Rose Rocket Coding Challenge - Driver Location Challenge

Due Date: Sunday, Jan 20, 2018 at 11:59PM EST
How to submit the challenge: Post your solution to GitHub and provide a link to your solution

Imagine there is a truck driver whose location we are interested in finding. The truck driver is located somewhere on a 200 x 200 grid.

There is a list of stops that the driver needs to visit along their route. A stop has the following properties:

- name: The name of the stop
- x: The x-coordinate position of the stop (integer from 0 - 200)
- y: The y-coordinate position of the stop (integer from 0 - 200)

The name of the stop will be an uppercase alphabet character (e.g. “A”, “B”, “C”, “D”, …). The driver must visit each stop along the route in ascending order of stop name (e.g. A -> B -> C -> D …). The provided stop names will be in the range A-Z and will not skip letters.

A straight line segment between two stops is referred to as a “leg”. Each leg has properties:

- startStop: The name of the starting stop
- endStop: The name of the ending stop
- speedLimit: The maximum speed limit that the driver can travel on this leg (integer from 1 - 200 in kilometres per hour)
- legID: Identifier for the leg. The legID will be the concatenation of the startStop name and the endStop name. E.g. The legID for startStop=”A” and endStop=”B” is “AB”.

The driver’s current position is specified by:

- activeLegID: The ID of the leg that the driver is currently driving on
- legProgress: The driver’s percentage progress (integer from 0 - 100) along the current leg (line segment).

We will provide the driver’s initial position, as well as the list of stops and legs that the driver has to visit (in JSON format) at the end of this specification.

Your task is to:

1. Add a REST API to retrieve the given list of stops and legs from server.
    - GET /legs
    - GET /stops
2. Visualize the stops
3. Add a REST API to get the driver’s current position.
    - GET /driver
4. Visualize the position of the driver

5. Visualize and highlight the completed section of the leg where the driver is, as well as completed legs.
6. Add a form to change the driver's current position.
    - Active leg must be selected via a dropdown menu
    - Percentage progress must be specified via a textbox
7. Add a REST API to update the driver's current position
    - PUT /driver
8. Have the visualizations in #4 and #5 reflect the updates to the driver's current position

Technology:

- Frontend: We are looking for React (with or without Redux) implementation.
- Backend: Any technology would be sufficient but we suggest to keep things simple. We recommend using Node.js with a basic in-memory array instead of a database. We encourage use of websockets for real-time modifications but any other solutions are welcome as well and will be considered as a proper solution.

Minimal Requirements:

- README.md: clear step-by-step instruction of how to run the app
- FEATURES.md: a list of extra improvements (or bonuses) available on your page for us to test and to make a note
- A fully working page without any bugs
- Clean code implementation and organization
- Attention to detail, good usability of the tool as well as functionality and edge case scenarios
- Good and intuitive documentation which describe step by step of how to build, run and test the app

Bonus 1

Suppose we have a new type of driver called a "bonus driver". The bonus driver's location is specified by properties:

- x: The x-coordinate position of the bonus driver on the map (integer from 0 - 200)
- y: The y-coordinate position of the bonus driver on the map (integer from 0 - 200)

Your task is:

1. Add a form where you can enter in the (x, y) coordinates of the bonus driver.
2. Add a REST API to update the bonus driver's current location
    a. PUT /bonusdriver
3. Add a REST API to get the bonus driver's current location
    a. GET /bonusdriver
4. Visualize the bonus driver and draw a line between the bonus driver's (x, y) coordinates and the closest stop. Also highlight the remaining path from the closest stop to the final stop.

Bonus 2
Given that each leg has a speed limit, calculate how long it will take the driver to complete the entire path (all of the legs). Also, calculate how much time is left for the driver based on their current position. Display both of these values somewhere on the page.

Bonus 3
Replace the percentage textbox with a slider and have the visualization update in realtime when you move the slider.

Additional Bonus Ideas:
- Error-handling for invalid input
- Using persistent data storage like MongoDB / Redis, etc
- Making the page aesthetically pleasing and user-friendly
- Unit tests

Initial Data
Driver location

```
{
	"activeLegID": "FG",
	"legProgress": "33"
}
```

Stops

```
[{
		"name": "A",
		"x": 20,
		"y": 10
	},
	{
		"name": "B",
		"x": 20,
		"y": 20
	},
	{
		"name": "C",
		"x": 25,
		"y": 30
	},
	{
		"name": "D",
		"x": 25,
		"y": 80
	},
	{
		"name": "E",
		"x": 30,
```

```
            "y": 100
      },
      {
            "name": "F",
            "x": 35,
            "y": 80
      },
      {
            "name": "G",
            "x": 35,
            "y": 30
      },
      {
            "name": "H",
            "x": 40,
            "y": 20
      },
      {
            "name": "I",
            "x": 40,
            "y": 10
      },
      {
            "name": "J",
            "x": 35,
            "y": 15
      },
      {
            "name": "K",
            "x": 25,
            "y": 15
      },
      {
            "name": "L",
            "x": 20,
            "y": 10
      }
]
```


Legs


```
[{
            "startStop": "A",
            "endStop": "B",
            "speedLimit": 100,
            "legID": "AB"
      },
      {
            "startStop": "B",
            "endStop": "C",
            "speedLimit": 60,
            "legID": "BC"
```

```
	},
	{
		"startStop": "C",
		"endStop": "D",
		"speedLimit": 80,
		"legID": "CD"
	},
	{
		"startStop": "D",
		"endStop": "E",
		"speedLimit": 120,
		"legID": "DE"
	},
	{
		"startStop": "E",
		"endStop": "F",
		"speedLimit": 40,
		"legID": "EF"
	},
	{
		"startStop": "F",
		"endStop": "G",
		"speedLimit": 40,
		"legID": "FG"
	},
	{
		"startStop": "G",
		"endStop": "H",
		"speedLimit": 100,
		"legID": "GH"
	},
	{
		"startStop": "H",
		"endStop": "I",
		"speedLimit": 100,
		"legID": "HI"
	},
	{
		"startStop": "I",
		"endStop": "J",
		"speedLimit": 50,
		"legID": "IJ"
	},
	{
		"startStop": "J",
		"endStop": "K",
		"speedLimit": 100,
		"legID": "JK"
	},
	{
		"startStop": "K",
		"endStop": "L",
		"speedLimit": 60,
```

```
            "legID": "KL"
      }
]
```

Bonus Driver Location

```
{
      "x": 50,
      "y": 55
}
```