FINAL PROJECT REPORT

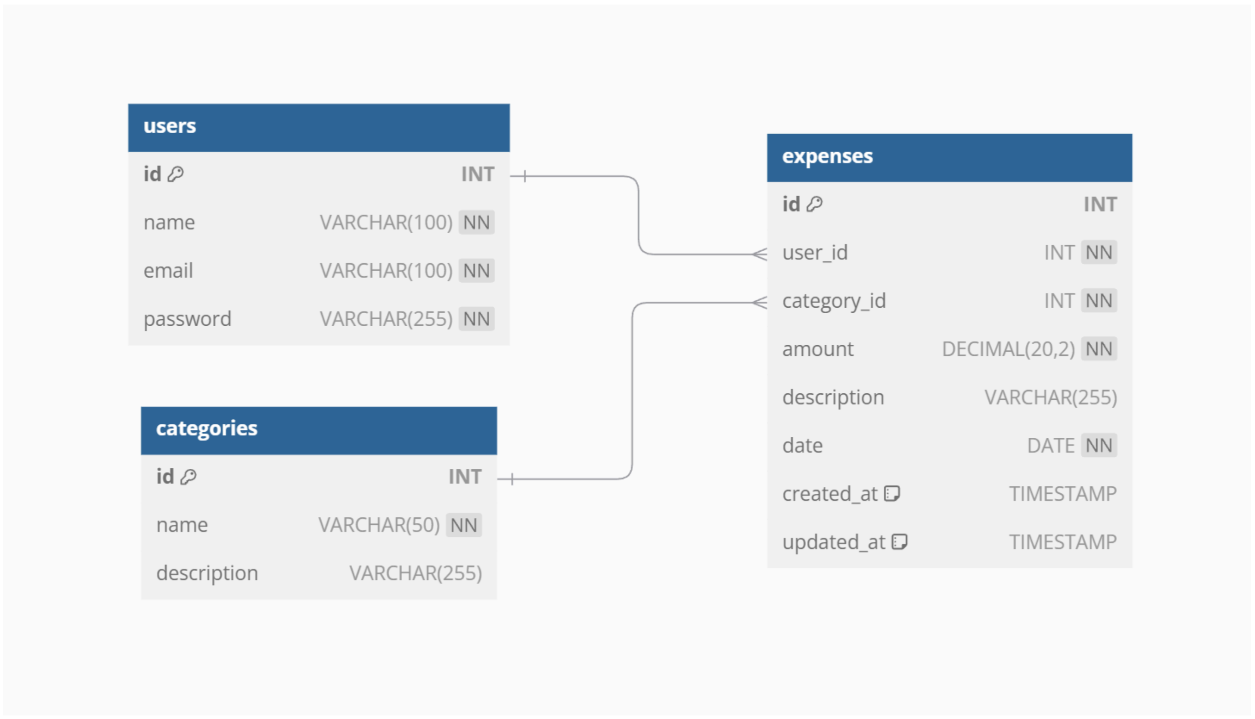
SEMESTER 1, ACADEMIC YEAR: 2024-2025

CT313H: WEB TECHNOLOGIES AND SERVICES

- **Project/Application name: Financial Management**
- **GitHub links:** **https://github.com/24-25Sem1-Courses/ct313h01-project-tnxk19**
- **Youtube video: https://www.youtube.com/watch?v=YN8rD6TT7eQ**
- **Student ID 1: B2111985**
- **Student Name 1: Tran Nguyen Xuan Khanh**
- **Student ID 2: B2014914**
- **Student Name 2: Luu Thai Hoa**
- **Class/Group Number : CT313HM01**

I. Introduction

- Project/application description: A financial management website that can help users manage their expenses.
- A list of tables and their structures in the database (could show a CDM/PDM diagram here).



users		
id	INT	
name	VARCHAR(100)	NN
email	VARCHAR(100)	NN
password	VARCHAR(255)	NN

categories		
id	INT	
name	VARCHAR(50)	NN
description	VARCHAR(255)	

expenses		
id	INT	
user_id	INT	NN
category_id	INT	NN
amount	DECIMAL(20,2)	NN
description	VARCHAR(255)	
date	DATE	NN
created_at	TIMESTAMP	
updated_at	TIMESTAMP	

- A task assignment sheet for each member if working in groups.
  BackEnd:

Student code	Student name	Task
B2014914	Luu Thai Hoa	Register, login CRUD for Expense
B2111985	Tran Nguyen Xuan Khanh	Init the project CRUD for Category Database design

FrontEnd:

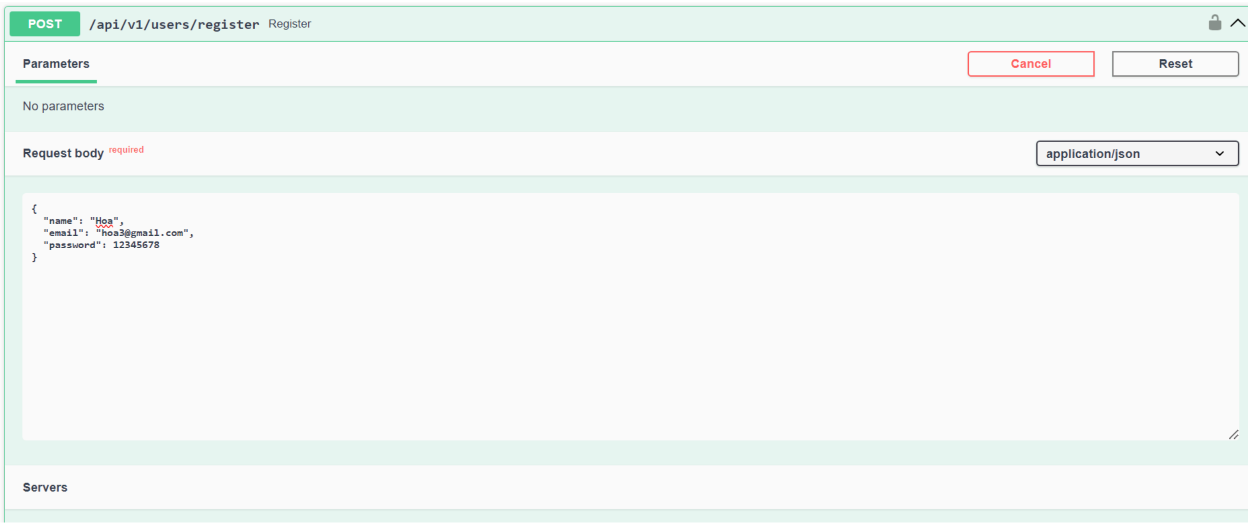
Student code	Student name	Task
B2014914	Luu Thai Hoa	Login, Signup, Logout and fix bug (if necessary)
B2111985	Tran Nguyen Xuan Khanh	Add Expense, Edit Expense, Homepage(includes CategoryList and CategoryHistory)

II. Details of implemented features

Back-end:

1. **Feature : Register**

- **Description:** this is a registering backend in swagger. It is used for registering a new account.
- **Screenshots:**



POST /api/v1/users/register Register

Parameters

Cancel Reset

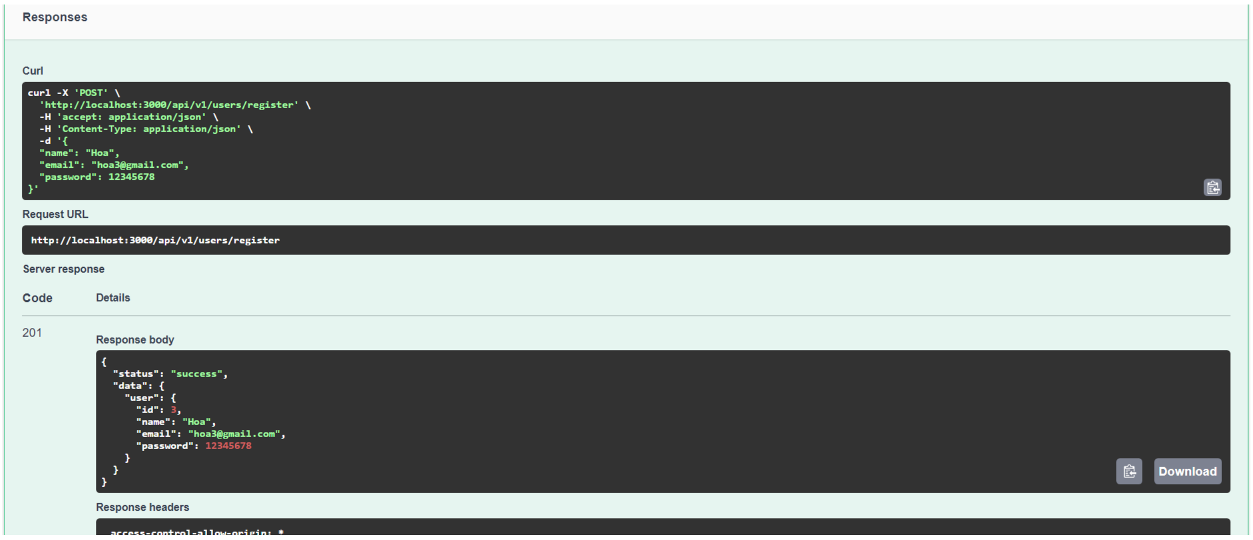
No parameters

Request body required

application/json

```
{
  "name": "Hoa",
  "email": "hoa3@gmail.com",
  "password": 12345678
}
```

Servers



Responses

Curl

```
curl -X 'POST' \
  'http://localhost:3000/api/v1/users/register' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Hoa",
  "email": "hoa3@gmail.com",
  "password": 12345678
}'
```

Request URL

```
http://localhost:3000/api/v1/users/register
```

Server response

Code	Details
201	Response body

```
{
  "status": "success",
  "data": {
    "user": {
      "id": 3,
      "name": "Hoa",
      "email": "hoa3@gmail.com",
      "password": 12345678
    }
  }
}
```

Download

Response headers

```
access-control-allow-origin: *
```

- **Implementation details:**
  + Endpoint : POST users/register
  + Data structure sent:

```
{
  "name": "Hoa",
  "email": "hoa3@gmail.com",
  "password": 12345678
}
```

