

## **RECIPE REST API USER GUIDE**

Recipe Rest API was created by using spring boot rest API frameworks.

This Rest API enables performing recipe operations like create, update, delete and list.

If you want to see these operations on swagger UI, you can visit the below swagger UI link.

swagger rest uri :

<http://localhost:8080/swagger-ui/>

If you entered once, you would probably see a web page like shown below.

The screenshot shows the Swagger UI interface for a REST API. At the top, the URL is displayed as `localhost:8080/swagger-ui/#/recipe-controller`. The interface is organized into sections for different controllers:

- recipe-controller** (Recipe Controller): This section contains seven API endpoints:
  - GET /api/v1/recipe/{recipeId}**: Description: performs to get one recipe
  - GET /api/v1/recipe/allrecipes**: Description: getAllRecipes() performs to get all recipes
  - POST /api/v1/recipe/create**: Description: createRecipe()> performs to create a recipe
  - DELETE /api/v1/recipe/delete/{recipeId}**: Description: deleteRecipeByld()> performs to delete a recipe
  - GET /api/v1/recipe/ingredients/{recipeId}**: Description: getRecipeIngredients()> performs to get all recipe ingredients
  - GET /api/v1/recipe/instructions/{recipeId}**: Description: getRecipeInstructions()> performs to get all recipe instructions
  - PUT /api/v1/recipe/update/{recipeId}**: Description: updateRecipe()> performs to update a recipe
- user-controller** (User Controller): This section contains three API endpoints:
  - POST /recipe/v1/users/authenticate**: Description: authenticate()> performs to log in recipe system
  - GET /recipe/v1/users/default**: Description: default request()> performs to default request in recipe system
  - POST /recipe/v1/users/signup**: Description: signUp()> performs to sign up in recipe system
- Models**: This section is currently empty.

This web page is the API documentation for the rest controller.

We can start to express the recipe rest operations.

## Recipe Controller :

Parent endpoint for this controller :

<http://localhost:8080/api/v1/recipe>

All API endpoints are secure. That's why you have to get a token. You can get the token by calling the User Controller endpoint.

We can use the login endpoint to get a bearer token.

This example was showed like below by using postman.

login post: <http://localhost:8080/login>

The screenshot shows a Postman interface with the following details:

- Request Method:** POST
- URL:** http://localhost:8080/login
- Body (JSON):**

```
1 {  
2   "username": "demo",  
3   "password": "12345"  
4 }  
5
```
- Headers (11):**
  - authorization → Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJkZW1vliwiZXhwIjoxNjQxNDI4MzIxLCJyb2xlcyI6W3siYXV0aG9yaXR5IjoiQURNSU4ifV19.qiIEheuCG4jGTWFbtVpkv3ZIWtU63bD66x4HPzD\_k7-i1iCaW2ACw\_6Yf6wNxoUgaVbSDYi5EHVWP1teiap6uA
  - cache-control → no-cache, no-store, max-age=0, must-revalidate
  - connection → keep-alive
  - content-length → 0
  - date → Wed, 05 Jan 2022 19:18:41 GMT
  - expires → 0
  - keep-alive → timeout=60
  - pragma → no-cache
  - x-content-type-options → nosniff
  - x-frame-options → DENY
  - x-xss-protection → 1; mode=block
- Status:** 200 OK
- Time:** 257 ms

if we entered an incorrect username and password, the system would show us a 403 forbidden error like below.

The screenshot shows a POST request to `http://localhost:8080/login`. The request body is a JSON object with fields `username` and `password`. The response status is **403 Forbidden**.

```
1 {  
2   "username": "demo",  
3   "password": "123456"  
4 }  
5
```

Status: **403 Forbidden** Time: **440 ms**

Now we can use this token on every request by adding it request HTTP header in the Authorization section.

Firstly, we should add a bearer token to every request before we call the rest endpoints.

We are able to add like below postman request.

