Test Engineer Challenge: Bagels API

What's Required

Your objective is to write a suite of automated tests to ensure that our "Bagels" API is valid to the listed specifications below. Your test suite is expected to have at least several failed tests as the current version of the API doesn't conform perfectly to the requirements (on purpose of course!). Please ensure that your test suite has good coverage of type validation, authentication, pagination, and filtering criteria.

You may write these tests using any language of your choice (Python would be a plus but definitely not required). If you do not have experience working with a test framework, you may choose to use Postman which provides a simple and GUI way to write HTTP methods and then wrap them in a test case using Javascript (It has many examples to get you started).

Please timebox yourself to 2-3 hours of testing.

When you're finished, please email us your test code and output. You may also include any additional information you need to share about your tests or make note of additional test cases you would have written if you had more time.

Thank you in advance for taking the time to complete this assignment. We will review your results and get back to you as soon as we can!

API Specification

API URL

https://cmb-bagels-api.herokuapp.com/bagels/

Basic Authentication

Username: cmb
Password: bagel

GET

Returns a list of bagel objects according to the filtering criteria passed in via query parameters.

Sample query:
https://cmb-bagels-api.herokuapp.com/bagels/?dist=100&origin=37.774929,-122.419416&min_age=21&max_age=29&limit=2&offset=0

Available Parameters

Parameter	Description
name	A full or partial and case-insensitive name of a Bagel
gender	Gender preference: m (male) or f (female)
dist	Maximum match distance in miles

origin	lat/long string of some location
min_age	Minimum age preference
max_age	Maximum age preference
limit	Maximum number of items to return
offset	The starting position of the items to return

Sample GET response from */bagels*


```
[
  {
    "id": 1,
    "name": "Taylor Swift",
    "age": 27,
    "gender": "female",
    "locations": [
      {
        "name": "San Francisco",
        "coordinates": [
          37.774929,
          -122.419416
        ]
      },
      {
        "name": "Oakland",
        "coordinates": [
          36.244273,
          -120.49804
        ]
      }
    ]
  },
  {
    "id": 2,
    "name": "Kevin Spacey",
    "age": 58,
```

```
    "gender": "male",
    "locations": [
      {
        "name": "Washington-DC",
        "coordinates": [
          38.89565,
          -76.943174
        ]
      }
    ]
  },
  {
    "id": 3,
    "name": "Emma Watson",
    "age": 28,
    "gender": "female",
    "locations": [
      {
        "name": "Los Angeles",
        "coordinates": [
          34.062264,
          -118.340361
        ]
      },
      {
        "name": "Daly City",
        "coordinates": [
          37.68941,
          -122.462532
        ]
      }
    ]
  },
  {
    "id": 4,
    "name": "Emilia Clarke",
    "age": 30,
    "gender": "female",
    "locations": [
      {
        "name": "Los Angeles",
        "coordinates": [
```

```
          34.043566,
          -118.391092
        ]
      }
    ]
  },
  {
    "id": 5,
    "name": "Chris Martin",
    "age": 40,
    "gender": "male",
    "locations": [
      {
        "name": "Whitestone, UK",
        "coordinates": [
          52.504531,
          -1.433243
        ]
      }
    ]
  }
]
```

GET query requirements

- All query parameters are optional. If a parameter is missing, no filtering should be performed for that specific field.
- All range queries should be inclusive (e.g. min_age=21&max_age=23 should return bagels with an age of 21, 22, or 23).
- Only locations that fall within the geographic distance from your origin should be included in the result.
- Paginating results should work as intended using the limit and offset parameters.

POST

Adds a new bagel object and its locations, returns newly created bagel object with "ID" or a validation error.

Sample POST data:

```
{
      "name":"Chris Hemsworth",
      "age":34,
      "gender":"m",
      "locations":[{"name":"Melbourne, AU","coordinates":
[-37.9252124,144.8277777]}]
}
```

Required Parameters

Parameter	Description
name	Full name of your Bagel
gender	The gender of your bagel: m or f
age	Age of your bagel (18-99)
locations	List of objects where each object has a **name** for its location and **coordinates** which is a tuple containing lat/long

Sample valid POST response to */bagels*


```
{
    "id": 2000057,
    "name": "Test User",
    "age": 27,
    "gender": "f",
    "activity": [
        {
            "name": "San Francisco",
            "coordinates": [
                37.774929,
                -12.419416
            ]
        }
    ]
}
```


POST requirements

- Newly created bagels can be found during a GET query
- Invalid POST data returns a descriptive error reason
- At least one location is required for new bagel creation