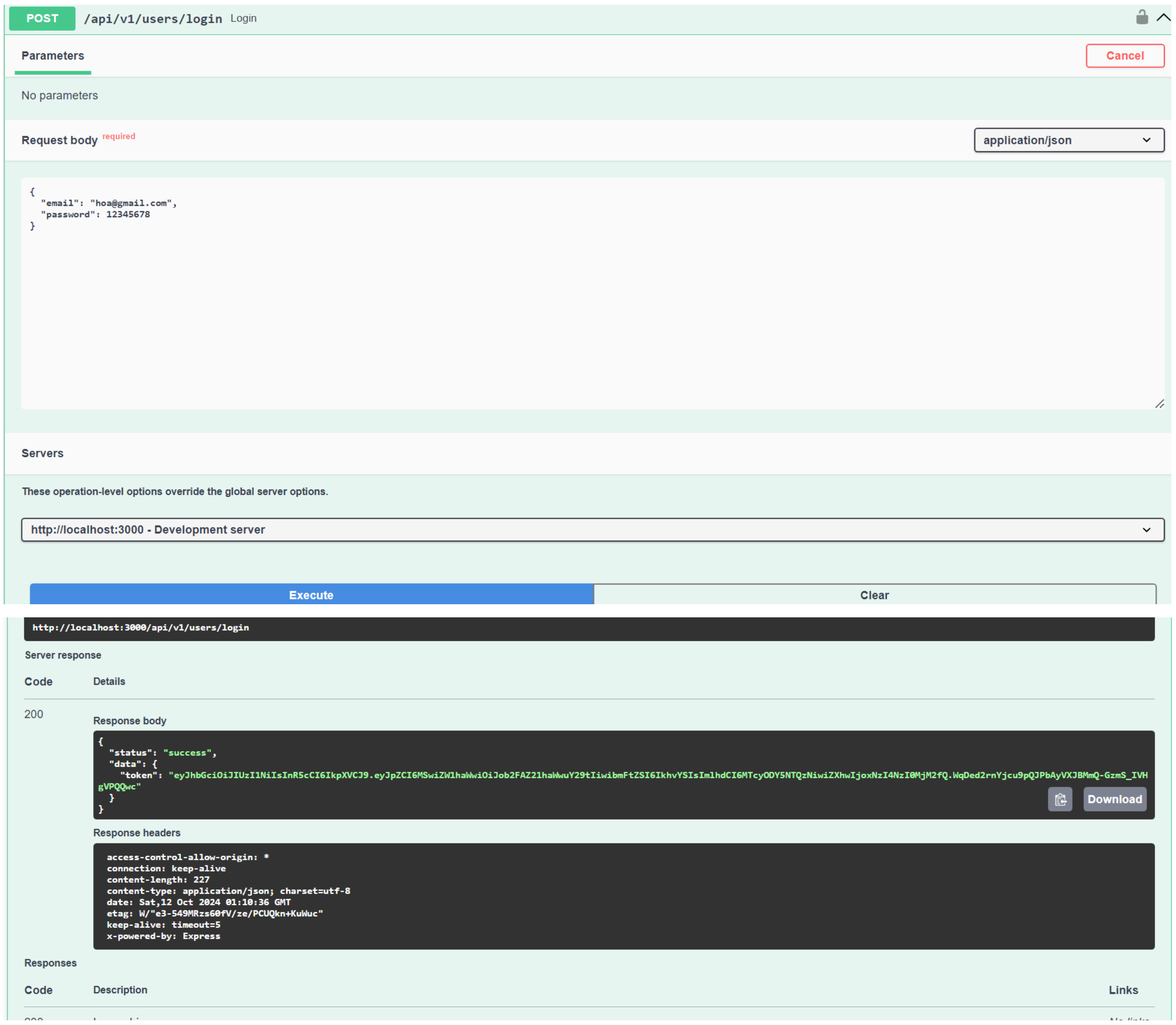
  + Data structure received:

```
{
  "status": "success",
  "data": {
    "user": {
      "id": 3,
      "name": "Hoa",
"email": "hoa3@gmail.com",
"password": 12345678
}
}
}
```

+ This feature reads and stores data in the users table.
+ This feature need to be implemented in authentication state

2. Feature : Login

- **Description:** This feature is used for login and provides users a token with an expiration date.
- **Screenshots:**



```
{
  "email": "hoa@gmail.com",
  "password": 12345678
}
```

```
http://localhost:3000/api/v1/users/login
```

Server response

Code | Details

200 | Response body

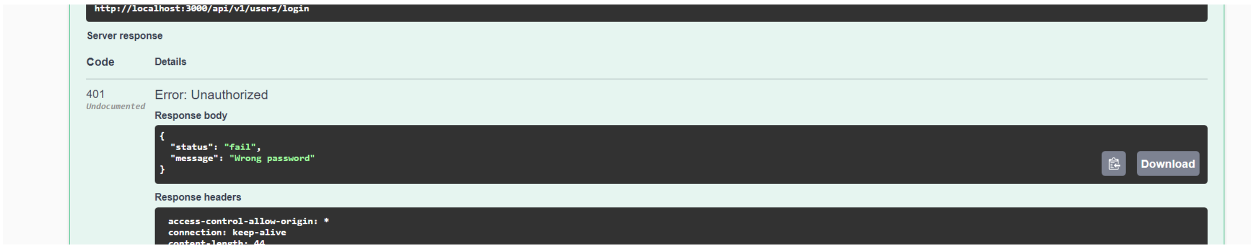
```
{
  "status": "success",
  "data": {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiZW1haWwiOiJob2FAZ21haWwuY29tIiwibmFtZSI6IkhvYSIsImlhdCI6MTcyODY5NTQzNiwiZXhwIjoxNzI4NzI0MjM2fQ.WqDed2rnYjcu9pQJPbAyVXJBMmQ-GzmS_IVHgVPQQwc"
  }
}
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 227
content-type: application/json; charset=utf-8
date: Sat,12 Oct 2024 01:10:36 GMT
etag: W/"e3-549MRzs60fV/ze/PCUQkn+KuWuc"
keep-alive: timeout=5
x-powered-by: Express
```

Responses

Code | Description | Links



```
http://localhost:3000/api/v1/users/login
```

Server response

Code Details

401 *Undocumented* Error: Unauthorized

Response body

```
{
  "status": "fail",
  "message": "Wrong password"
}
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
```

- **Implementation details:**
  + Endpoint: POST users/login
  + Data structure sent:

    ```
    {
      "email": "hoa@gmail.com",
      "password": 12345678
    }
    ```
  + Data structure received:

    If the email and password is correct:

    ```
    {
      "status": "success",
      "data": {
        "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiZW1haWwiOiJob2FAZ21haWwuY29tIiwibmFtZSI6IkhvYSIsImlhdCI6MTcyODY5NTQzNiwiZXhwIjoxNzI4NzI0MjM2fQ.WqDed2rnYjcu9pQJPbAyVXJBMmQ-GzmS_IVHgVPQQwc"
      }
    }
    ```

    If the email or password is incorrect:

    Wrong email:

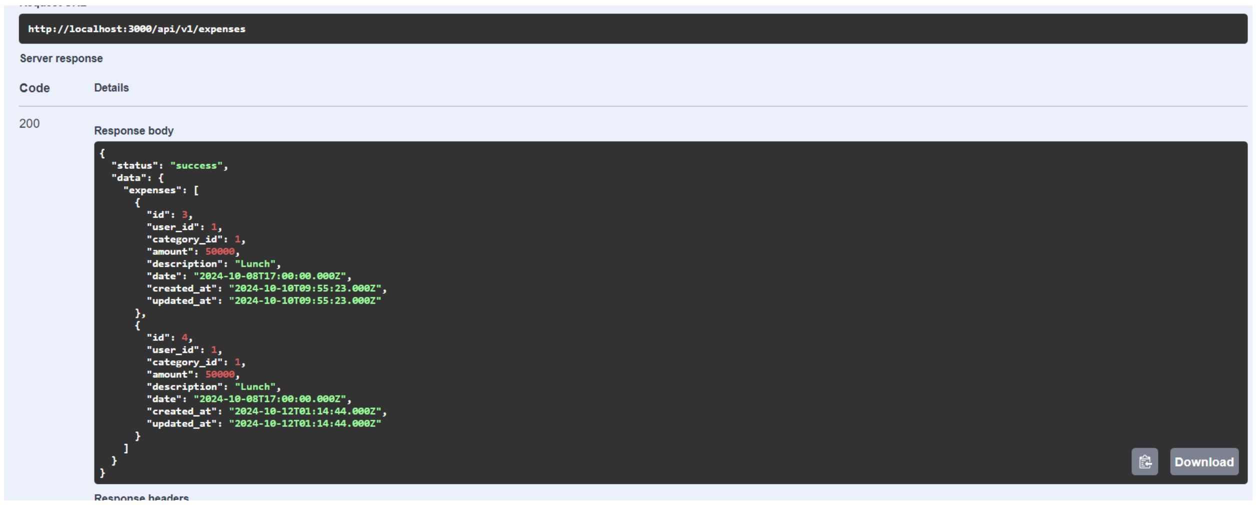
    ```
    {
      "status": "fail",
      "message": "Wrong email"
    }
    ```

    Wrong password:

    ```
    {
      "status": "fail",
      "message": "Wrong password"
    }
    ```
  + This feature read and store data in the database
  + This feature need to be implemented in authentication state

3. **Feature : get expenses**

- **Description:** This feature is used for authenticated users to get all their expenses.
- **Screenshots:**



```
http://localhost:3000/api/v1/expenses
```

Server response

Code Details

200

Response body

```
{
  "status": "success",
  "data": {
    "expenses": [
      {
        "id": 3,
        "user_id": 1,
        "category_id": 1,
        "amount": 50000,
        "description": "Lunch",
        "date": "2024-10-08T17:00:00.000Z",
        "created_at": "2024-10-10T09:55:23.000Z",
        "updated_at": "2024-10-10T09:55:23.000Z"
      },
      {
        "id": 4,
        "user_id": 1,
        "category_id": 1,
        "amount": 50000,
        "description": "Lunch",
        "date": "2024-10-08T17:00:00.000Z",
        "created_at": "2024-10-12T01:14:44.000Z",
        "updated_at": "2024-10-12T01:14:44.000Z"
      }
    ]
  }
}
```