The screenshot shows the Postman application interface. At the top, there is a banner message: "You must use v7.0 or higher to access your workspaces and collections. See what's new". Below the banner, the URL bar shows "http://localhost:8080/api/v1/recipe/4". The "Headers" tab is selected, displaying two headers: "Authorization" (with value "Bearer eyJhbGciOiJIUzIwMjQ4...") and "Content-Type". The "Body" tab is also visible. At the bottom right, the status is shown as "Status: 403 Forbidden Time: 347 ms".

## ENDPOINTS:

### 1. /api/v1/recipe/{recipId}

Future: performs to get a recipe that given recipe id.

Endpoint : <http://localhost:8080/api/v1/recipe/{recipId}>

Type : GET

Parameter :

```
{
  name : recipId
  type : integer
}
```

Example get request: <http://localhost:8080/api/v1/recipe/4>

If you fetch this request, the result will be shown below.

## Results recipe :

```

{
  "recipId": 4,
  "recipeName": "tava ızgara",
  "creationDate": "05-01-2022 16:01",
  "veganYN": "N",
  "suitableCount": 23,
  "recipeIngredientsList": [
    {
      "ingredient_id": 4,
      "ingredientName": "gdfgdfgfdgdg"
    }
  ],
  "recipeInstructionModelList": [
    {
      "instruction_id": 4,
      "instructionDetail": "fdfsfdsf"
    }
  ]
}

```

If we give a recipe parameter that is not registered on the system before, the system result will throw a recipe that could not be found exception like below.

Example get request: <http://localhost:8080/api/v1/recipe/10>

The screenshot shows a Postman interface with the following details:

- Method:** GET
- URL:** http://localhost:8080/api/v1/recipe/10
- Headers:** Authorization (Bearer eyJhbGciOiJIUzUxMiJ9eyJzdWlIiOjKZW1vliwiZxhwijox...), Content-Type (application/json)
- Status:** 404 Not Found
- Body (Pretty):**

```
1 {  
2   "returnMessage": "FAILED",  
3   "returnCode": "0003",  
4   "details": "Recipe could not be found!"  
5 }
```

## 2. /api/v1/recipe/create

Future: creates a recipe for our application by using this endpoint.

Endpoint : <http://localhost:8080/api/v1/recipe/create>

Type : POST

Parameter :

```
{  
  "creationDate": "string",  
  "recipieId": 0,  
  "recipeIngredientsList": [  
    {  
      "ingredientName": "string",  
      "ingredient_id": 0  
    }  
  ],  
  "recipeInstructionModelList": [  
    {  
      "instructionDetail": "string",  
      "instruction_id": 0  
    }  
  ],
```

```
        "recipeName": "string",
        "suitableCount": 0,
        "veganYN": "string"
    }
```

Example post request: <http://localhost:8080/api/v1/recipe/create>

request body :

```
{
    "recipeName": "fresh bean",
    "veganYN": "Y",
    "recipeIngredientsList": [
        {
            "ingredientName": "fresh bean"
        },
        {
            "ingredientName": "pepper"
        },
        {
            "ingredientName": "onion"
        }
    ],
    "recipeInstructionModelList": [
        {
            "instructionDetail": "hot up onion"
        },
        {
            "instructionDetail": "add bean"
        },
        {
            "instructionDetail": "put on pepper"
        }
    ]
}
```

As result like below:

```
1 {
2     "recipeId": 5,
3     "recipeName": "taze fasulye",
4     "creationDate": "05-01-2022 23:12",
5     "veganYN": "y",
6     "suitableCount": 10,
7     "recipeIngredientsList": [
8         {
9             "ingredient_id": 5,
10            "ingredientName": "taze fasulye"
11        },
12        {
13            "ingredient_id": 6,
14            "ingredientName": "soğan"
15        },
16        {
17            "ingredient_id": 7,
18            "ingredientName": "salça"
19        }
20    ],
21    "recipeInstructionModelList": [
22        {
23            "instruction_id": 5,
24            "instructionDetail": "soğanları kavur"
25        },
26        {
27            "instruction_id": 6,
28            "instructionDetail": "taze fasulye ekle"
29        },
30        {
31            "instruction_id": 7,
32            "instructionDetail": "salça at"
33        }
34    ]
35 }
```

