Recipes API Documentation

This API provides endpoints to manage recipes, including creating, reading, updating, and deleting recipe data. The API is built using Express.js and uses MongoDB for data storage. The endpoints allow for retrieving all recipes, searching recipes by various criteria, adding new recipes, and more.

Base URL

`/recipes-services/recipes`

Endpoints:

1. Get All Recipes

Retrieve a list of all recipes stored in the database.

- **URL:** `/getAllRecipes`
- **Method:** `GET`

Response:

- **200 OK: Returns an array of all recipes.**
- **500 Internal Server Error: If an error occurs while fetching recipes.**

Example Response:

```
[
  {
    "_id": "64b7a1e9f1ad3f18a94c15d1",
    "name": "Spaghetti Bolognese",
    "ingredients": ["spaghetti", "minced meat", "tomato sauce"],
    "instructions": "Cook spaghetti and mix with sauce.",
    "category": "Italian",
    "image": "spaghetti.jpg"
  }
]
```

2. Get Recipe By ID

Retrieve details of a specific recipe by its unique ID.

- **URL**: `/getRecipeById`
- **Method**: `GET`
- **Query Parameter**:
  - `id` (required): The unique identifier of the recipe.
- **Response**:
  - **200 OK**: Returns the recipe object corresponding to the provided ID.
  - **404 Not Found**: If the recipe with the given ID does not exist.
  - **500 Internal Server Error**: If an error occurs during the request.

Example Request:

```
GET /getRecipeById?id=64b7a1e9f1ad3f18a94c15d1
```

Example Response:

```
{
  "_id": "64b7a1e9f1ad3f18a94c15d1",
  "name": "Spaghetti Bolognese",
  "ingredients": ["spaghetti", "minced meat", "tomato sauce"],
  "instructions": "Cook spaghetti and mix with sauce.",
  "category": "Italian",
  "image": "spaghetti.jpg"
}
```

3. Search Recipe

Search for recipes based on name, ingredients, or category.

- **URL**: `/searchRecipe`
- **Method**: `GET`
- **Query Parameters**:
  - `name` (optional): Search recipes by name.
  - `ingredient` (optional): Search recipes containing a specific ingredient.
  - `category` (optional): Search recipes by category.
- **Response**:
  - **200 OK**: Returns an array of recipes that match the search criteria.
  - **404 Not Found**: If no matching recipes are found.
  - **500 Internal Server Error**: If an error occurs during the request.

Example Request:

```
GET /searchRecipe?name=Spaghetti&category=Italian
```

Example Response:

```
[
  {
    "_id": "64b7a1e9f1ad3f18a94c15d1",
    "name": "Spaghetti Bolognese",
    "ingredients": ["spaghetti", "minced meat", "tomato sauce"],
    "instructions": "Cook spaghetti and mix with sauce.",
    "category": "Italian",
    "image": "spaghetti.jpg"
  }
]
```

4. Add One Recipe

Add a new recipe to the database.

- **URL**: `/addOneRecipe`
- **Method**: `POST`
- **Request Body**:
  - `name` (required): The name of the recipe.
  - `ingredients` (required): A list of ingredients used in the recipe.
  - `instructions` (required): The preparation instructions.
  - `category` (required): The category (e.g., Italian, Mexican).
  - `image` (required): A URL or filename of the recipe image.
- **Response**:
  - **201 Created**: Recipe created successfully.
  - **400 Bad Request**: If required fields are missing or invalid.
  - **500 Internal Server Error**: If an error occurs during the request.

Example Request:

```
{
  "name": "Grilled Cheese Sandwich",
  "ingredients": ["bread", "cheese", "butter"],
  "instructions": "Grill the sandwich with butter until golden brown.",
  "category": "American",
  "image": "grilled-cheese.jpg"
}
```

5. Add Many Recipes

Add multiple recipes to the database at once.