Response headers

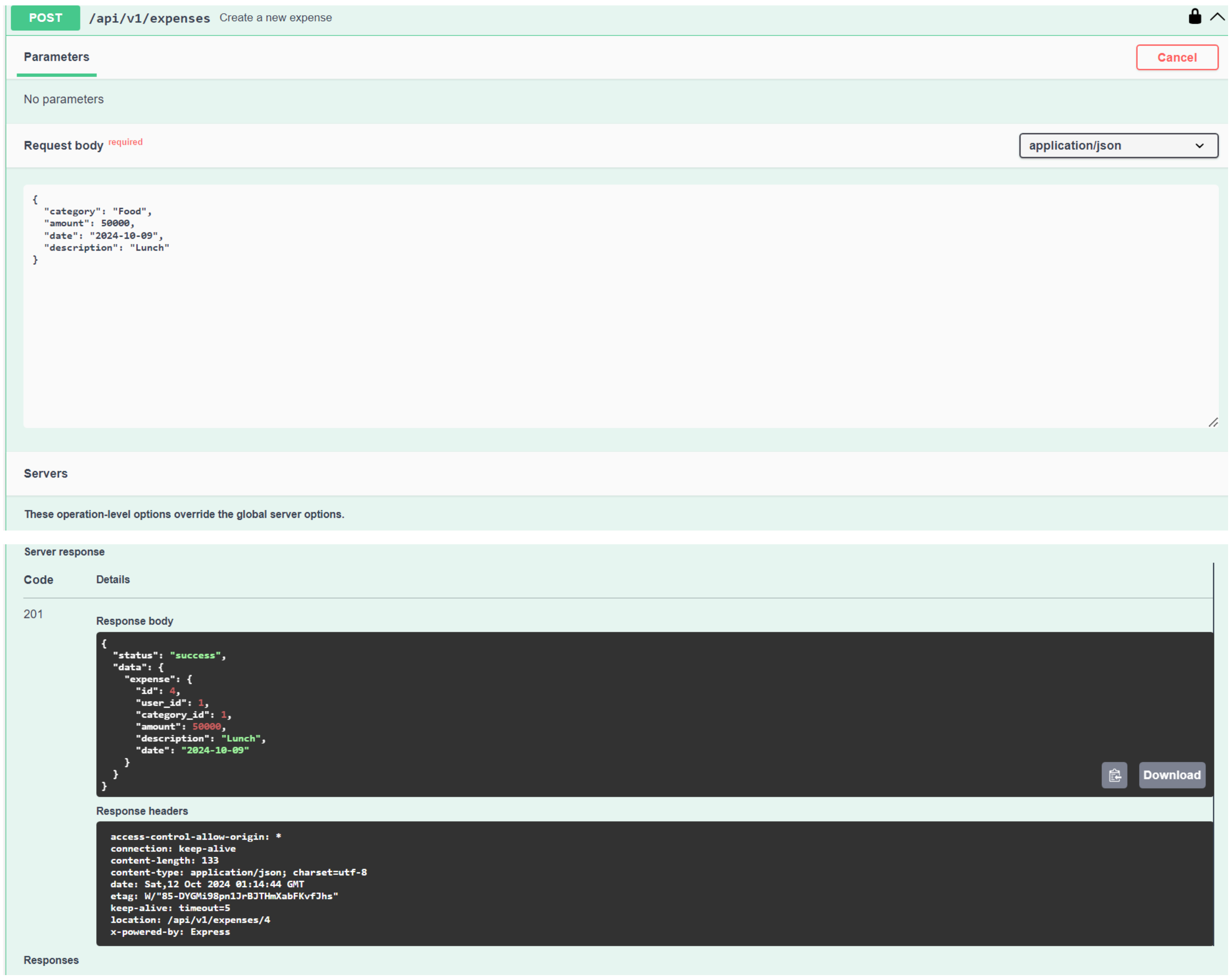
- **Implementation details:**
  + Endpoint: GET expenses
  + Data structure sent:
    Header : JWT_TOKEN
  + Data structure received:

```
{
  "status": "success",
  "data": {
    "expenses": [
      {
        "id": 3,
        "user_id": 1,
        "category_id": 1,
        "amount": 50000,
        "description": "Lunch",
        "date": "2024-10-08T17:00:00.000Z",
        "created_at": "2024-10-10T09:55:23.000Z",
        "updated_at": "2024-10-10T09:55:23.000Z"
      },
      {
        "id": 4,
        "user_id": 1,
        "category_id": 1,
        "amount": 50000,
        "description": "Lunch",
        "date": "2024-10-08T17:00:00.000Z",
        "created_at": "2024-10-12T01:14:44.000Z",
        "updated_at": "2024-10-12T01:14:44.000Z"
      }
    ]
  }
}
```

+ This feature read data in the database
+ This feature need to be implemented in the UI of users

4. Feature : post expenses

- **Description:** this feature is used for authenticated users to post expenses.
- **Screenshots:**



```
{
  "category": "Food",
  "amount": 50000,
  "date": "2024-10-09",
  "description": "Lunch"
}
```

Servers
These operation-level options override the global server options.
Server response
Code
Details
201
Response body

```
{
  "status": "success",
  "data": {
    "expense": {
      "id": 4,
      "user_id": 1,
      "category_id": 1,
      "amount": 50000,
      "description": "Lunch",
      "date": "2024-10-09"
    }
  }
}
```

```
access-control-allow-origin: *
connection: keep-alive
content-length: 133
content-type: application/json; charset=utf-8
date: Sat,12 Oct 2024 01:14:44 GMT
etag: W/"85-DYGMi98pn1JrBJTHmXabFKvfJhs"
keep-alive: timeout=5
location: /api/v1/expenses/4
x-powered-by: Express
```

- **Implementation details:**
  + Endpoint: POST expenses
  + Data structure sent:
    Header : JWT_TOKEN

```
{
 "category": "Food",
 "amount": 50000,
 "date": "2024-10-09",
 "description": "Lunch"
}
```

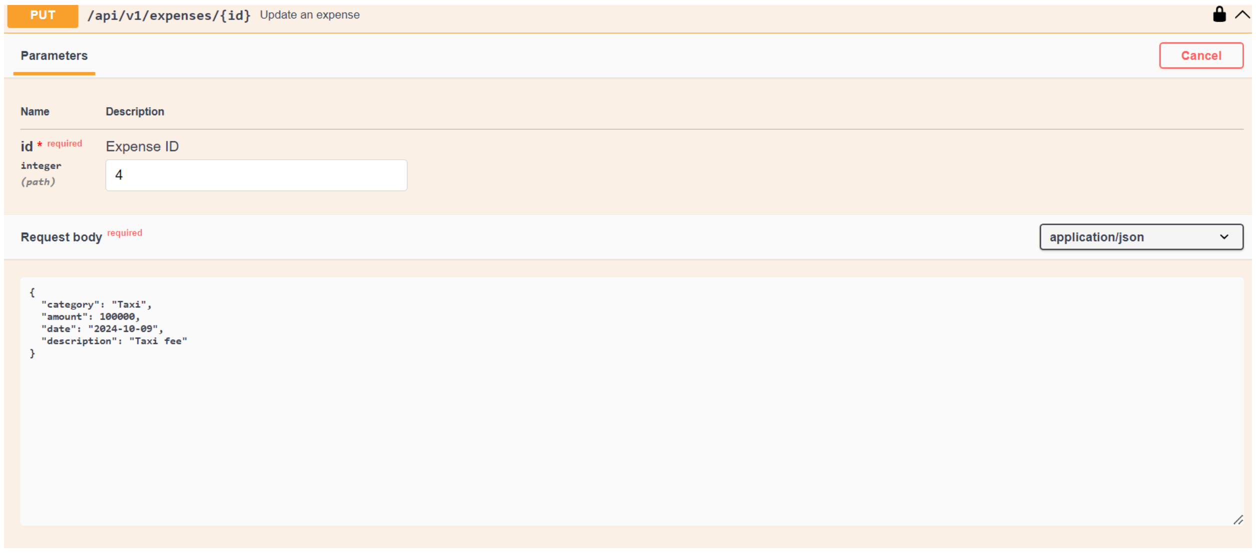
+ Data structure received:

```
{
 "status": "success",
 "data": {
  "expense": {
   "id": 4,
   "user_id": 1,
   "category_id": 1,
   "amount": 50000,
   "description": "Lunch",
   "date": "2024-10-09"
  }
 }
}
```

+ This feature read and store in database
+ This feature need to be implemented in the UI of users

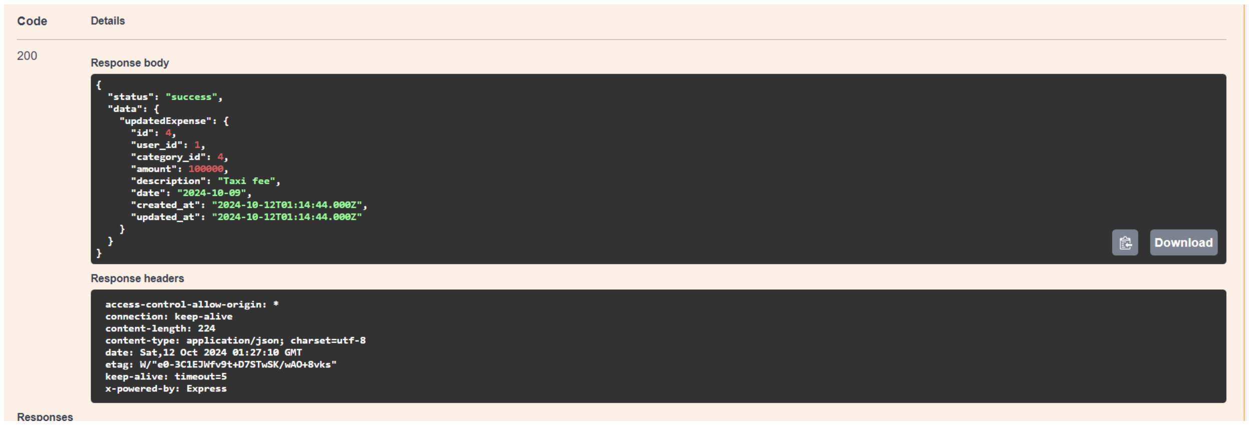
5. Feature : PUT expenses

- **Description:** this feature is used for authenticated users to update their expenses.
- **Screenshots:**



id: 4

```
{
  "category": "Taxi",
  "amount": 100000,
  "date": "2024-10-09",
  "description": "Taxi fee"
}
```


