Oowlish React/Node.js/JS Challenge

Dear Candidate,

First of all, we are happy you got this far on our hiring process, congratulations! We created this small challenge to give you an opportunity to show your coding skills. We hope you enjoy it as much as we do. You have, roughly, up to one week to send it in.

The Task

We'd like you to create a simple NodeJS REST/GraphQL API which provides data about customers and also a frontend Dashboard that shows information about the customers. You are supposed to follow the constraints below:

1. Use monorepo;
2. Create a model to represent the customer. To use a database or not is your choice but you must use this file as a source of information `customers.json` (File provided with the test);
3. Create two extra virtual fields for latitude and longitude, and fill them up with the customer's address using the Google Maps API;
4. Implement a REST API with three endpoints:
   - Total customers by city:

```
[
    {
        "city": "Warner, NH",
        "customers_total": 20
    },
    ...
]
```

   - List customers by city:

```
[
    {
        "id": 1,
        "first_name": "Laura",
        "last_name": "Richards",
        "email": "lrichards0@reverbnation.com",
        "company": "Meezzy"
    },
    ...
]
```

   - Get a single customer by its `id`;

```
{
    "id": 1,
    "first_name": "Laura",
    "last_name": "Richards",
    "email": "lrichards0@reverbnation.com",
    "gender": "Female",
    "company": "Meezzy",
    "city": "Warner, NH",
    "title": "Biostatistician III",
    "lat": "43.3044615",
    "long": "-71.9650652"
}
```


5. Create a Dashboard using cards to group and **count customers by city**. The cards should be clickable to the next route **show customers of the city**;
6. Create a **paginated** view to list the **customers by city**. The items should be clickable to the next route **show customer detail**;
7. Create a view that shows the **customer detail with a map** showing its location;
8. Implement a simple authentication page using the authentication service **Auth0**, you can follow this documentation. It should protect only the frontend, the API can be public.
9. Write **README.md** instructions on how to get your code up-and-running;
10. Send us your code using a private **Gitlab** (@oowlish-career) or **Github** (@oowlishcareer) repository. Add to the project the respective Oowlish user as a "Maintainer" if applicable.

We kindly ask you not to publish your implementation, to avoid online spoilers.

Feel free to use any libs you deem necessary, but know that we'll also consider your choices around this topic.

Requirements:

- One-command setup:
  - Dockerfile or HSQL (Memory Database) so that the test is totally runnable within itself. No need for external software/configuration (database, auth, etc).
- Unit test.

Non-required features:

- State Management (Redux).

Our review

The focus of the review is on answering some questions such as:

- How clearly can you separate concerns in your code?
- Does the application have good error handling?
- Are SOLID, DRY principles violated?
- How good have you designed your models? Do they make sense?
- Are your docs clear?
- Did you write good/meaningful unit tests?
- Does your solution contain any dead/redundant code?
- Does your application launch?
- Does the application do what it promises?
- Can we find bugs or minor flaws?
- Did you provide a complete product ready to deploy?

The following things most probably will not impress us:

- Overengineering;
- Implementing additional features;
- Typos, grammar mistakes in documentation and code.

Extra points for:

- GraphQL schema/endpoint for data consumption;
- Well written documentation.
- CI/CD;
- Automated Testing (running on CI/CD);

Having that said, good luck and happy coding!