### 3. /api/v1/recipe/allrecipes

Future: performs to get all recipes

Endpoint : <http://localhost:8080/api/v1/recipe/allrecipes>

Type : GET

Parameter : No parameter

Example get request: <http://localhost:8080/api/v1/recipe/allrecipes>

! You must use v7.0 or higher to access your workspaces and collections. See what's new

### GENERATE CODE SNIPPETS

HTTP

Copy to Clipboard

```
1 GET /api/v1/recipe/allrecipes HTTP/1.1
2
3 Host: localhost:8080
4
5 Content-Type: application/json
6
7 Authorization: Bearer eyJhbGciOiJIUzUxMiJ9
    .eyJzdWIiOiJkZW1vIiwiZXhwIjoxNjQxNDI5ODU1LCJyb2xlcyI6W3siYXV0aG9yaXR5IjoiQU
    RNSU4ifV19.IxFZ_iC56m0ronjyqpqCQ11VGj0WIIsGCV0Z3ks8rldmGP33FE5TgG9Dd-PTKlk
    -LPBq6B5Ag8E0CI9uKUGZX_Q
8
9 Cache-Control: no-cache
10
11
12
```

Push request and we can see all of the recipes on the system.

GET <http://localhost:8080/api/v1/recipe/allrecipes>

Headers (2)		Body	Pre-request Script	Tests	Code
<input checked="" type="checkbox"/> Key	Value				Description
<input checked="" type="checkbox"/> Content-Type	application/json				Bulk Edit
<input checked="" type="checkbox"/> Authorization	Bearer eyJhbGciOiJIUzIwMjQ...eyJzdWliOijKZW1vliwiZhwlijoxNjQxNDI5OD...				Presets ▾
New key		Value			Description

Body (1) Headers (11) Test Results Status: 200 OK Time: 419 ms

Pretty Raw Preview JSON

```

1 [
2   {
3     "recipeId": 1,
4     "recipename": "deneme",
5     "creationDate": "05-01-2022 15:25",
6     "veganYN": "Y",
7     "suitableCount": 20,
8     "recipeIngredientsList": [
9       {
10         "ingredient_id": 1,
11         "ingredientName": "fsdfsdfs"
12       }
13     ],
14     "recipeInstructionModelList": [
15       {
16         "instruction_id": 1,
17         "instructionDetail": "add"
18       }
19     ],
20   },
21   {
22     "recipeId": 2,
23     "recipename": "sdasdaa",
24     "creationDate": "05-01-2022 15:49",
25     "veganYN": "N",
26     "suitableCount": 0,
27     "recipeIngredientsList": [

```

#### 4. /api/v1/recipe/delete/{recipeId}

Future: performs to remove that given recipe id.

Endpoint : <http://localhost:8080/api/v1/recipe/delete/{recipeId}>

Type : DELETE

Parameter :

```
{
  name : recipelid
  type : integer
}
```

Example delete request:<http://localhost:8080/api/v1/recipe/delete/4>

HTTP

Copy to Clipboard

```
1 DELETE /api/v1/recipe/delete/2 HTTP/1.1
2
3 Host: localhost:8080
4
5 Content-Type: multipart/form-data; boundary=----WebKitFormBoundary7MA4YWxkTrZu0gW
6
7 Authorization: Bearer eyJhbGciOiJIUzUxMiJ9
8     .eyJzdWIiOiJkZW1vIiwiZXhwIjoxNjQxNDI5ODU1LCJyb2xlcyI6W3siYXV0aG9yaXR5IjoiQU
9     RNSU4ifV19.IxZF_iC56m0ronjyqpqCQ11VGj0WIIsGCV0Z3ks8rldmGP33FE5TgG9Dd-PTKlk
10    -LPBq6B5Ag8E0CI9uKUGZX_Q
11
12 Cache-Control: no-cache
13 Postman-Token: b8d0c993-013b-ce98-121b-1226046cfb2d
14
15 -----WebKitFormBoundary7MA4YWxkTrZu0gW--
```

Push the delete request and we can see the response ok message.