Code Details

200

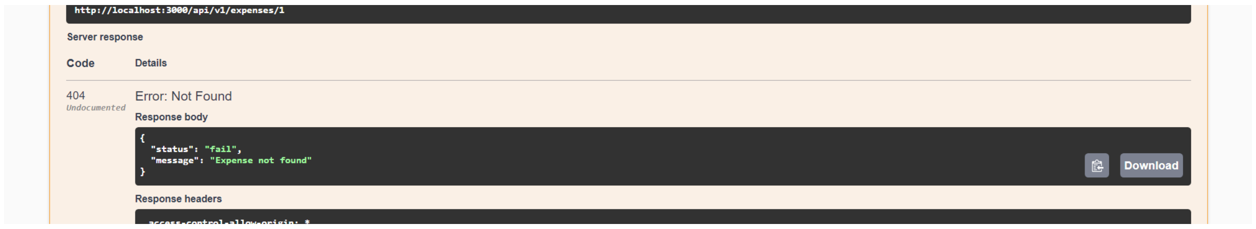
Response body

```
{
  "status": "success",
  "data": {
    "updatedExpense": {
      "id": 4,
      "user_id": 1,
      "category_id": 4,
      "amount": 100000,
      "description": "Taxi fee",
      "date": "2024-10-09",
      "created_at": "2024-10-12T01:14:44.000Z",
      "updated_at": "2024-10-12T01:14:44.000Z"
    }
  }
}
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 224
content-type: application/json; charset=utf-8
date: Sat,12 Oct 2024 01:27:10 GMT
etag: W/"e0-3C1EJWfv9t+D7STwSK/wAO+8vks"
keep-alive: timeout=5
x-powered-by: Express
```

Responses


```
http://localhost:3000/api/v1/expenses/1
```

Server response

Code Details

404 *Undocumented* Error: Not Found

Response body

```
{
  "status": "fail",
  "message": "Expense not found"
}
```

Response headers

```
access-control-allow-origin: *
```

- **Implementation details:**
  + Endpoint: PUT expenses/{id}
  + Data structure sent:

    Header : JWT_TOKEN

    ```
    {
      "category": "Taxi",
      "amount": 100000,
      "date": "2024-10-09",
      "description": "Taxi fee"
    }
    ```
  + Data structure received:

    If found expense:

    ```
    {
      "status": "success",
      "data": {
        "updatedExpense": {
          "id": 4,
          "user_id": 1,
          "category_id": 4,
          "amount": 100000,
          "description": "Taxi fee",
          "date": "2024-10-09",
          "created_at": "2024-10-12T01:14:44.000Z",
          "updated_at": "2024-10-12T01:14:44.000Z"
    ```

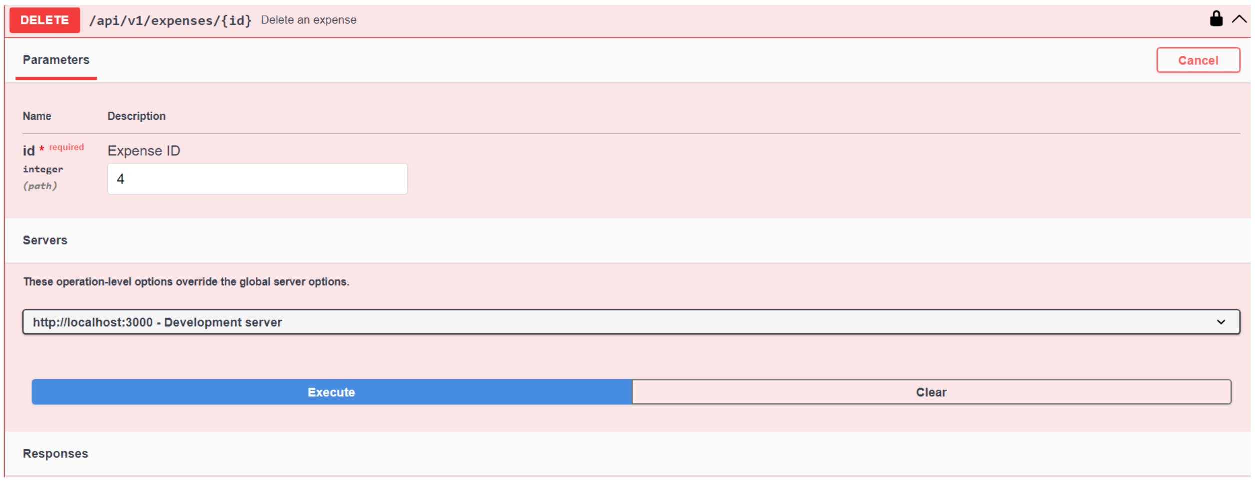
```
      }
    }
  }
```
If not:

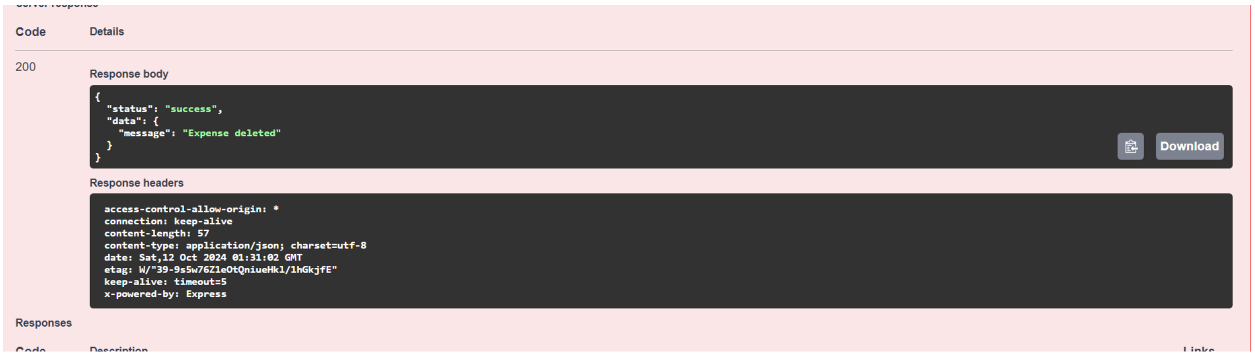
```
{
  "status": "fail",
  "message": "Expense not found"
}
```

+ This feature read and store in database
+ This feature need to be implemented in the UI of user

6. Feature : Delete expenses

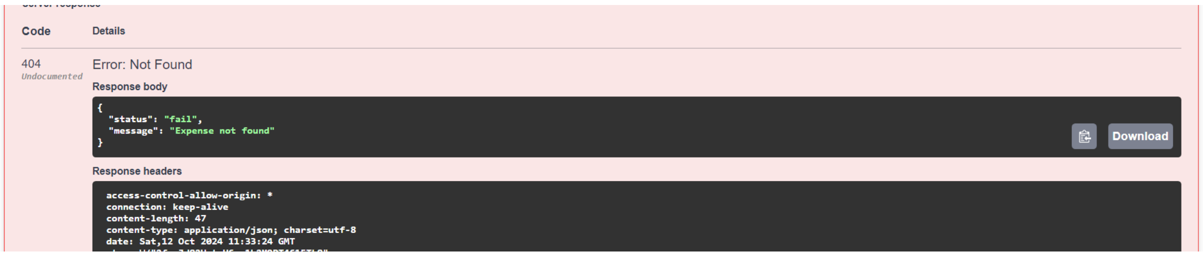
- **Description:** this feature is used for authenticated users to delete their expenses.
- **Screenshots:**





```
{
  "status": "success",
  "data": {
    "message": "Expense deleted"
  }
}
```

```
access-control-allow-origin: *
connection: keep-alive
content-length: 57
content-type: application/json; charset=utf-8
date: Sat,12 Oct 2024 01:31:02 GMT
etag: W/"39-9s5w76Z1eOtQniueHkl/1hGkjfE"
keep-alive: timeout=5
x-powered-by: Express
```



Error: Not Found

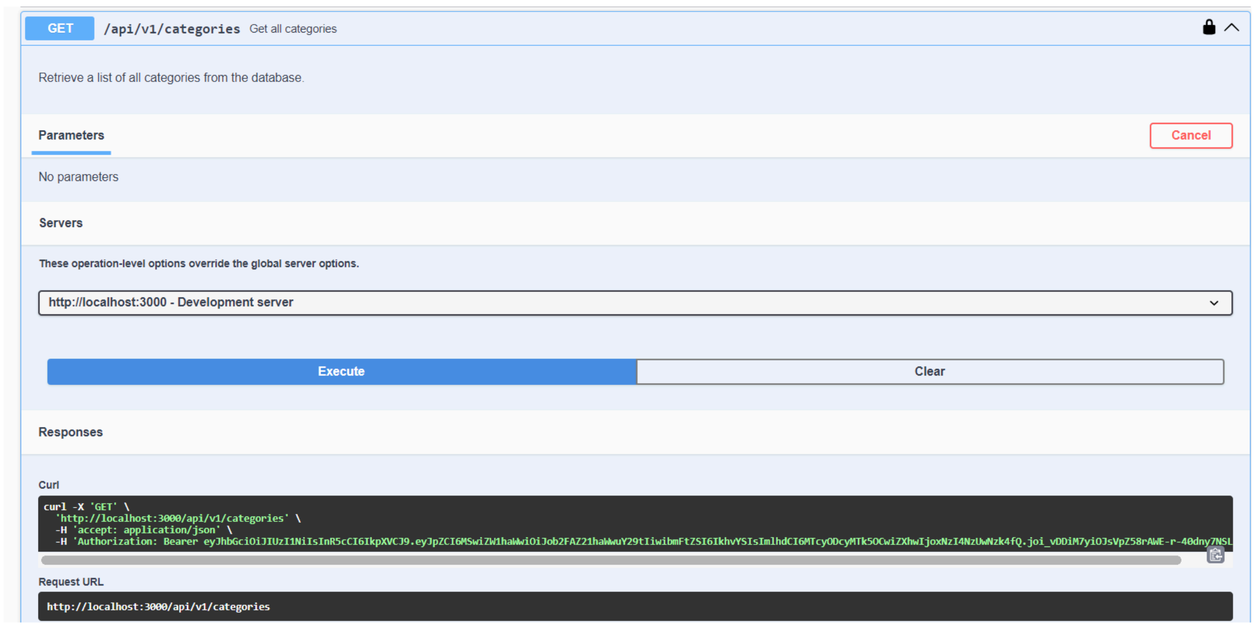
```
{
  "status": "fail",
  "message": "Expense not found"
}
```

```
access-control-allow-origin: *
connection: keep-alive
content-length: 47
content-type: application/json; charset=utf-8
date: Sat,12 Oct 2024 11:33:24 GMT
```

