CS-546 Lab 7

A Recipe API

For this lab, you will create a simple server that provides an API for someone to Create, Read, Update, and Delete recipes. These recipes will be stored in a database named **lab7-recipes**.

This recipe database will also provide support for creating, reading, updating, and deleting comments for a recipe.

The recipe object

```
{
  _id: "A uuid",
  title: "Recipe title",
  ingredients: [
    {
      name: "Ingredient name",
      amount: "portion amount"
    }
  ],
  steps: [
    "First step",
    "Second step",
    "Third step"
  ],
  comments: []
}
```

For example, a fried egg recipe:

```
{
  _id: "bd8fa389-3a7a-4478-8845-e36a02de1b7b",
  title: "Fried Eggs",
  ingredients: [
    {
      name: "Egg",
      amount: "2 eggs"
    },
    {
      name: "Olive Oil",
      amount: "2 tbsp"
    },
  ],
  steps: [
```

```
    "First, heat a non-stick pan on medium-high until hot",
    "Add the oil to the pan and allow oil to warm; it is ready the oil immediately sizzles upon contact with a drop of water.",
    "Crack the egg and place the egg and yolk in a small prep bowl; do not crack the yolk!",
    "Gently pour the egg from the bowl onto the oil",
    "Wait for egg white to turn bubbly and completely opaque (approx 2 min)",
    "Using a spatula, flip the egg onto its uncooked side until it is completely cooked (approx 2 min)",
    "Remove from oil and plate",
    "Repeat for second egg"
  ],
  comments: []
}
```

Comments

Your comment will be stored on the recipe document.

```
{
  _id: "A uuid",
  poster: "poster name",
  comment: "the comment"
}
```

For example:

```
{
  _id: "9b527da1-67c0-4c13-ae99-3c1288ff2975",
  poster: "Gordan Ramsay",
  comment: "These eggs are delicious!"
}
```

Packages you will use:

You will use the **express** package as your server.

You can read up on express (Links to an external site.)Links to an external site. on its home page. Specifically, you may find the API Guide section on requests (Links to an external site.)Links to an external site. useful.

You will use the **node-uuid** package in order to generate unique id's to use as your identifiers.

You can read up on node-uuid (Links to an external site.)Links to an external site. on the Github project page.
You will also use the mongodb (Links to an external site.)Links to an external site. package.
You may use the lecture 4 code (Links to an external site.)Links to an external site. as a guide.

You may use the lecture 5 code (Links to an external site.)Links to an external site. as a guide.
You may use the lecture 6 code (Links to an external site.)Links to an external site. as a guide.

You must save all dependencies to your package.json file

Your Routes

verb	path	description
GET	`/recipes`	Responds with a list of all recipes in the format of `{_id: RECIPE_ID, title: RECIPE_TITLE}`
GET	`/recipes/:id`	Responds with the full content of the specified recipe
POST	`/recipes`	Creates a recipe with the supplied data in the request body, and returns the new recipe
PUT	`/recipes/:id`	Updates the specified recipe with **only** the supplied changes, and returns the updated recipe
DELETE	`/recipes/:id`	Deletes the recipe
GET	`/comments/recipe/:recipeId`	Returns a list of all comments in the specified recipe, in the format of: `{_id: COMMENT_ID, recipeId: RECIPE_ID, recipeTitle: RECIPE_TITLE, poster: COMMENT_NAME, comment: COMMENT}`
GET	`/comments/:commentId`	Returns the comment specified by that commentId in the format of `{_id: COMMENT_ID, recipeId: RECIPE_ID, reciipeTitle: RECIPE_TITLE, poster: COMMENT_NAME, comment: COMMENT}`
POST	`/comments/:recipeId/`	Creates a new comment with the supplied data in the request body for the stated recipe, and returns the new comment
PUT	`/comments/:recipeId/:commentId`	Updates the specified comment for the stated recipe with **only** the supplied changes, and returns the updated comment
DELETE	`/comments/:id`	Deletes the comment specified

Any issues should result in a properly failed status code and a description of the error in JSON.

Requirements

1. You **must not submit** your node_modules folder
2. You **must remember** to save your dependencies to your package.json folder
3. You must do basic error checking in each function

1. Check for arguments existing and of proper type.
2. Throw if anything is out of bounds (ie, trying to perform an incalculable math operation or accessing data that does not exist)
3. If a function should return a promise, you should mark the method as an `async` function and return the value. Any promises you use inside of that, you should *await* to get their result values. If the promise should reject, ten you should throw inside of that promise in order to return a rejected promise automatically. Thrown exceptions will bubble up from any awaited call that throws as well, unless they are caught in the async method.
4. You **must remember** to update your package.json file to set `app.js` as your starting script!
5. You **must** submit a zip, rar, tar.gz, or .7z archive or you will lose points, named in the followign format: `LastName_FirstName_CS546_SECTION.zip` (or, whatever the file extension may be). You will lose points for not submitting an archive.