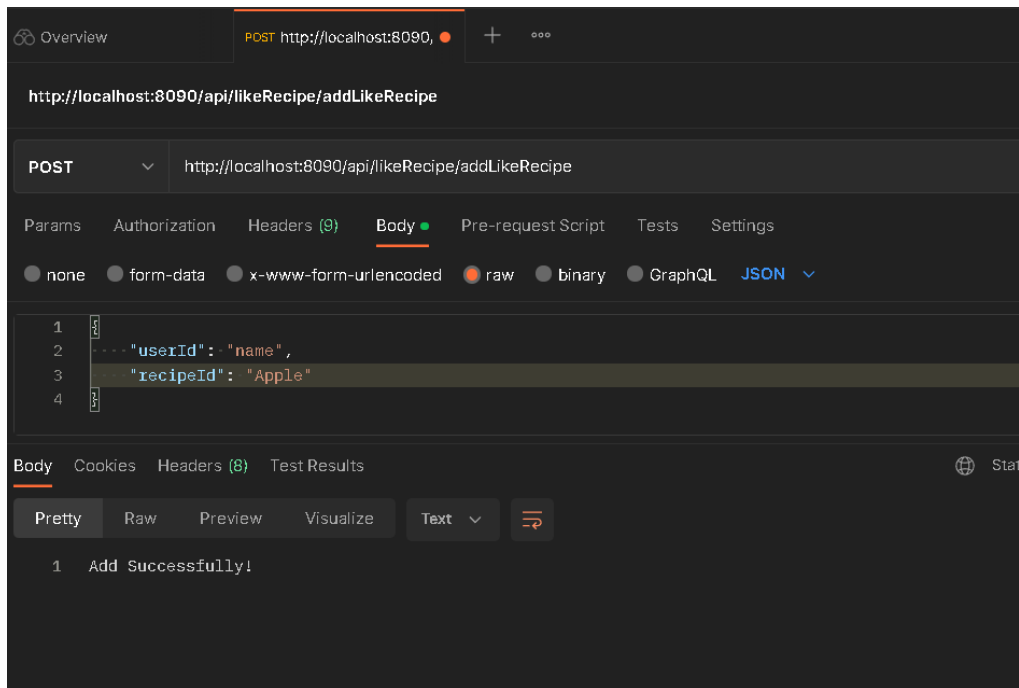


API type	link	body	description
Get	http://localhost:8090/api/likeRecipe/findAll	None	get all likes in the database
Post	http://localhost:8090/api/likeRecipe/addLikeRecipe	{ "userId": "name", "recipeId": "Apple" }	Add a new record for user to like the recipe
Post	http://localhost:8090/api/likeRecipe/findByUserId	{ "userId": "name" }	Find recipes that the user liked
Post	http://localhost:8090/api/likeRecipe/delete	{ "userId": "name", "recipeId": "Apple" }	Delete the record for user to cancel the like for the recipe
Post	http://localhost:8090/api/fitnessGoal/addFitnessGoal	{ "userId": "SampleUserId1", "goal": "goal" }	Add Fitness Goal
Post	http://localhost:8090/api/fitnessGoal/findByUserId	{ "userId": "SampleUserId2" }	find fitness goal by user id
Post	http://localhost:8090/api/fitnessGoal/delete	{ "userId": "SampleUserId2" }	delete fitness goal by user id
Get	http://localhost:8090/api/fitnessGoal/findAll	None	get all fitness goals in database

