Rose Rocket Coding Challenge

Due date
Jan 24th 23:59:59 EST

How to submit the challenge

Post your solution to GitHub and provide a link to your solution by emailing it to Jade, at **jade.g@roserocket.com with the subject line: "Rose Rocket coding challenge - Feb 2021"**

Problem overview

In logistics, a dispatcher is someone who is responsible for organizing the movement of resources and personnel. In trucking, a dispatcher oversees the tasks that drivers should complete.

As a dispatcher, I need a basic application to manage the tasks of my drivers to keep my organization running smoothly and my customers happy! This means keeping track of *where, when, and what* my drivers will be doing.

For this exercise, we'll assume that every movement will consist of:

- **Start Location Latitude/Longitude**: Where and when the driver picks something up
- **End Location Latitude/Longitude**: Where and when the driver drops something off
- **Freight Description**: Arbitrary string that describes the freight that is moved

Problem specifications

Part A: Dispatcher/Driver Movement Map

Implement an application view where a **single** dispatcher can create and track all movements of a **single** driver on the map for every good that needs to be picked up and then dropped off.

1. As a dispatcher, I should be able to *create* new movements for arbitrary goods that should be picked up at a *Start Location* and dropped off at the *End Location*.
2. Dispatchers should be able to see the list of movements and the description of freight.
3. Every movement should be represented on the map with 2 points connected with a line. (for simplicity purposes you could represent it as a straight line)
4. The app should not allow dispatchers to create identical movements (*Start Location, End Location, and Freight Description*) in a user-friendly manner.

5. Dispatchers should be able to *update* existing movements' *Start Location, End Location, and Freight Description*. The app should confirm with a dispatcher If the updated movement is identical to an existing one.
6. Dispatchers should be able to *delete* movements.
7. As a dispatcher, I should be able to make a clear connection between the list of the available movements and their visual representation on the map (examples: through color or through labels or by other means).

Part B: Creating a Driver Route

Example uses Cities but it can be implemented with lat/long instead.

1. As a dispatcher, I want to be able to generate a *Driver Route* that consists of all available locations that need to be visited.
    a. Please Note: that it does not make sense for a driver to visit *End Location* prior *to Start Location* for a particular freight.
2. It is OK if a Driver Route starts and ends in the same city.
    a. Example: If there are just 2 available movements: [Toronto -> Montreal] and [Montreal -> Toronto] then the *Driver Route* should be [Toronto -> Montreal -> Toronto] or [Montreal -> Toronto -> Montreal].

Additional Bonus Ideas

Take the opportunity to wow us by adding additional features listed below, or coming up with your own.

- Make UI more user-friendly for dispatchers by expanding the movement model and allowing dispatchers to enter *City Name* along with *Lat/Lng* for movement *Start* and *End Locations*.
- Ability to see *Driver Route* on the map (either alongside with the movements or ability to toggle between *Movements* or *Driver Route* views)
- We want to be sure that the *Driver Route* is optimized. We do not want to see 2 or more same locations in a row if it could be optimized.
    - Example: If there are available movements that visit the same city: [Toronto -> Montreal] and [Montreal -> Ottawa] then the *Driver Route* should be [Toronto -> Montreal -> Ottawa] instead of [Toronto -> Montreal -> Montreal -> Ottawa].
- We want to be sure that the *Driver Route* has some logic to optimize for visiting the nearest cities first.
    - Example: If there are just 2 available movements: [Toronto -> Montreal] and [Scarborough -> Ottawa] then the driver route should start with [Toronto -> Scarborough -> …] or [Scarborough -> Toronto -> …] any other route would result in a *Driver Route* that would be highly unoptimized.
- Being able to visualize routes as polylines that follow the road network, instead of simple straight lines.

- Design code and project in such a way that it is easy for developers to make modifications on the routing algorithm or replace it completely with different implementation on demand.
- Making the page aesthetically pleasing and user-friendly
- Any extra ideas or improvements you can think of. Please document these in the FEATURES.md

Minimal Requirements

- README.md: clear step-by-step instructions on how to run the app. (Please keep any extra installation requirements and external apps/services to a minimum. We recommend asking your friend to see if it is easy to understand the instructions listed in the README file and if they work.)
- FEATURES.md: a list of extra improvements (or bonuses) available on your page for us to test and to make a note
- A working page without any obvious issues. (Please state in README.md if reviewers need to be aware of any incomplete work or anything which is not working as expected)
- Required to run on Linux/Unix

More info and suggestions

As long as the above criteria are met, you can implement any GUI/UX experience you think is appropriate.

User login/authentication is not required. Only one dispatcher will ever need to “log in” to manage their drivers.

You are welcome to use any libraries that you think would help you. However, please ensure that we can easily install and run your application.

You do not need to implement persistence! You will not need to use databases to implement the requirements. This will not only make it easier for you to complete this exercise, but it will also make it easier for us to test!

Good luck!