Backend/Full-Stack Engineering Takehome Test

In this exercise, your goal is to demonstrate to us that you have a solid understanding of python when used as a backend service, that uses a database and exposes a REST interface.

This is a fairly open-ended coding challenge. Use your skills and experience to show us what you're capable of! Be as creative as you wish. Anything above and beyond the minimum requirements will work in your favour.

Expected Time: 2-4 hours

GOOD LUCK!

Scenario

Using a python web framework, you must build a REST web service that exposes CRUD functionality to a database that stores carbon usage data for customers.

Database

The database should have at least this structure. Feel free to make your data structure more complex, but be prepared to explain it if it is not obvious.

user	
id	long
name	string

usage	
id	long
user_id	id
usage_type_id	long
usage_at	datetime
amount	float

usage_types	
id	long
name	string

unit	string
factor	float

Data
usage_types

id	name	unit	factor
100	electricity	kwh	1.5
101	water	kg	26.93
102	heating	kwh	3.892
103	heating	l	8.57
104	heating	m3	19.456

Requirements

- The backend must be made in a python web framework
- You must use an ORM such as SQLAlchemy or Django
- Your backend supports authentication (It's built in to Django)
- The data in the usage_types table above is pre-loaded
- Your API is secured with token-based authentication (such as JWT)
- Your API supports pagination, sorting and a filter by time range when retrieving
- Your API should support all CRUD actions. Try to think about how a frontend might need to use this to allow users to submit, edit or delete their usage and retrieve it.
- You may use an in-memory database like SQLite
- Your files should be in a Github repository we can access.

- The README in the root directory of your repository should provide instructions so that we can run and test it locally. We *highly* recommend you set it up with Docker to make both our lives easier.
- Your API is documented somewhere and mentioned in your README.
- Please also include in your README how long it took you to complete this task, and if you had any challenges that you had to overcome.

Full Stack Challenge (optional)

If you want to really impress us with your skills, you'll also be able to make a simple UI on top of the backend you just made! Your UI should be able to do the following:

- Written as an SPA in React / TypeScript
- Authenticate against the backend
- Form to create a new "usage". The user should be able to enter when the activity occurred and select an appropriate usage type, amount and unit.
- View a paginated list of all usages
    - All previously created usages should also include the calculated emissions
    - Emissions calculated should not change if usage_types.factor changes
    - Button to delete a usage, with a confirmation modal
- A simple chart (you may use a 3rd-party library) showing emissions over time
- README explaining how to build and run this environment locally and in a container against the backend you made in the first part of the exercise (You may combine both parts into one README explaining how to get the whole project to run)

GOOD LUCK!