- **Implementation details:**
    + Endpoint : DELETE expenses/{id}
    + Data structure sent:
      Header : JWT_TOKEN
    + Data structure received:
      If found expense:
      ```
      {
        "status": "success",
        "data": {
          "message": "Expense deleted"
        }
      }
      ```
      If not:
      ```
      {
        "status": "fail",
        "message": "Expense not found"
      }
      ```
    + This feature read and delete in database
    + This feature need to be implemented in the UI of user

7. **GetAllCategory**

- **Description:** this feature is used for authenticated users to get their category
- **Screenshots:**



GET `/api/v1/categories` Get all categories

Retrieve a list of all categories from the database.

Parameters

No parameters

Servers

These operation-level options override the global server options.

http://localhost:3000 - Development server

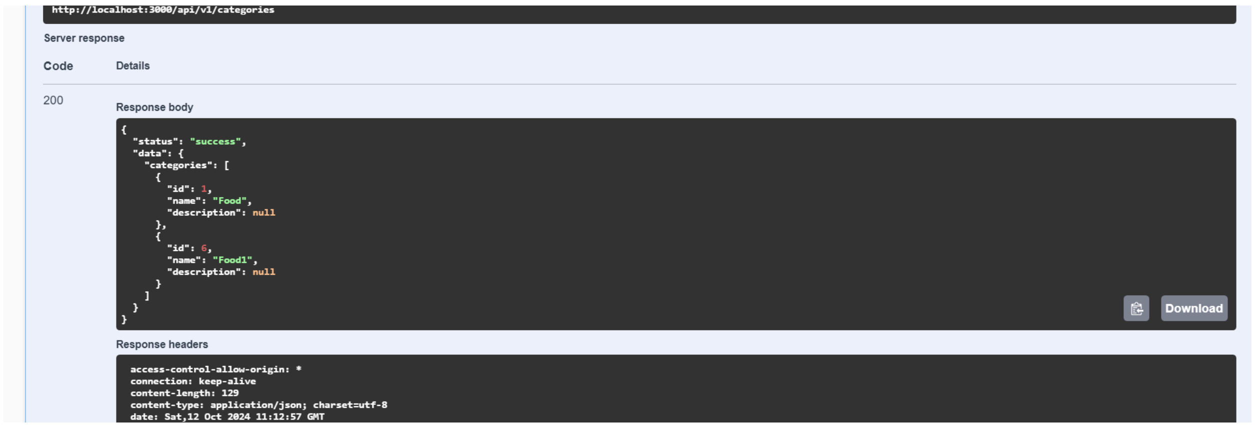
Responses

Curl

```
curl -X 'GET' \
  'http://localhost:3000/api/v1/categories' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiZW1haWwiOiJob2FAZ21haWwuY29tIiwibmFtZSI6IkhvYSIsImlhdCI6MTcyODcyMTk5OCwiZXhwIjoxNzI4NzUwNzk4fQ.joi_vDDiM7yiOJsVpZ58rAWE-r-40dny7NSL
```

Request URL

```
http://localhost:3000/api/v1/categories
```



```
http://localhost:3000/api/v1/categories
```

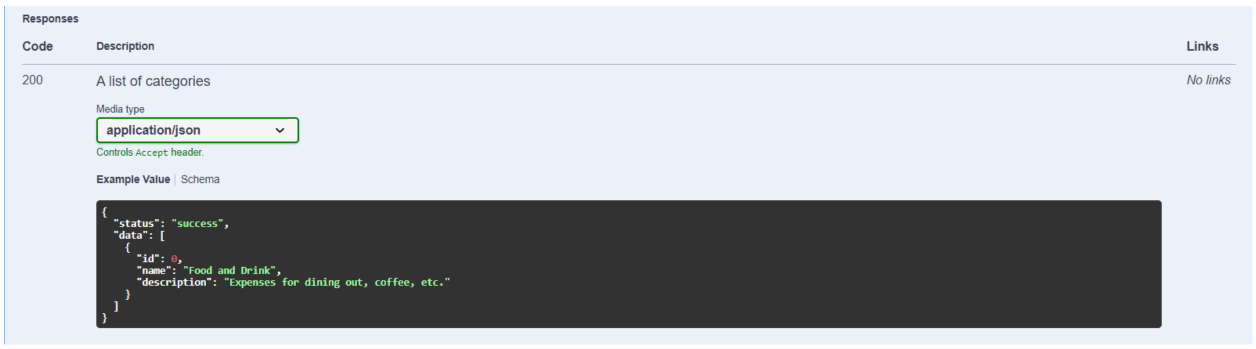
Server response

Code	Details
200	Response body

```
{
  "status": "success",
  "data": {
    "categories": [
      {
        "id": 1,
        "name": "Food",
        "description": null
      },
      {
        "id": 6,
        "name": "Food1",
        "description": null
      }
    ]
  }
}
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 129
content-type: application/json; charset=utf-8
date: Sat,12 Oct 2024 11:12:57 GMT
```



Responses

Code	Description	Links
200	A list of categories	*No links*

Media type: application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "status": "success",
  "data": [
    {
      "id": 0,
      "name": "Food and Drink",
      "description": "Expenses for dining out, coffee, etc."
    }
  ]
}
```

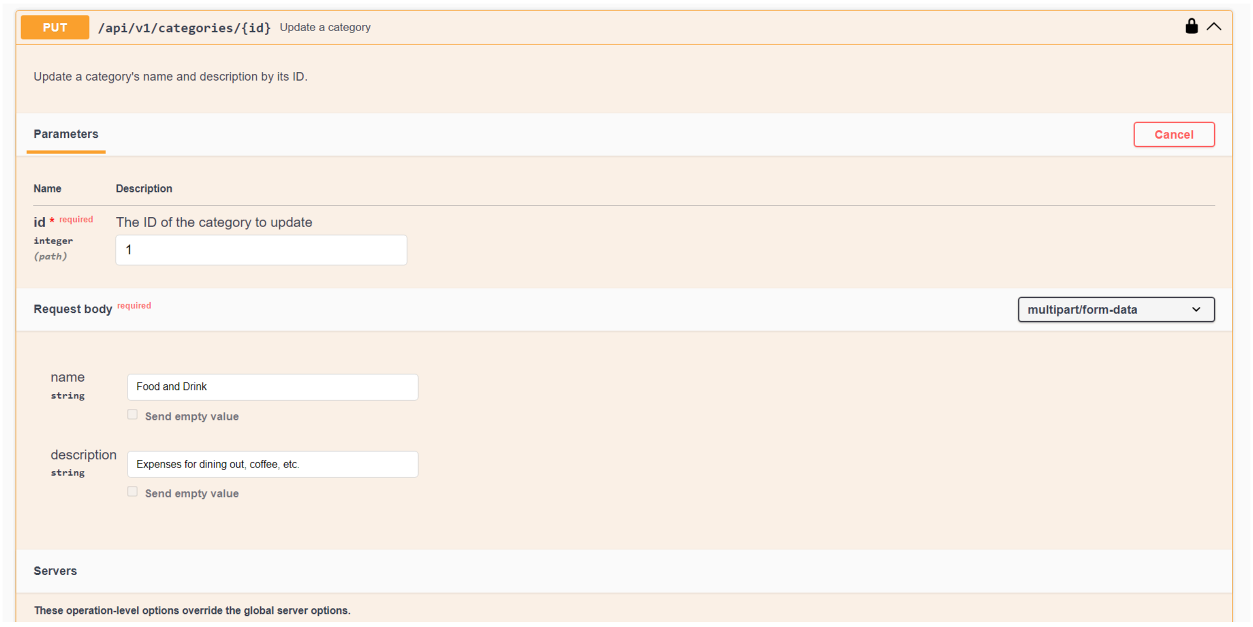
- **Implementation details:**
  - Endpoint : GET categories
  - Data structure sent:
    Header : JWT_TOKEN
  - Data structure received:
    {
     "status": "success",

```
"data": {
  "categories": [
    {
      "id": 1,
      "name": "Food",
      "description": null
    },
    {
      "id": 6,
      "name": "Food1",
      "description": null
    }
  ]
}
}
```

+ This feature read the database
+ This feature need to be implemented in the UI of user

8. Update the category

- **Description:** this feature is used for authenticated users to update a category
- **Screenshots:**



PUT /api/v1/categories/{id} Update a category

Update a category's name and description by its ID.

Parameters

Cancel

Name	Description
id * required integer (path)	The ID of the category to update: 1

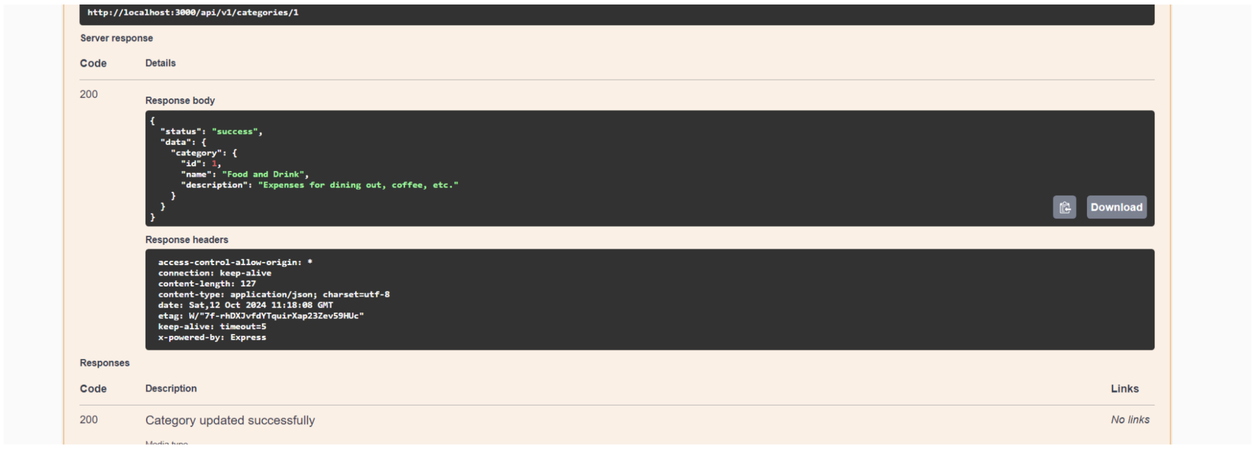
Request body required — multipart/form-data

name string: Food and Drink
Send empty value

description string: Expenses for dining out, coffee, etc.
Send empty value

Servers

These operation-level options override the global server options.



