Mothership Backend Programming Challenge - 2019

Create a basic automatic dispatch system for some Mothership shipments! Your app will be able to read from a list of shipments that need to be picked up and a list of available drivers and their locations, and hit a live REST endpoint to dispatch these shipments to nearby drivers.

You are given the following:

- JSON file representing a list of drivers with their current locations
    - driverId is the key of each JSON object (1, 2, 3...)
- JSON file representing a list of 5 active shipments and their pickup locations
    - shipmentId is the key of each JSON object (65289023243, 3823958290...)

Create an app that runs a periodic task every 10 seconds that dispatches each of the 5 shipments to the three closest drivers to the shipment. Dispatch a shipment to a driver by hitting the dispatch REST endpoint, detailed below.

In each round of the dispatch task, these dispatches can be sent out in parallel or one after another. If none of the three drivers accept the dispatch for a shipment in that round, expand your radius to dispatch to the next 3 closest drivers in the next round.

Allow your periodic task to continue to run and repeat this process for a shipment until a dispatch for the shipment has been Accepted, or you've dispatched to all the available drivers.

Do not dispatch a single shipment to the same driver more than once. You may dispatch multiple shipments to the same driver, even if they have accepted the dispatch for a different shipment.

Please include unit tests to showcase your experience writing tests.

When completed, zip your project and send it to jeremy@mothership.com, or share via Dropbox or Google Drive.

Example Dispatch Lifecycle

Here's an example (using only 3 shipments instead of 5, and 8 drivers).

Round 1 of dispatches:
Shipment 1 dispatches to drivers 1, 2, and 3, the three closest drivers to Shipment 1.
- Driver 1 denies, Driver 2 accepts. Ignore Driver 3's response.
Shipment 2 dispatches to drivers 3, 5, and 8, the three closest drivers to Shipment 2.
- Driver 3, 5 and 8 all deny.
Shipment 3 dispatches to drivers 5, 1, and 2, the three closest drivers to Shipment 3.
- Driver 5, 1 and 2 all deny.

Round 2 of dispatches (10 seconds later)
Shipment 1 needs no action because it has already been accepted by Driver 2.
Shipment 2 dispatches to drivers 1, 6 and 2, the three closest remaining drivers to Shipment 2.
- Driver 1, 6 and 2 all deny.
Shipment 3 dispatches to drivers, 3, 4 and 6, the three closest remaining drivers to Shipment 3.
- Driver 3 and 4 deny, Driver 6 accepts.

Round 3 of dispatches (10 seconds later)
Shipment 1 needs no action because it has already been accepted by Driver 2.
Shipment 2 dispatches to drivers 4 and 7, the only drivers Shipment 2 hasn't been dispatched to yet.
- Driver 4 and 7 deny.
Shipment 3 needs no action because it has already been accepted by Driver 6.

Since Shipment 2 has no one else to dispatch to, and Shipment 1 and Shipment 3 have been accepted and need no further action, then your app can end here, and the task does not need to run any further.

In this example, your app will have finished running in 20 seconds through 3 rounds of dispatches.

Dispatch REST Endpoint

Create a dispatch for a shipment by hitting the following endpoint:
`POST http://challenge.shipwithbolt.com/driver/:driverId/dispatch`
Replace :driverId in the URL with the driverId you are dispatching to
Pass in as the body in JSON format:

```
{
   shipmentId: x (string)
}
```

If the driverId is invalid (does not exist in the driver JSON file), a 404 error will be returned.

If the shipmentId is missing or invalid (does not exist in the shipment JSON file), a 400 error will be returned.

If the driverId is valid, you will receive the following payload:

```
{
   response: 'Accepted' or 'Denied' (string),
   driverId: x (number),
   shipmentId: x (string)
}
```

The response will be random - **Accepted** 15% of the time, and **Denied** 85% of the time.

Notes:

- Feel free to use an existing formula or solution for determining proximity from a shipment to a driver. **You are not required nor expected to create this formula from scratch.**
- No time limit with this programming challenge - take your time and showcase your coding style and skills. We're looking for submissions that emulate the concepts and practices of what you would write in a **production codebase**.
- This project **should not take you more than 3 hours of development time**.
- If you find yourself spending more than 3 hours writing code, we love the effort, but you are probably scope-creeping or doing too much! For example, we had one candidate spin up his own API mirroring the functionality of our provided endpoint, and also a PostGIS database hosted on his dedicated server. Too much, too much.
- Use any programming language that you are comfortable with.

Feel free to reach out with any questions or feedback at jeremy@mothership.com.