Flask Cupcakes

Springboard

Download our code

In this exercise, you will build a JSON API, test it using Insomnia, write integration tests, and build a HTML/JS frontend.

Part Zero: Set Up

Make a virtual environment and install the dependencies.

Make your project a Git repo.

Part One: Cupcake Model

Create ***Cupcake*** model in ***models.py***.

It should have the following columns:

- ***id***: a unique primary key that is an auto-incrementing integer
- ***flavor***: a not-nullable text column
- ***size***: a not-nullable text column
- ***rating***: a not-nullable column that is a float
- ***image***: a non-nullable text column. If an image is not given, default to https://tinyurl.com/demo-cupcake

Make a database called ***cupcakes***.

Once you've made this, you can run our ***seed.py*** file to add a few sample cupcakes to your database.

Part Two: Listing, Getting & Creating Cupcakes

Make routes for the following:

GET /api/cupcakes

Get data about all cupcakes.

Respond with JSON like: `{cupcakes: [{id, flavor, size, rating, image}, ...]}` .

The values should come from each cupcake instance.

GET /api/cupcakes/*[cupcake-id]*

Get data about a single cupcake.

Respond with JSON like: `{cupcake: {id, flavor, size, rating, image}}` .

This should raise a 404 if the cupcake cannot be found.

POST /api/cupcakes

Create a cupcake with flavor, size, rating and image data from the body of the request.

Respond with JSON like: `{cupcake: {id, flavor, size, rating, image}}` .

Test that these routes work in Insomnia.

We've provided tests for these three routes; these test should pass if the routes work properly.

You can run our tests like:

```
(venv) $ python -m unittest -v tests
```

Part Three: Update & Delete Cupcakes

Make routes for the following:

PATCH /api/cupcakes/*[cupcake-id]*

Update a cupcake with the id passed in the URL and flavor, size, rating and image data from the body of the request. You can always assume that the entire cupcake object will be passed to the backend.

This should raise a 404 if the cupcake cannot be found.

Respond with JSON of the newly-updated cupcake, like this: `{cupcake: {id, flavor, size, rating, image}}` .

DELETE /api/cupcakes/*[cupcake-id]*

This should raise a 404 if the cupcake cannot be found.

Delete cupcake with the id passed in the URL. Respond with JSON like `{message: "Deleted"}` .

Test these routes in Insomnia.

Part Four: Write More Tests

Add tests for the PATCH and DELETE routes.

Part Five: Start on the frontend

Make this route:

GET /

This should return an HTML page (via ***render_template***). This page should be entirely static (the route should just render the template, without providing any information on cupcakes in the database). It should show simply have an empty list where cupcakes should appear and a form where new cupcakes can be added.

Write Javascript (using axios and jQuery) that:

- queries the API to get the cupcakes and adds to the page
- handles form submission to both let the API know about the new cupcake and updates the list on the page to show it

(You do not need to use WTForms to make this form; this is a possibility in the further study.)

Further Study

- Add tests to make sure that the GET/PATCH/DELETE routes return a 404 when the cupcake cannot be found.
- Add functionality for searching for cupcakes where you can type in a search term, submit to the backend and see a newly filtered list of cupcakes.

  HINT: Make sure a search term is passed to the backend and that you are using a ***LIKE*** or ***ILIKE*** SQL query to search.
- Refactor your front-end code to be object-oriented using class methods to ***fetchAllCupcakes*** and ***createCupcakes*** and instance methods for updating and deleting cupcakes as well as searching for cupcakes.
- Refactor your HTML page to render a form created by WTForms.
- Enhance your search functionality so that you do not need to wait to submit to filter by flavors.
- Add functionality on the front-end to update a cupcake.
- **Are you still here??** Then add another table for ingredients. When you add or edit a cupcake, you can identify what ingredients you need for that cupcake. You should also have a page where you can add or edit ingredients.

Solution

View our solution