```
http://localhost:3000/api/v1/categories/1
```

Server response

Code | Details

200

Response body

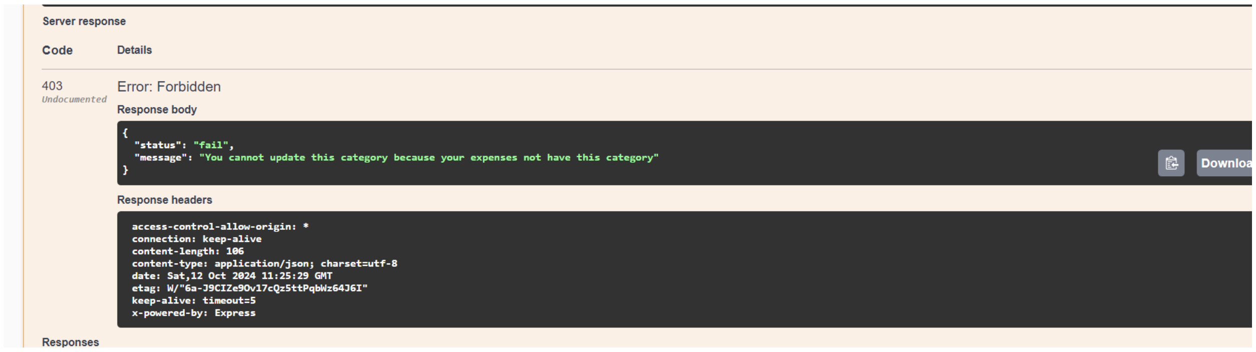
```
{
  "status": "success",
  "data": {
    "category": {
      "id": 1,
      "name": "Food and Drink",
      "description": "Expenses for dining out, coffee, etc."
    }
  }
}
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 127
content-type: application/json; charset=utf-8
date: Sat,12 Oct 2024 11:18:08 GMT
etag: W/"7f-rhDXJvfdYTquirXap23Zev59HUc"
keep-alive: timeout=5
x-powered-by: Express
```

Responses

Code	Description	Links
200	Category updated successfully	No links



Server response

Code | Details

403 *Undocumented* Error: Forbidden

Response body

```
{
  "status": "fail",
  "message": "You cannot update this category because your expenses not have this category"
}
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 106
content-type: application/json; charset=utf-8
date: Sat,12 Oct 2024 11:25:29 GMT
etag: W/"6a-J9CIZe9Ov17cQz5ttPqbWz64J6I"
keep-alive: timeout=5
x-powered-by: Express
```

Responses

- **Implementation details:**
  + Endpoint : PUT categories/{id}
  + Data structure sent:
    Header : JWT_TOKEN
    Body:
      - name: string
      - description : string
  + Data structure received:
    If user have that category in their expense:

```
{
  "status": "success",
  "data": {
    "category": {
      "id": 1,
      "name": "Food and Drink",
      "description": "Expenses for dining out, coffee, etc."
    }
  }
}
```

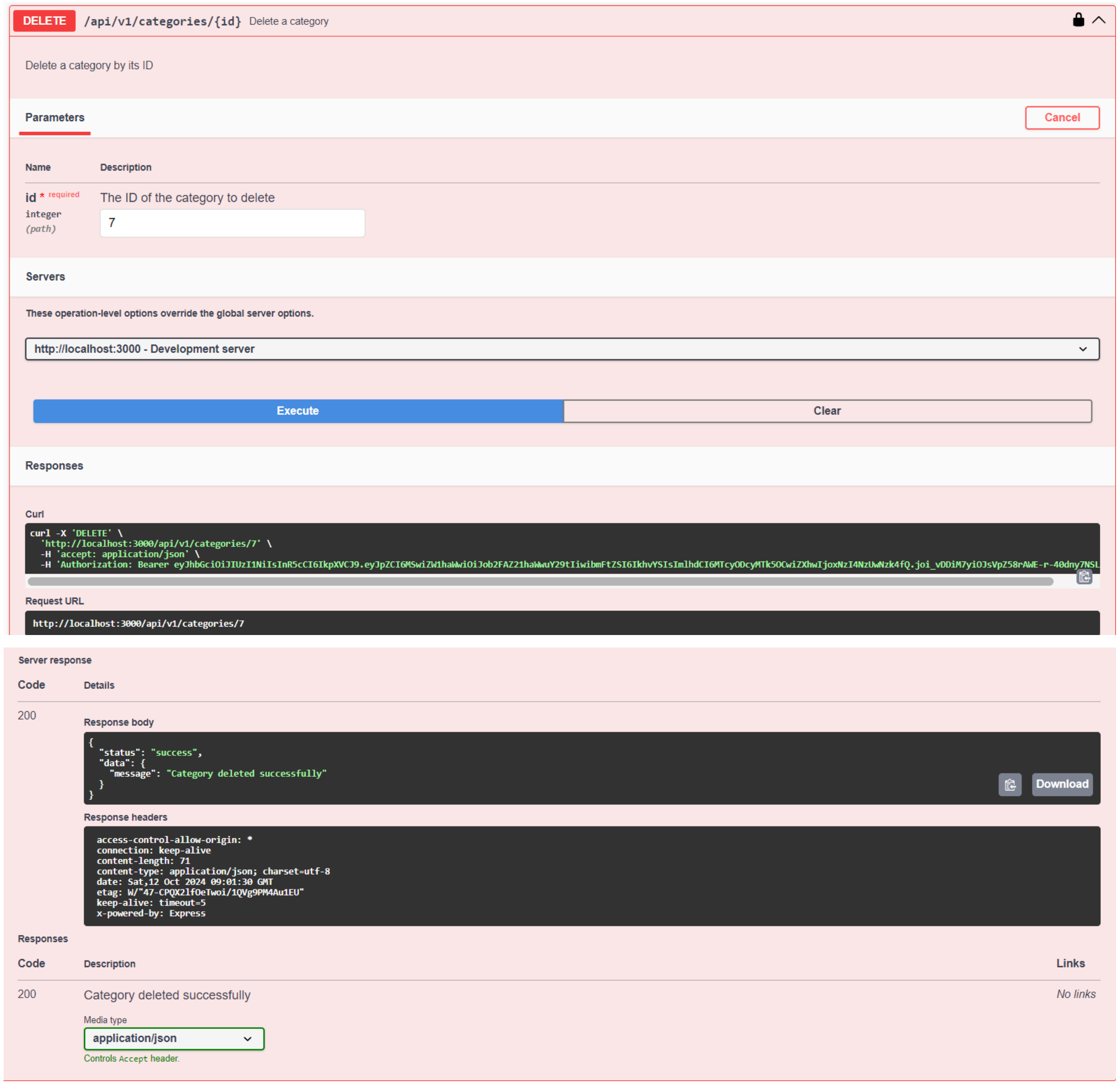
If not:

```
{
  "status": "fail",
  "message": "You cannot update this category because your expenses not have this category"
}
```

+ This feature read and write in database
+ This feature need to be implemented in the UI of user

9. Delete the category

- **Description:** this feature is used for authenticated users to delete a category
- **Screenshots:**



DELETE
/api/v1/categories/{id} Delete a category
Delete a category by its ID
Parameters
Cancel
Name
Description
id * required
integer
(path)
The ID of the category to delete
7
Servers
These operation-level options override the global server options.
http://localhost:3000 - Development server
Execute
Clear
Responses
Curl

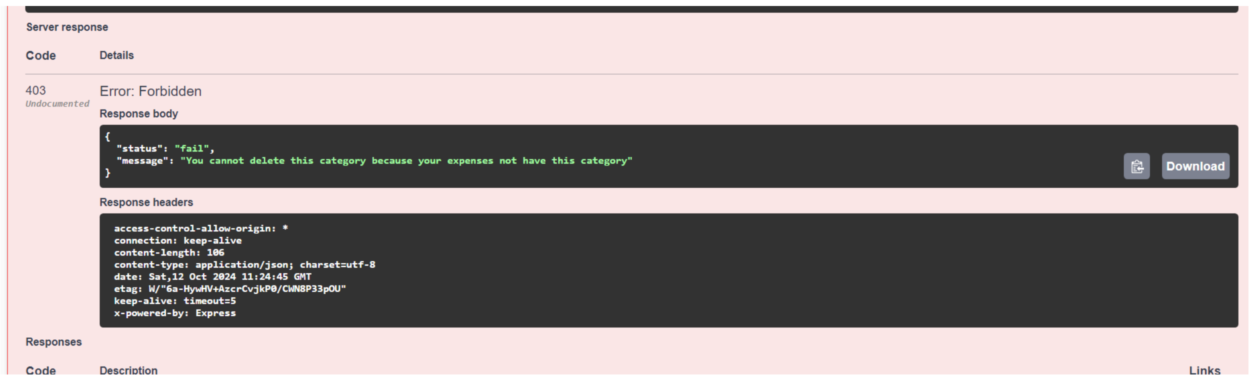
```
curl -X 'DELETE' \
  'http://localhost:3000/api/v1/categories/7' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiZW1haWwiOiJob2FAZ21haWwuY29tIiwibmFtZSI6IkhvYSIsImlhdCI6MTcyODcyMTk5OCwiZXhwIjoxNzI4NzUwNzk4fQ.joi_vDDiM7yiOJsVpZ58rAWE-r-40dny7NSL
```