The screenshot shows the Postman interface with a successful API call. The request method is 'DELETE' to the URL 'http://localhost:8080/api/v1/recipe/delete/5'. The 'Headers' tab is selected, showing two headers: 'Content-Type' set to 'application/json' and 'Authorization' set to a valid JWT token. The response status is '204 No Content' with a time of '113 ms'. The JSON response pane is empty.

If we give a recipe parameter that is not registered on the system before, the system result will throw a recipe that could not be found exception like below.

Example delete request: <http://localhost:8080/api/v1/recipe/delete/10>

The screenshot shows the Postman interface with a failed API call. The request method is 'DELETE' to the URL 'http://localhost:8080/api/v1/recipe/delete/10'. The 'Headers' tab is selected, showing 'Content-Type' as 'application/json' and 'Authorization' as a valid JWT token. The response status is '404 Not Found' with a time of '410 ms'. The JSON response pane displays an error message: { "returnMessage": "FAILED", "resultCode": "0003", "details": "Recipe could not be found!" }.

## 5. /api/v1/recipe/update/{recipId}

Future: performs to update that given recipe object.

Endpoint : <http://localhost:8080/api/v1/recipe/update/{recipId}>

Type : PUT

Parameter :

Request Path Parameter:

```
{  
    name : recipId  
    type : integer  
}
```

Request Body Parameter:

```
recipeDto:  
{  
    "recipeIngredientsList": [  
        {  
            "ingredientName": "string",  
            "ingredient_id": integer  
        }  
    ],  
    "recipeInstructionModelList": [  
        {  
            "instructionDetail": "string",  
            "instruction_id":integer  
        }  
    ],  
    "recipeName": "string",  
    "suitableCount": integer,  
    "veganYN": "string"  
}
```

Example put request:<http://localhost:8080/api/v1/recipe/update/1>

HTTP ▾

Copy to Clipboard

```
1 Content-Type: application/json
2 Authorization: Bearer eyJhbGciOiJIUzUxMiJ9
   .eyJzdWIiOiJkZW1vIiwiZXhwIjoxNjQxNDI5ODU1LCJyb2xlcyI6W3siYXV0aG9yaXR5IjoiQURNSU4if
   V19.IxZF_iC56m0ronjyqpqCQ11VGj0WIIsGCVOZ3ks8rlmGP33FE5TgG9Dd-PTK1k
   -LPBq6B5Ag8E0CI9uKUGZX_Q
3 Cache-Control: no-cache
4 Postman-Token: 71fb3fb4-76e9-abe9-7bc4-7f3249476a0f
5
6 {
7     "recipeId": 1,
8     "recipeName": "deneme",
9     "creationDate": "05-01-2022 15:25",
10    "veganYN": "Y",
11    "suitableCount": 20,
12    "recipeIngredientsList": [
13        {
14            "ingredient_id": 1,
15            "ingredientName": "fsdfsfdsx"
16        }
17    ],
18    "recipeInstructionModelList": [
19        {
20            "instruction_id": 1,
21            "instructionDetail": "add"
22        }
23    ]
24 }
```

Push the put request and we can see the response ok message and updated recipe object.