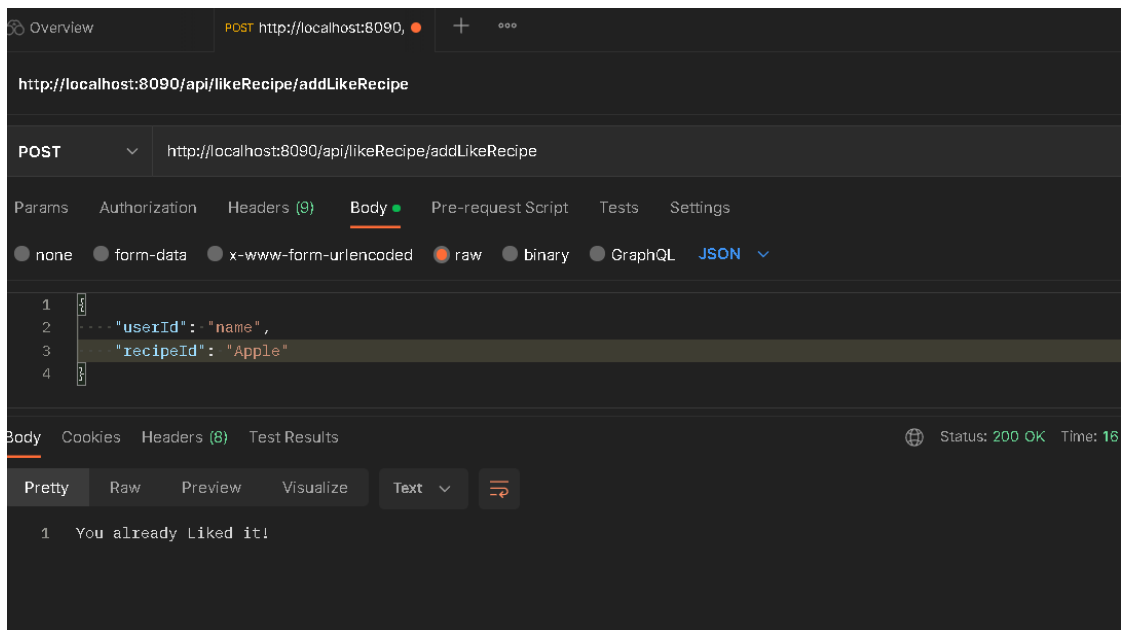
Postman Api Tests

1. AddLikeRecipe

Not existed



Already existed



2. findAll

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8090/api/likeRecipe/findAll`
- Method:** `GET`
- Body:** This request does not have a body.
- Response Body:** The response is displayed in JSON format, showing two objects:

```
1  [
2    {
3      "id": 1,
4      "userId": "name",
5      "recipeId": "Lemon"
6    },
7    {
8      "id": 2,
9      "userId": "name",
10     "recipeId": "Apple"
11   }
12 ]
```

3. findByUserId

The screenshot displays a REST client interface with the following details:

- URL:** `http://localhost:8090/api/likeRecipe/findByUserId`
- Method:** `POST`
- Body Type:** `JSON` (selected from a dropdown menu that also includes `none`, `form-data`, `x-www-form-urlencoded`, `raw`, `binary`, and `GraphQL`).
- Request Body (JSON):**

```
1 {  
2   "userId": "name"  
3 }
```
- Response Body (JSON, Pretty view):**

```
1 [  
2   {  
3     "id": 1,  
4     "userId": "name",  
5     "recipeId": "Lemon"  
6   },  
7   {  
8     "id": 2,  
9     "userId": "name",  
10    "recipeId": "Apple"  
11  }  
12 ]
```

4. delete existed

The screenshot shows a REST client interface with the URL `http://localhost:8090/api/likeRecipe/delete`. The method is **POST**. The **Body** tab is selected, showing a JSON payload:

```
{  "userId": "name",  "recipeId": "Apple"}
```

. The **Test Results** tab is also visible, showing a single line of text: `Delete Successfully!`. The status bar at the bottom right indicates a status of 200.

Not existed

The screenshot shows the same REST client interface with the URL `http://localhost:8090/api/likeRecipe/delete` and the same JSON payload:

```
{  "userId": "name",  "recipeId": "Apple"}
```

. The **Test Results** tab shows a single line of text: `There is no such recipe`. The status bar at the bottom right indicates a status of 200.

FitnessGoal add

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8090/api/fitnessGoal/addFitnessGoal`
- Method:** `POST`
- Body:** A JSON object: `{ "userId": "SampleUserId2", "goal": "goal" }`
- Status:** `200 OK`, Time: `4.17 s`, Size: `269 B`
- Response:** `1 Add Successfully`

find by user id

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8090/api/fitnessGoal/findByUserId`
- Method:** `POST`
- Body:** A JSON object: `{ "userId": "SampleUserId2" }`
- Status:** `200 OK`, Time: `676 ms`, Size: `300 B`
- Response:** A JSON object: `{ "id": 3, "userId": "SampleUserId2", "goal": "goal" }`

delete

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8090/api/fitnessGoal/delete`
- Method:** `POST`
- Body:** `"userId": "SampleUserId2"`
- Status:** 200 OK
- Time:** 2.25 s
- Size:** 272 B
- Response:** `1 Delete Successfully`

findAll

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8090/api/likeRecipe/findAll`
- Method:** `GET`
- Query Params:** A table with columns KEY, VALUE, and DESCRIPTION.
- Status:** 200 OK
- Time:** 2.38 s
- Size:** 302 B
- Response:**

```
1 {
2   "id": 2,
3   "userId": "SampleUserId1",
4   "goal": "goal"
5 }
6
7
```