Intro

Welcome to our takehome coding quiz! We're excited to see what you can do. Our aim is for you to have fun while demonstrating your coding skills, interpretation of requirements, and ability to communicate ideas effectively. You should not spend more than 1-2 working sessions completing this. You may use whatever open-source software, libraries and tools you want as long as it's documented with an explanation of why you used it ("because this is what I know" is a totally acceptable answer!). While we prefer React and Node.js, use whatever stack you're comfortable with.

Ensure your solution is well-documented and easily runnable with at most two commands from a Makefile for a local app or a README on how to run a hosted version. You can assume the reviewer has Docker installed or you can share a link to a replit (or similar). This isn't about scalability or DevOps, but about how you approach problem-solving and design. We're here to support you, so if you have questions, reach out. Good luck, and enjoy!

Requirements

Novellia Pets is a new service spun off from Novellia where we gather and display medical records for your furry friend. You will be building the MVP of Novellia Pets which will consist of:

- Creating a new pet (new account)
- Adding a new record to a pet
- A dashboard to see all existing pets
- One additional feature you think would be useful / cool

Every pet has

- A name
- The type of animal
- Name of owner
- Date of Birth

Every pet can have two kinds of medical records

- Vaccines which have the name of the vaccine and a date it was administered
- Allergies which have the name of the allergy, the pet's reactions (e.g. hives, rash) and severity (mild or severe)

User Stories

- As a user, I can add a new pet
- As a user, I can add a new vaccination or allergy record to my pet
- As an admin, I can see all pets added and their records

Please make sure to discuss

- How and why you modeled the data structure(s) the way you did
- How and why you structured your API(s)

- How and why you decided on the page(s) you built
- What improvements you'd make if you want to build this for real

Notes

- You do not need to add login/authentication/sessions.
- You can use any kind of persistence you want (redis/memcached, in-memory, sqlite, SQL database like postgres or MySQL) as long as it persists while the app is running
- When on-site, we will be adding a third type of record (e.g. could be a lab result, vital, visit or something else) so be sure to take that into account when designing and building this.
- You should account for basic user error and handle it gracefully.