COMP 120: Web Programming and Engineering

Lab: The Ride-Hailing Service, Part 3 on Week 5

Revisit Part 2 on Week 4

Overview

Last week, I gave you an API to request a ride: **`https://jordan-marsh.herokuapp.com/rides`**. This week, I am decomissioning the server that provided the API. Therefore, your map (from week 4) will no longer work. To make your map work again, you must recreate the server-side.

Instructions

You will recreate the server-side that will respond to your request for a ride. At minimum, your server must have one API route: **`POST /rides`**. If you do the bare minimum adhering to the basic requirements of **`POST /rides`**, you can earn up to 8.5 points out of 10 points. If you put in more work, you can earn more than 9 points.

Getting Started

Do the The Server Side using Node.js and Heroku lab first. That is a signficant starting point. It is recommended that you use Heroku to deploy your server side work.

Basic Requirements for `POST /rides`

1. Must return JSON
2. The required fields and *exact field names* for submission to this API are **`username`**, **`lat`**, and **`lng`**. If a submission is missing any one of the required fields, return the following JSON as the response: **`{"error":"Whoops, something is wrong with your data!"}`**
3. If all the required fields are submitted with request, API shall return a JSON array of locations of vehicles. A vehicle in the JSON array is an object with the mandatory keys: **`username`**, **`lat`**, **`lng`**, and **`created_at`** where **`lat`** and **`lng`** are numbers. Example output:
```
[{"_id":"5cdf411856e9c200042989d7","username":"JANET","lat":42.354951,"lng":-71.0509,"created_
05-17T23:17:44.427Z"},
{"_id":"5cf583aafbbfe80004456918","username":"mXfkjrFw","lat":42.3453,"lng":-71.0464,"created_
06-03T20:31:38.378Z"},
{"_id":"5cf583aafbbfe80004456919","username":"nZXB8ZHz","lat":42.3662,"lng":-71.0621,"created_
06-03T20:31:38.611Z"},
{"_id":"5cf583aafbbfe8000445691a","username":"Tkwu74WC","lat":42.3603,"lng":-71.0547,"created_
06-03T20:31:38.786Z"},
{"_id":"5cf583aafbbfe8000445691b","username":"5KWpnAJN","lat":42.3472,"lng":-71.0802,"created_
06-03T20:31:38.932Z"},
{"_id":"5cf583abfbbfe8000445691c","username":"uf5ZrXYw","lat":42.3663,"lng":-71.0544,"created_
06-03T20:31:39.077Z"},
{"_id":"5cf583acfbbfe8000445691d","username":"VMerzMH8","lat":42.3542,"lng":-71.0704,"created_
06-03T20:31:40.400Z"}]
```
4. A note about security: with the exception of the data requirements above, please note that I did not mention a thing about security or error handling in the requirements, including authentication. There is a reason for this, as you will see in the next lab.
5. Cross-Origin Resource Sharing must be enabled for **`POST /rides`**. Without this, you will encounter an error in the JavaScript console of your web browser when you load your map.

Once you have completed the basic requirements of the server, all you need to do is to change the URL **`https://jordan-marsh.herokuapp.com/rides`** to **`https://[YOUR_APP_IDENTIFIER_RANDOMLY_GENERATED].herokuapp.com/rides`** in your map and your map should work again. If your map does not work, or if JavaScript changes are required to your previous work, then something is wrong with your server.

Going Beyond

These items are written vaguely to mirror real-world style and to give you some flexibility.

1. The basic requirements for **`POST /rides`** do not include storing the ride request from a passenger into a database. Store the ride request data **`username`**, **`lat`**, and **`lng`** into a Postgres database. The nodepsqlapp example is a Node.js

+ Express + Postgres system. One requirement: **`lat`** and **`lng`** must be stored as floating point numbers.
2. If you store ride request data from a passenger (see above), then it only make sense to store data on vehicles as well. Build an API route, an HTTP POST route, for vehicles to "check-in" their availability to pick up passengers. Store the vehicle data **`username`**, **`lat`**, and **`lng`** into a Postgres database. The fields are consistent with ride requests from passengers.
3. Write a **`GET /passenger.json`** API route that returns a list of all passenger records for a given username as a JSON string if record(s) exist in database. The mandatory parameter for this API is **`username`**. If the **`username`** query parameter is empty, not provided, or no results found, return empty JSON list **`[]`**.
4. Write a **`GET /vehicle.json`** API route that returns a list of all vehicle records for a given username as a JSON string if record(s) exist in database. The mandatory parameter for this API is **`username`**. If the **`username`** query parameter is empty, not provided, or no results found, return empty JSON list **`[]`**.
5. Write a **`GET /`** route --home page, the root, the index. Accessing this on a web browser (e.g., **`https://[YOUR_APP_IDENTIFIER_RANDOMLY_GENERATED].herokuapp.com/`**) shall display list of all the vehicles and/or passengers in the database. Simply outputting JSON as the page is unacceptable; route must return HTML.
6. Build another web application + server with the same basic requirements except using a different programming language and framework (e.g., using Django or Flask for Python, Rails or Sinatra for Ruby). Assess the differences and similarities in your README.

References

1. The Express web framework API reference (official)
2. The **`nodepsqlapp`** example I wrote
3. Node.js + Express + Cross-Domain Scripting (Stack Overflow)
4. The Node.js + Heroku Lab
5. Getting Started with Node.js on Heroku - Provision a Database

Performance Optimization Requirements

Be sure to:

- Load CSS first, head section
- Minify CSS
- Move JavaScript includes and code to the bottom of the HTML before the closing body tag
- Minify JavaScript code

The **`README`** File

Every assignment shall include a **`README`** file that describes the work. This description must:

1. Identify what aspects of the work have been correctly implemented and what have not.
2. Identify anyone with whom you have collaborated or discussed the assignment.
3. Say approximately how many hours you have spent completing the assignment.
4. Be written in either text format (thus **`README.txt`**) or in Markdown (thus **`README.md`**). No other formats will be accepted.

Submitting Part 3

1. Add me as a collaborator (using my email address **`mchow[AT]cs[DOT]tufts[DOT]edu`**) for your web application in Heroku. You can do this by logging on to Heroku, go to the dashboard for your web application, under "Access", click on the "Add collaborator" button, enter my email address, and click "Save changes".
2. Remember, your map must now work with your server and not the one I provided to you. Change the URL in your map from **`https://jordan-marsh.herokuapp.com/rides`** to **`https://[YOUR_APP_IDENTIFIER_RANDOMLY_GENERATED].herokuapp.com/rides`**, and push your changes to your private GitHub repository (i.e., **`comp120-*****/notuber`**)

Assessment

This lab is worth 10 points, 11.5 maximum.

- (1 point) **`README`**
- (1 point) Updated map that now connects to your server and not my server **`jordan-marsh.herokuapp.com`**
- (6.5 points) Working web application on **`herokuapp.com`** with Cross-Origin Resource Sharing enabled on **`POST /rides`** resource
- One of the following:
    1. (1 point) Accomplish one of the "Going Beyond" items
    2. (2 points) Accomplish two of the "Going Beyond" items
    3. (2 points + 1 bonus point = 3 bonus points) Accomplish three or more of the "Going Beyond" items