-edges.js

(“20160202214947-edges.js” is in the seeders folder)

This will generate a new set of links in the DB and the effect of changing the links in the DB can be seen immediately on the browser after around T seconds.

D. Curvature of links

- The links dataset obtained from the AJAX call are mapped to a set of SVG path elements. These SVG path elements use the quadratic Bezier curve (https://developer.mozilla.org/en-US/docs/Web/SVG/Tutorial/Paths#Bezier\_Curves) to draw the arc from source node to target node. The control point used is located at X% (currently set to 40%, see “update\_links” function inside “draw\_graph\_for\_display” function in “public/javascripts/graph.js”) of the distance between the source/target node to the midpoint from the midpoint. See figure below. If the computed control point is outside the SVG stage, its x or y coordinates is changed so that the control point will be inside the SVG stage.