PUT ▾ http://localhost:8080/api/v1/recipe/update/1

Params Send Save

```
17
18 ]
19 }
```

Body Cookies (1) Headers (11) Test Results Status: 200 OK Time: 83 ms

Pretty Raw Preview JSON

```
1 {
2     "recipeId": 1,
3     "recipeName": "deneme",
4     "creationDate": "05-01-2022 15:25",
5     "veganYN": "Y",
6     "suitableCount": 20,
7     "recipeIngredientsList": [
8         {
9             "ingredient_id": 1,
10            "ingredientName": "fsdfsfdsx"
11        }
12    ],
13    "recipeInstructionModelList": [
14        {
15            "instruction_id": 1,
16            "instructionDetail": "add"
17        }
18    ]
19 }
```

## 6. /api/v1/recipe/ingredients/{recipId}

Future: performs to get all recipe ingredients that given recipe id.

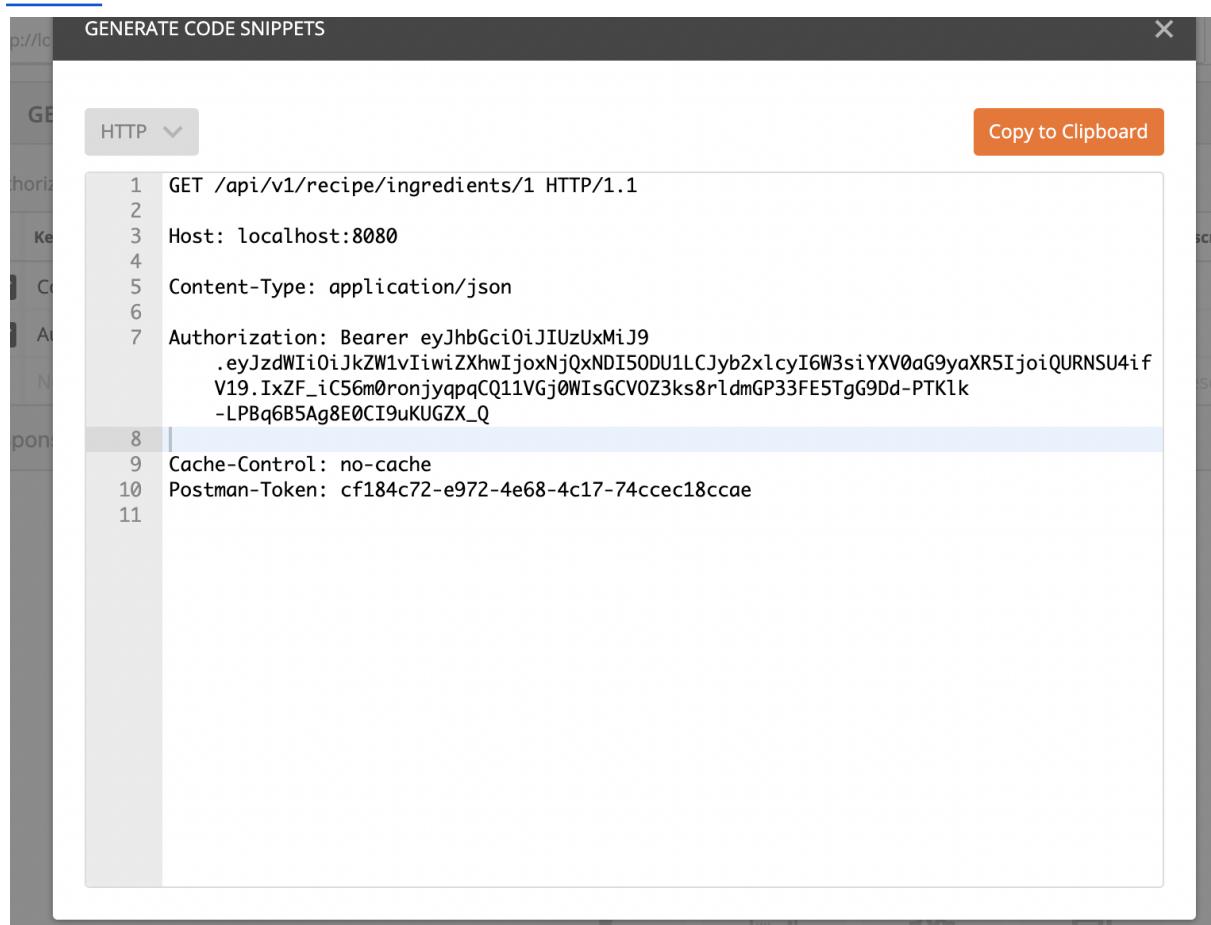
Endpoint : <http://localhost:8080/api/v1/recipe/ingredients/{recipId}>

