Full-Stack Coding Challenge

Introduction

Here at DUST Identity, we enable customers to manage their physical goods via their digital twin, which we call "Things". We enable a secure, trusted, digital connection to the physical twin via our secure **D**iamond **U**nclonable **S**ecurity **T**ags (DUST) .

Logistics

- Take no more than 3 hours to complete what you can. Seriously, stop after 3 hours!
- When done, please provide us with either a link to a private GitHub repository or an archive of your code along with instructions on how to run it.
- There will be a follow-up code review with the team to review your work. This is an opportunity for us to ask questions and understand how you work the problem as well as for you to expound on anything you didn't get to, would have done differently, etc...
- Technology - use whatever languages, libraries, and frameworks you are most comfortable with.
- Questions- If you have any, ask! A good rule of thumb is that if you are blocked for longer than 40 minutes, shoot us a note!

Pre-Configured Environments

If you don't want to set up an environment from scratch, we've provided a few starting points for you. Getting started guides for each can be found in the README.md files within each directory.

API

- Python + Flask (found at `./python-flask-boilerplate`)
- Node + Express (found at `./node-express-boilerplate`)

Front End

- React (via Create React App, found at `./react-app-boilerplate`)

Note: There is no preferential treatment given to using or not using these environments, they are only here for your convenience.

Prompt

As a user, I want to see a list of Things and be able to sort and filter them in order to find the Thing that I am looking for

Requirements

- UI
  - Display a table of Things
  - The table should support pagination
  - Provide at least 1 filtering mechanic
  - Provide at least 1 sorting mechanic
- API
  - Provide an endpoint for returning the list of Things needed to render the table. Use REST best practices
  - The endpoint should read from the provided JSON file of Things

Technical Considerations

Our first customers are ready to use the product immediately, so we don't have a lot of time. To get us up and running quickly, we will forgo a database and rely on JSON file storage.

While you only need to implement the retrieval of the JSON file, it is imperative that your design allows for us to swap out data stores easily. For example, we will want to replace the JSON file storage with a relational DB, and we have already had inquiries from customers about being able to store their data in their own external systems which we will need to retrieve it from.

The JSON file can be found at `./db.json`