Full-stack Software Engineer Interview Exercise

- Follow the directions below to build two parts
    - Back end: a web service
    - Front end: single page web app
- Create a `README.md` that explains how to run the project
- Check all of the above into a new public repository in GitHub
- Email the repo link to your HR contact!

Problem

At PeopleNet we're all about the transportation industry. For this exercise we would like to help route a truck from point A to point B on a fictional "map".

You will be creating a full stack solution that allows the user to input a text based maze on the front end. The back end should solve this maze and find the shortest route from the start to the finish. Here's a hint, take a look at Breadth-first search (BFS).

The front end should provide visualizations of the maze itself, and the final solution path, including an output of the number of steps it took to solve that particular maze. Don't be afraid to be creative with the visualizations.

The project should include all the necessary build tasks to run locally, so that we can try it out.

A valid maze may contain the following types of characters:

- `.` represents an open road
- `#` represents a blocked road
- `A` represents the starting point

- `B` represents the destination point

Anything outside the bounds of the array should be considered a wall. In addition, you may only move in horizontal or vertical directions. Diagonal movements are not allowed.

Example input maze:

```
##########
#A...#...#
#.#.##.#.#
#.#.##.#.#
#.#....#B#
#.#.##.#.#
#....#...#
##########
```

Example simplified visual of the solution, using `@` to represent the correct path. The shortest number of steps in this case is 14. An ASCII representation like this is not recommended however for the final solution.

```
##########
#A@@.#...#
#.#@##.#.#
#.#@##.#.#
#.#@@@@#B#
#.#.##@#@#
#....#@@@#
##########
```

Goal

Your solution should be able to solve the included mazes `maze1.txt` (pictured above), `maze2.txt`, and `maze3.txt` correctly. Each maze solution should run in under a minute on reasonable hardware.

Suggested Technologies

You don't have to use these, but these are what we use and recommend.

- Java/Groovy, (Spring, Spring Boot)
- NodeJS (6+), (Express, Hapi)

- AngularJS (any version)
- React
- Material Design