Type : GET

Parameter :

```
{  
    name : recipId  
    type : integer  
}
```

Example get request: <http://localhost:8080/api/v1/recipe/ingredients/1>



The screenshot shows the 'GENERATE CODE SNIPPETS' dialog in Postman. The tab 'HTTP' is selected. The code snippet is as follows:

```
1 GET /api/v1/recipe/ingredients/1 HTTP/1.1  
2  
3 Host: localhost:8080  
4  
5 Content-Type: application/json  
6  
7 Authorization: Bearer eyJhbGciOiJIUzUxMiJ9  
.eyJzdWIiOiJkZW1vIiwidXNjQxNDI5ODU1LCJyb2xlcyI6W3siYXV0aG9yaXR5IjoQURNSU4if  
V19.IxZF_iC56m0ronjyqpqCQ11VGj0WIIsGCV0Z3ks8rldmGP33FE5TgG9Dd-PTK1k  
-LPBq6B5Ag8E0CI9uKUGZX_Q  
8  
9 Cache-Control: no-cache  
10 Postman-Token: cf184c72-e972-4e68-4c17-74ccce18ccae  
11
```

A red button labeled 'Copy to Clipboard' is visible in the top right corner of the dialog.

If you fetch this request, the result will be shown below.

## 7. /api/v1/recipe/instructions/{recipId}

Future: performs to get all recipe ingredients that given recipe id.

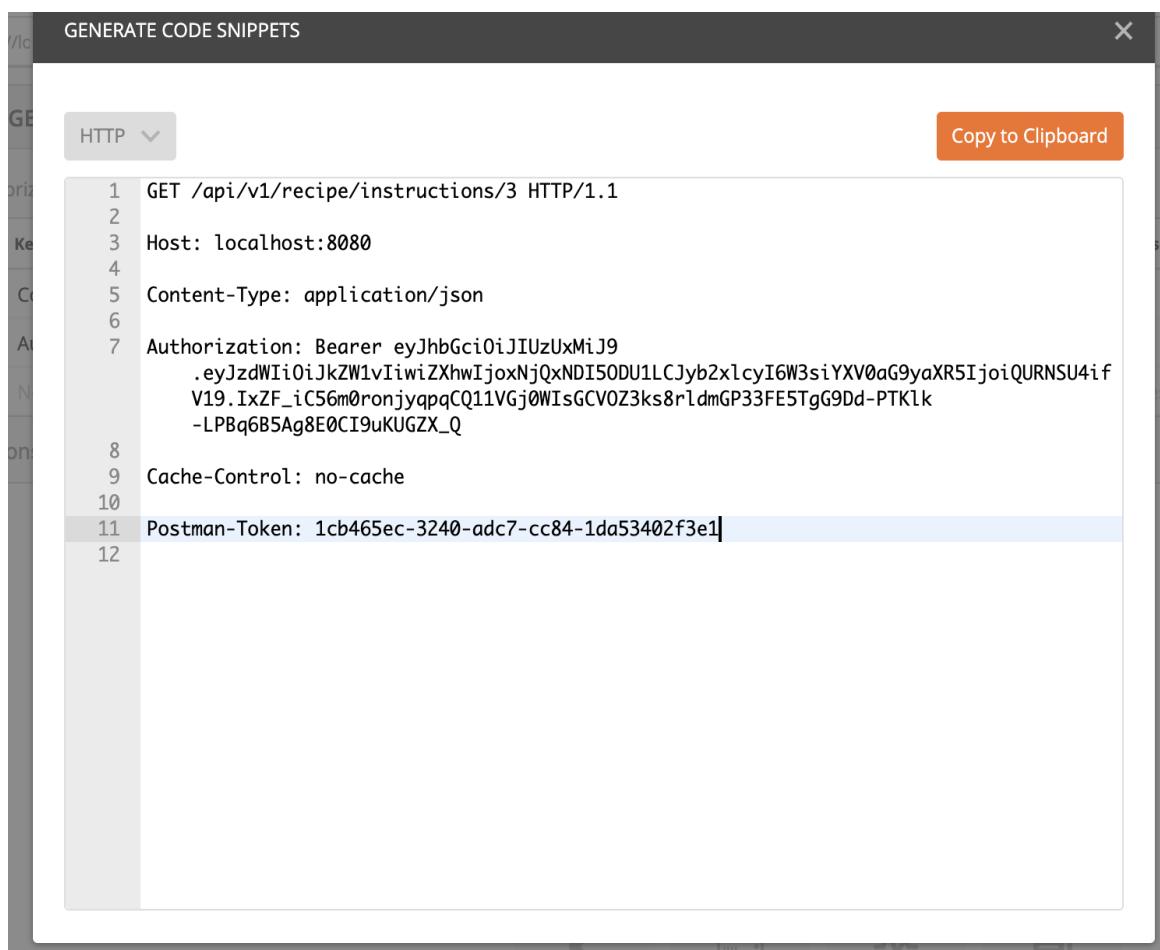
Endpoint : <http://localhost:8080/api/v1/recipe/instructions/{recipId}>