- **URL**: `/addManyRecipes`
- **Method**: `POST`
- **Request Body**: An array of recipe objects, each containing:
  - `name` (required): The name of the recipe.
  - `ingredients` (required): A list of ingredients used in the recipe.
  - `instructions` (required): The preparation instructions.
  - `category` (required): The category (e.g., Italian, Mexican).
  - `image` (required): A URL or filename of the recipe image.
- **Response**:
  - **201 Created**: Recipes added successfully.
  - **400 Bad Request**: If required fields are missing or invalid.
  - **500 Internal Server Error**: If an error occurs during the request.

Example Request:

```
[
  {
    "name": "Tacos",
    "ingredients": ["taco shells", "beef", "cheese", "lettuce"],
    "instructions": "Fill taco shells with beef and toppings.",
    "category": "Mexican",
    "image": "tacos.jpg"
  },
  {
    "name": "Caesar Salad",
```

```
    "ingredients": ["lettuce", "croutons", "parmesan", "Caesar
dressing"],

    "instructions": "Toss lettuce with dressing and croutons.",

    "category": "Salad",

    "image": "caesar-salad.jpg"

  }

]
```

6. Update Recipe

Update the details of an existing recipe in the database.

- **URL**: `/updateRecipe`
- **Method**: `PUT`
- **Request Body**:
  - `id` (required): The unique identifier of the recipe to update.
  - Other fields that can be updated include:
    - `name`: The name of the recipe.
    - `ingredients`: A list of ingredients used in the recipe.
    - `instructions`: The preparation instructions.
    - `category`: The category (e.g., Italian, Mexican).
    - `image`: A URL or filename of the recipe image.
- **Response**:
  - **200 OK**: Recipe updated successfully.
  - **400 Bad Request**: If required fields are missing or invalid.
  - **404 Not Found**: If the recipe with the given ID does not exist.
  - **500 Internal Server Error**: If an error occurs during the update.

Example Request:

```
{

  "id": "64b7a1e9f1ad3f18a94c15d1",

  "name": "Updated Spaghetti Bolognese",

  "ingredients": ["spaghetti", "minced meat", "tomato sauce",
"garlic"],

  "instructions": "Cook spaghetti and mix with sauce and garlic.",

  "category": "Italian",

  "image": "updated-spaghetti.jpg"

}
```

Example Response:

```
{

  "message": "Recipe updated successfully."

}
```

7. Delete Recipe

Delete a recipe from the database by its unique ID.

- **URL**: `/deleteRecipe`
- **Method**: `DELETE`
- **Query Parameter**:
  - `id` (required): The unique identifier of the recipe to delete.
- **Response**:
  - **200 OK**: Recipe deleted successfully.
  - **404 Not Found**: If the recipe with the given ID does not exist.
  - **500 Internal Server Error**: If an error occurs during the deletion.

Example Request:

```
DELETE /deleteRecipe?id=64b7a1e9f1ad3f18a94c15d1
```

Example Response:

```
{
  "message": "Recipe deleted successfully."
}
```

Recipe Schema

The `recipeSchema` defines the structure of a recipe document in the database:

```
const recipeSchema = new mongoose.Schema({
  name: {
    type: String,
    required: true,
  },
  ingredients: {
    type: [String],
    required: true,
  },
  instructions: {
    type: String,
    required: true,
  },
  category: {
    type: String,
    required: true,
  },
```

```
  image: {
    type: String,
    required: true,
  },
});
```

Fields:

- **name** (String): The name of the recipe (e.g., "Spaghetti Bolognese").
- **ingredients** (Array of Strings): A list of ingredients used in the recipe (e.g., ["spaghetti", "tomato sauce"]).
- **instructions** (String): The steps to prepare the recipe.
- **category** (String): The category of the recipe (e.g., "Italian", "Mexican").
- **image** (String): A URL or filename for an image of the dish.

Responses

Each response follows the standard format:

- **Success**: Returns the corresponding data with status codes `200`, `201`, etc.
- **Error**: Returns status codes like `400`, `404`, or `500` with appropriate error messages.