

Back-End Practice Assessment

The goal of this practice assessment is to test your backend development skills. The objective is to write a simple JSON API.

This practice assessment will **not** be graded, however the official assessment will be graded based on the following criteria (so it is good practice to keep these categories in mind while completing this practice assessment):

- **Correctness:** Is your solution complete and does it pass different test cases?
- **Code Organization, Readability, & Maintainability:** Is your code easy to read and well organized?
- **Code Performance:** Is your code efficient? Did you use appropriate data structures?
- **Best Practices:** Did you utilize good programming practices (write unit tests, avoid anti-patterns)? Did you show a good grasp of your language/framework of choice?

You can use one of the following programming languages to complete the assessment: **Javascript (NodeJS), Python, Ruby, Java, Go, or Rust**. You may use any framework for your language of choice.

Before you tackle the assessment, it may be helpful to review in your language of choice:

1. Creating a simple JSON API

If you have never written a JSON API before, here are a few resources that can help you for different languages:

- Python - [Flask \(Flask JSON API\)](#)
- Javascript - [Node + Express](#)
- Java - [Spring Boot \(JSON API\)](#)
- Ruby - [Rails JSON API](#)

JSON Recipe File

You can paste the following JSON into a file called `data.json`, and use it as your data source

```
{ "recipes": [ { "name": "scrambledEggs", "ingredients": [ "1 tsp oil", "2 eggs", "salt" ], "instructions": [ "Beat eggs with salt", "Heat oil in pan", "Add eggs to pan when hot", "Gather eggs into curds, remove when cooked", "Salt to taste and enjoy" ] }, { "name": "garlicPasta", "ingredients": [ "500mL water", "100g spaghetti", "25mL olive oil", "4 cloves garlic", "Salt" ], "instructions": [ "Heat garlic in olive oil", "Boil water in pot", "Add pasta to boiling water", "Remove pasta from water and mix with garlic olive oil", "Salt to taste and enjoy" ] }, { "name": "chai", "ingredients": [ "400mL water", "100mL milk", "5g chai masala", "2 tea bags or 20 g loose tea leaves" ], "instructions": [ "Heat water until 80 C", "Add milk, heat until 80 C", "Add tea leaves/tea bags, chai masala; mix and steep for 3-4 minutes", "Remove mixture from heat; strain and enjoy" ] } ] }
```

Part 1

Build a GET route that returns all recipe names.

A GET request to `http://localhost:3000/recipes` returns: **Response body (JSON):** `{ "recipeNames": ["scrambledEggs", "garlicPasta", "chai"] }` **Status: 200**

Part 2

Build a GET route that takes a recipe name as a **string** param. Return the ingredients and the number of steps in the recipe as JSON

A GET request to `http://localhost:3000/recipes/details/garlicPasta` returns:
If recipe exists: Response body (JSON): `{ "details": { "ingredients": ["500mL water", "100g spaghetti", "25mL olive oil", "4 cloves garlic", "Salt"], "numSteps": 5 } }` **Status: 200** --- **If recipe does NOT exist: Response body (JSON):** `{}` **Status: 200**

Part 3

Add a POST route that can add additional recipes in the existing format to the backend with support for the above routes.

A POST request to **http://localhost:3000/recipes** with body { "name": "battered Bagel", "ingredients": ["1 bagel", "butter"], "instructions": ["cut the bagel", "spread butter on bagel"] } returns: **Response body: None Status: 201**

Error Response:

If the recipe already exists:

Response body (JSON): { "error": "Recipe already exists" } Status: 400

Part 4

Add a PUT route that can update existing recipes.

A PUT request to **http://localhost:3000/recipes** with body { "name": "batteredB agel", "ingredients": ["1 bagel", "2 tbsp butter"], "instructions": ["cut the bagel", "spread butter on bagel"] } returns: **Response body: None Status: 204**

Error Response:

If the recipe doesn't exist:

Response body (JSON): { "error": "Recipe does not exist" } Status: 404