Type : GET

Parameter :

```
{  
    name : recipId  
    type : integer  
}
```

Example get request: <http://localhost:8080/api/v1/recipe/instructions//1>



The screenshot shows the 'GENERATE CODE SNIPPETS' dialog in Postman. The method is set to 'HTTP'. The code snippet is as follows:

```
1 GET /api/v1/recipe/instructions/3 HTTP/1.1  
2  
3 Host: localhost:8080  
4  
5 Content-Type: application/json  
6  
7 Authorization: Bearer eyJhbGciOiJIUzUxMiJ9  
.eyJzdWIiOiJkZW1vIiwiZXhwIjoxNjQxNDI5ODU1LCJyb2xlcyI6W3siYXV0aG9yaXR5IjoiQURNSU4if  
V19.IxZF_iC56m0ronjyqpqCQ11VGj0WIsgCV0Z3ks8rldmGP33FE5TgG9Dd-PTKlk  
-LPBq6B5Ag8E0CI9uKUGZX_Q  
8  
9 Cache-Control: no-cache  
10  
11 Postman-Token: 1cb465ec-3240-adc7-cc84-1da53402f3e1|  
12
```

A 'Copy to Clipboard' button is visible in the top right corner of the dialog.

If you fetch this request, the result will be shown below.

The screenshot shows the Postman application interface. At the top, there is a header bar with 'GET' selected, the URL 'http://localhost:8080/api/v1/recipe/instructions/1', and buttons for 'Send', 'Save', and 'Code'. Below the header, the 'Headers' tab is active, showing two entries: 'Content-Type' with value 'application/json' and 'Authorization' with value 'Bearer eyJhbGciOiJIUzUxMiJ9eyJzdWIiOiJkZW1vliwiZXhwIjox...'. There is also a 'New key' row for adding more headers. Below the headers, the 'Body' tab is active, showing a JSON response. The response is a single object with an array of instructions:

```
1 - [  
2 -   {  
3 -     "instruction_id": 1,  
4 -     "instructionDetail": "add"  
5 -   }  
6 - ]
```

At the bottom right of the body panel, there are status and time indicators: 'Status: 200 OK' and 'Time: 185 ms'.