```
http://localhost:3000/api/v1/categories/7
```

Server response
Code
Details
200
Response body

```
{
  "status": "success",
  "data": {
    "message": "Category deleted successfully"
  }
}
```

```
access-control-allow-origin: *
connection: keep-alive
content-length: 71
content-type: application/json; charset=utf-8
date: Sat,12 Oct 2024 09:01:30 GMT
etag: W/"47-CPQX2lfOeTwoi/1QVg9PM4Au1EU"
keep-alive: timeout=5
x-powered-by: Express
```


Server response

Code Details

403 Undocumented Error: Forbidden

Response body

```
{
  "status": "fail",
  "message": "You cannot delete this category because your expenses not have this category"
}
```

Response headers

```
 access-control-allow-origin: *
 connection: keep-alive
 content-length: 106
 content-type: application/json; charset=utf-8
 date: Sat,12 Oct 2024 11:24:45 GMT
 etag: W/"6a-HywHV+AzcrCvjkP0/CWN8P33pOU"
 keep-alive: timeout=5
 x-powered-by: Express
```

Responses

Code Description Links

- **Implementation details:**
  + Endpoint : DELETE categories/{id}
  + Data structure sent:

    Header : JWT_TOKEN
  + Data structure received:

    If user have that category :

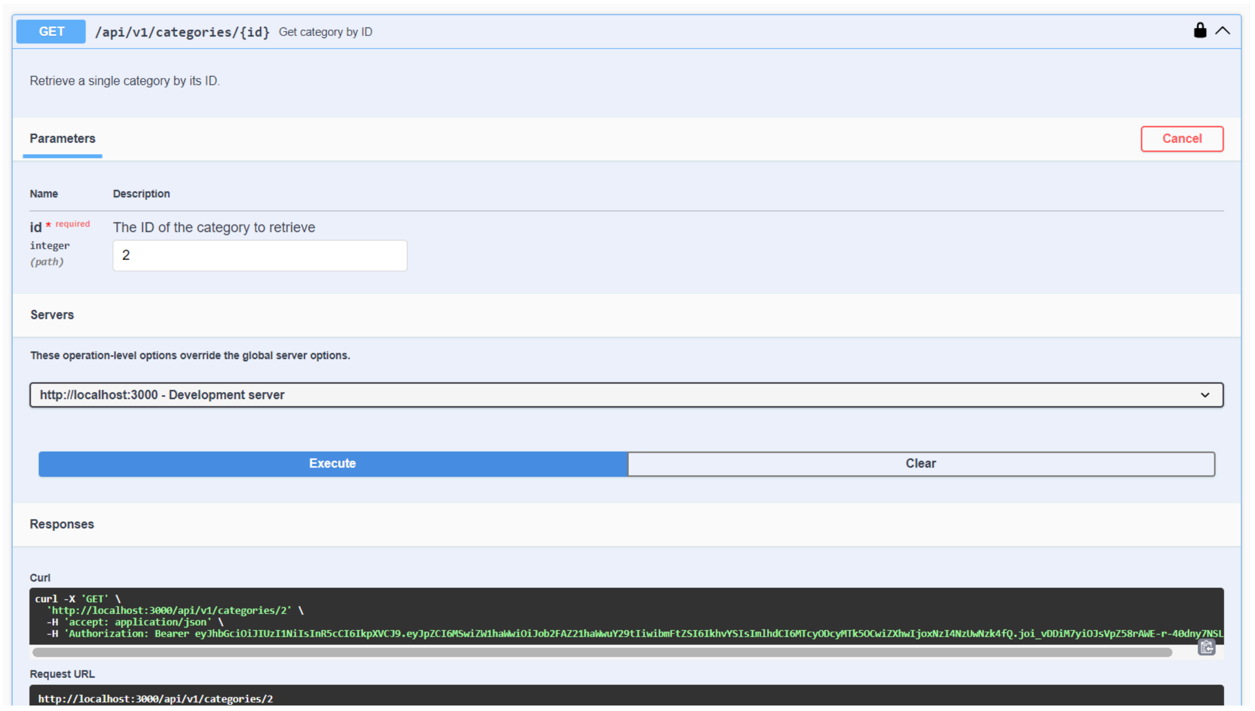
    {
      "status": "success",
      "data": {
        "message": "Category deleted successfully"
      }
    }

    If not:

    {
      "status": "fail",
      "message": "You cannot delete this category because your expenses not have this category"
    }
  + This feature read and delete in database
  + This feature need to be implemented in the UI of user

10. Get category by id

- **Description:** this feature is used for authenticated users to get category by id
- **Screenshots:**



GET `/api/v1/categories/{id}` Get category by ID

Retrieve a single category by its ID.

Name	Description
id * required integer *(path)*	The ID of the category to retrieve 2

Servers

These operation-level options override the global server options.

http://localhost:3000 - Development server

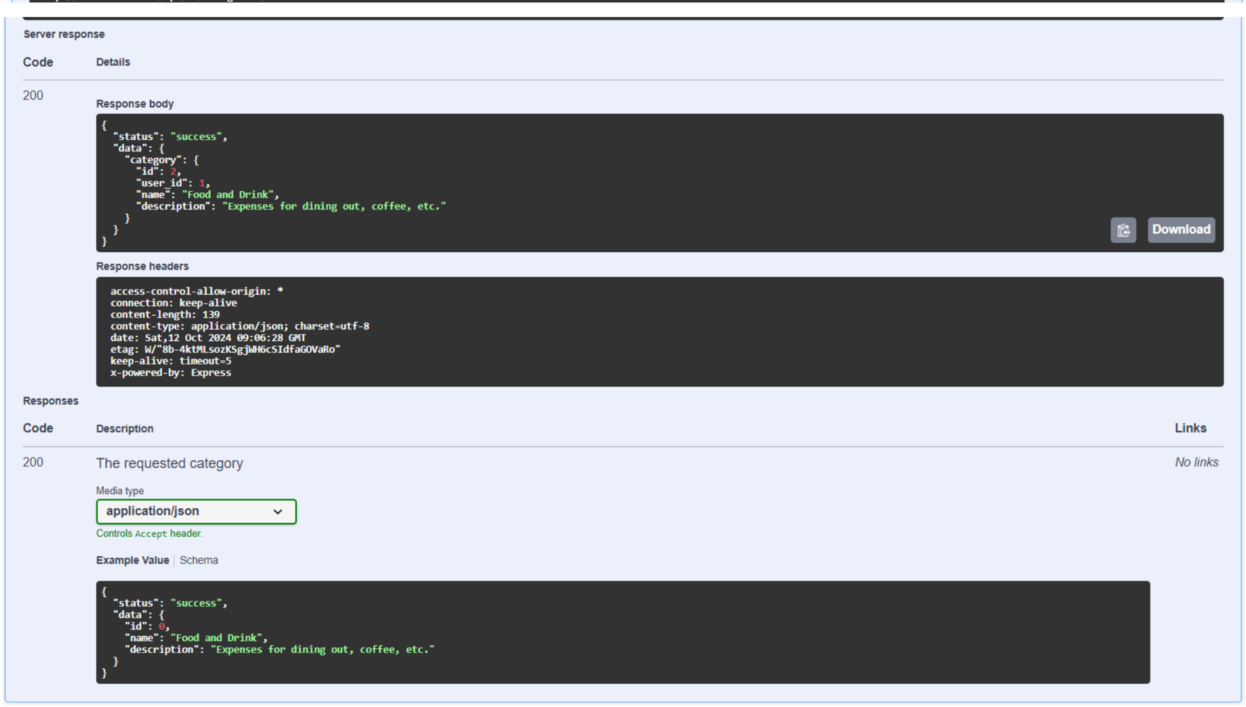
Responses

Curl

```
curl -X 'GET' \
  'http://localhost:3000/api/v1/categories/2' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiZW1haWwiOiJob2FAZ21haWwuY29tIiwibmFtZSI6IkhvYSIsImlhdCI6MTcyODcyMTk5OCwiZXhwIjoxNzI4NzUwNzk4fQ.joi_vDDiM7yiOJsVpZ58rAWE-r-40dny7NSL
```

Request URL

```
http://localhost:3000/api/v1/categories/2
```



Server response

Code	Details
200	Response body

```
{
  "status": "success",
  "data": {
    "category": {
      "id": 2,
      "user_id": 1,
      "name": "Food and Drink",
      "description": "Expenses for dining out, coffee, etc."
    }
  }
}
```

Response headers

```
 access-control-allow-origin: *
 connection: keep-alive
 content-length: 139
 content-type: application/json; charset=utf-8
 date: Sat,12 Oct 2024 09:06:28 GMT
 etag: W/"8b-4ktMLsozKSgjWH6cSIdfaGOVaRo"
 keep-alive: timeout=5
 x-powered-by: Express
```

Responses

Code	Description	Links
200	The requested category	*No links*

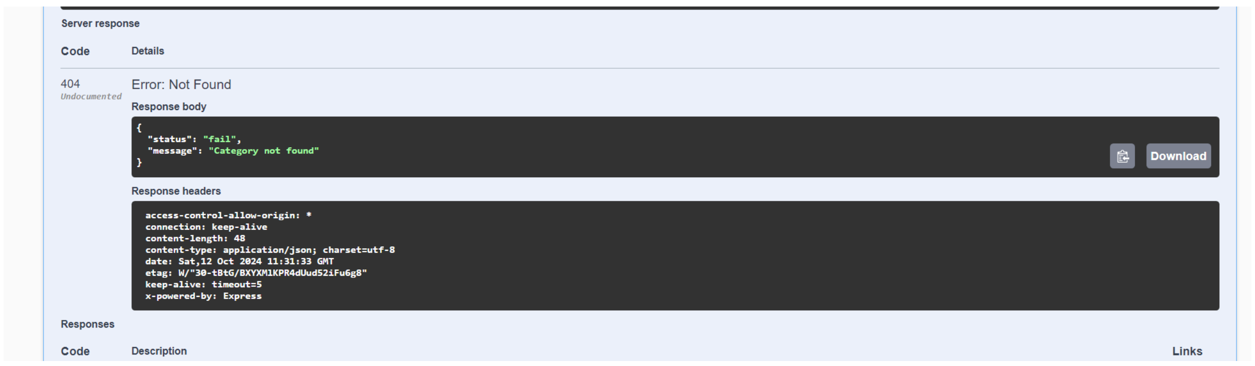
Media type

application/json

Controls `Accept` header.

Example Value | Schema

```
{
  "status": "success",
  "data": {
    "id": 0,
    "name": "Food and Drink",
    "description": "Expenses for dining out, coffee, etc."
  }
}
```


Server response

Code Details

404 Undocumented Error: Not Found

Response body

```
{
  "status": "fail",
  "message": "Category not found"
}
```

Response headers

```
access-control-allow-origin: *
connection: keep-alive
content-length: 48
content-type: application/json; charset=utf-8
date: Sat,12 Oct 2024 11:31:33 GMT
etag: W/"30-tBtG/BXYXMlKPR4dUud52iFu6g8"
keep-alive: timeout=5
x-powered-by: Express
```

Responses

Code Description Links

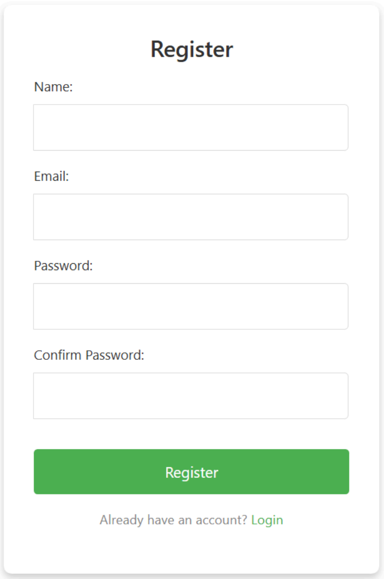
- **Implementation details:**
  - Endpoint: GET categories/{id}
  - Data structure sent:
    Header : JWT_TOKEN
  - Data structure received:
    If found:
    ```
    {
      "status": "success",
      "data": {
        "category": {
          "id": 3,
          "name": "Food and Drink",
          "description": "Expenses for dining out, coffee, etc."
        }
      }
    }
    ```
    If not:
    ```
    {
      "status": "fail",
      "message": "Category not found"
    }
    ```
  - This feature read in database

+ This feature need to be implemented in the UI of user

Front-end:

1. **Register**

- **Description: A feature for users to register an account.**
- **Screenshots:**



- **Implementation details:**

  1. Any libraries other than the libraries introduced in the class (none)

  2. Which server-side APIs does this feature use?

  Endpoint: /api/v1/users/register

  Method: POST

  Description:

  This API receives user details (name, email, password) from the client.

  Checks if the email already exists.

  Hashes the password using bcrypt.

  Saves the new user's information into the database.

  Returns a response indicating whether registration was successful or failed.

3. Does this feature read/store data? In which table?

Data read/write tables:

Table: users

Read columns: email (to check for duplicates).

Write columns: name, email, password (hashed).

4. Which client-side states are needed to implement this feature?

Client-side states:

Form state:

name, email, password, confirmPassword: stores input data from the user.

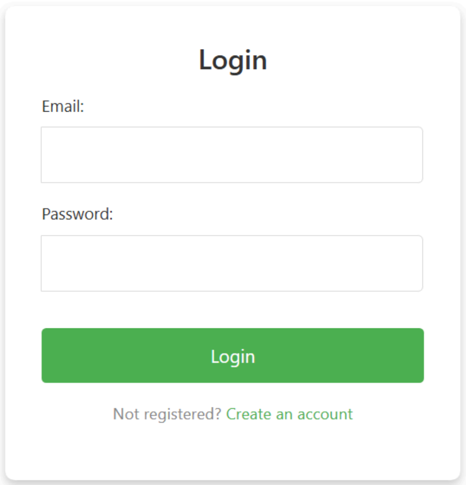
UI state:

message: displays feedback messages (e.g., "Wrong password" or "Registration successful").

isSubmitting (if added): represents the state when the registration request is being processed.

2. Login

- **Description: A feature for users to login.**
- **Screenshots:**



- **Implementation details:**

  1. **Any libraries other than the libraries introduced in the class (none)**
  2. **Which server-side APIs does this feature use?**

     Endpoint: /api/v1/users/login

       Method: POST

       Description:

          Accepts the user’s email and password.

          Checks the database for a user with the provided email.

          Verifies the password using bcrypt.

          If the login is successful:

            Generates a JSON Web Token (JWT) and sends it to the client.

          If the login fails:

            Sends an error response with an appropriate message.

3. Does this feature read/store data? In which table?

Data read/write tables:

Table: users

Read columns: email (to find the user), password (for verification).

No data is stored or modified in this feature; it only validates user credentials and returns a token if successful.

4. Which client-side states are needed to implement this feature?

Client-side states:

Form state:

email and password: Stores the user’s input for authentication.

UI state:

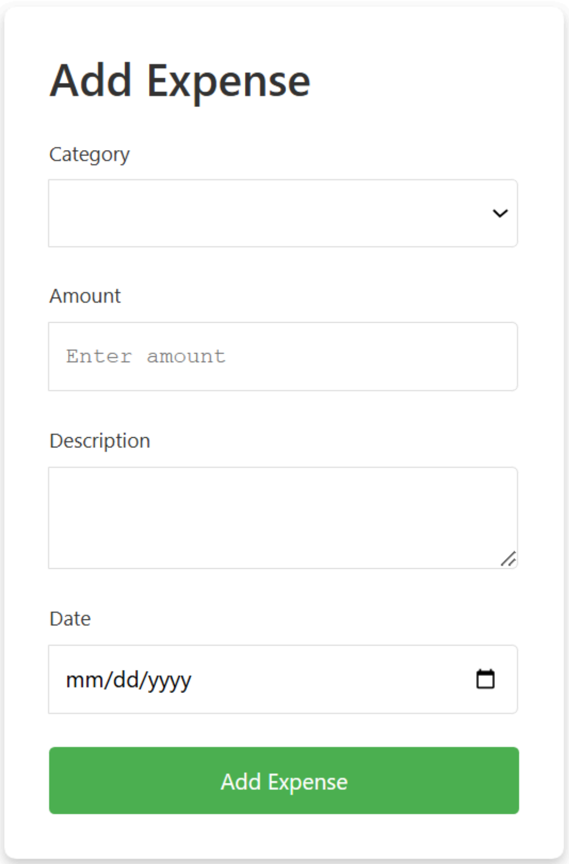
message: Displays feedback messages (e.g., "Login successful" or "Login failed. Please try again.").

Session state:

The authentication token (authToken) is stored in sessionStorage upon successful login.

3. Add expense

- **Description:**
- **Screenshots:**



- **Implementation details:**

  1. Any libraries other than the libraries introduced in the class?(None)

  2. Which server-side APIs does this feature use?

  The AddExpense feature interacts with the following server-side API:

  API: POST /api/v1/expenses
  Sends a request to add a new expense with data passed in the request body.

  3. Does this feature read/store data? In which table?

  Read:

  Categories Table: Reads categories via the API (GET /api/v1/categories) to display them in the form.

Store:

Expenses Table: The new expense data is stored in the expenses table.

Data sent includes:

category_id or category_name (if adding a new category).

amount (the expense amount).

description (details about the expense).

date (the date of the expense).

4. Which client-side states are needed to implement this feature?

The required client-side states include:

categories (list of categories): Stores the list of categories to display in the dropdown.

selectedCategory (selected category): Tracks the category selected by the user in the form.

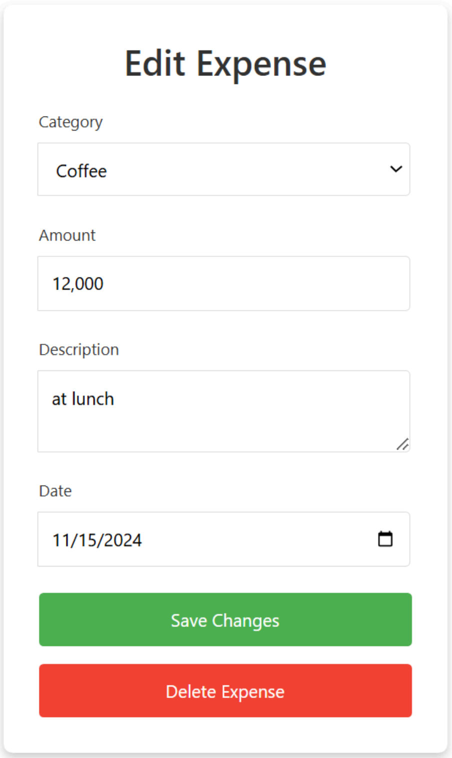
isAddingNewCategory: Determines whether a new category is being added.

newCategoryName: Stores the name of the new category when entered by the user.

expense: Stores the expense data, including fields: category_id, amount, description, and date.

4. Edit expense

- **Description:**
- **Screenshots:**



Edit Expense

Category

Coffee

Amount

12,000

Description

at lunch

Date

11/15/2024

- **Implementation details:**

1. Any libraries other than the libraries introduced in the class?(None)

2. Which server-side APIs does this feature use?

The EditExpense feature interacts with the following server-side APIs:

GET /api/v1/categories: Retrieves the list of available categories to populate the category dropdown.

GET /api/v1/expenses: Fetches all expenses to identify the specific expense being edited.

PUT /api/v1/expenses/:id: Updates the expense data for the given expense ID with the modified details.

3. Does this feature read/store data? In which table?

Read:

Categories Table: Reads the list of categories via the API (GET /api/v1/categories) for display in the dropdown.

Expenses Table: Reads the expense being edited via the API (GET /api/v1/expenses).

Store:

Expenses Table:

Updates the expense data (category, amount, description, and date) via the API (PUT /api/v1/expenses/:id).

4. Which client-side states are needed to implement this feature?

The following client-side states are used to implement the EditExpense feature:

expense (object): Stores the current expense details (e.g., category, amount, description, date) for editing.

categories (array): Holds the list of available categories fetched from the server to populate the dropdown.

selectedCategory (string): