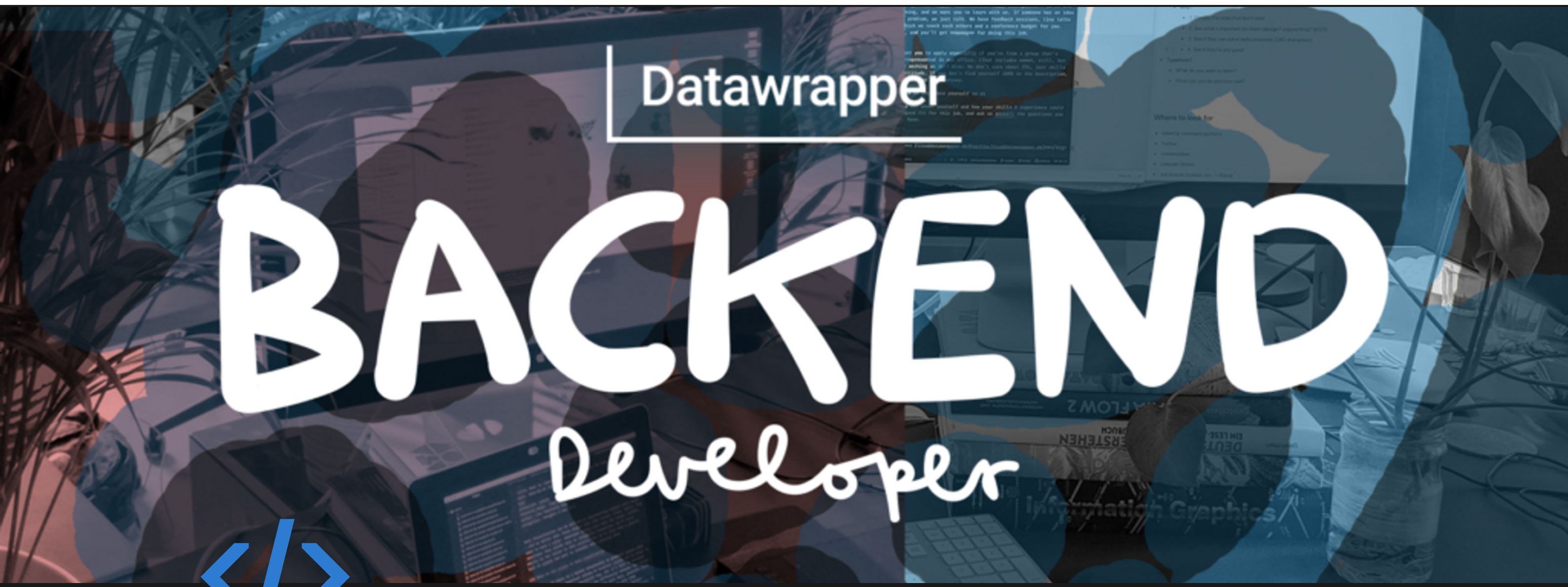

Full Stack Backend Developer

Hello! 👋

We're delighted that you're considering becoming part of the LongShot AI team. We're eager to interact with individuals who are driven, passionate and excited about using their unique skillsets to create extraordinary solutions. If a challenging opportunity that allows you to collaborate with a world-class group of tech enthusiasts appeals to you, we encourage you to undertake this assignment.

Objective

The goal of this assignment is to craft a compact back-end application that serves an API to manage and categorize stock for a small-scale grocery store. This store includes numerous spaces for item storage, with each item having unique properties and storage requirements. Some items also come with an expiration date, necessitating their disposal post-expiration. Create the necessary database structure and API's for the operation as specified below.

Specification

The developed API should possess the capability to carry out the following functionalities and comply with these requirements:

1. Storage Spaces

- A storage space is defined by a unique name and a maximum limit (e.g., 100 items).
- A storage space can be equipped with refrigeration capabilities.
- The API should allow the creation of new storage spaces.
- The functionality to rename existing storage spaces should be available.
- Deletion of a storage space is possible only when it is unoccupied i.e., devoid of any assigned items.
- The total number of items in a storage space should never exceed its maximum limit.
- The API should be able to retrieve a list of all items allocated to a specific storage space.

2. Item Types

- Every item type possesses a distinctive name (like "Ice Cream Sandwich") and an indicator specifying if it requires refrigeration.
- One item type can be associated with multiple items, i.e., there can be many items of type "Ice Cream Sandwich."
- The API should facilitate the creation of new item types.
- The option to rename existing item types should be available.
- Deletion of an existing item type is possible only when there are no items linked to that type.

3. Items

- Each item is associated with an item type and an expiration date.
- An item should be assigned to one specific storage space only.
- Items that require refrigeration can't be stored in a non-refrigerated space.
- The API should facilitate the creation of new items, with the condition that the item's expiry date must be in the future.
- Items can be relocated to a different storage space, given the constraints are satisfied.
- Modifying existing items, i.e., changing the type or expiration date, is not allowed.
- The API should allow the removal of existing items.

- It should provide a list of all items, optionally sorted by expiration date. This list should also support pagination.

Implementation Details

Languages and Frameworks

You can choose between Python and Node.js for this assignment. If Python is your preferred choice, ensure to use a modern, asynchronous Python framework, such as FastAPI . If you choose Node.js, ensure you use a modern Node.js framework that supports TypeScript. Regardless of your choice, always use the latest stable version of your chosen languages and frameworks.

Database

Please utilize a non-relational / NoSql database, like MongoDB, for this assignment. Also, Use an Object-Document Mapping (ODM) framework (e.g. Mongoose / pymongo) to handle the database operations.

External Libraries and Sources

You're free to use any external library to aid your assignment completion, as long as its license is non-restrictive or compatible with the MIT license. Also, you're free to refer to external sources such as StackOverflow, provided you clearly acknowledge and cite any code not written by yourself.

Feel free to make assumptions and but remember to note them down.

Deliverable

Your code's repository and version history should be part of your submission. Your submission should be standalone and executable with a single command, with Docker and docker-compose being the preferred packaging method. Please include a **README** with your submission, detailing the instructions to run your solution and a description of what you have built.

Possible to deploy on any platform and share a Swagger Link or you can create a video of all operations.

Share the private repo with longshot-ai (https://github.com/longshot-ai)

Good Luck 👍 and Feel Free to ask any questions over email.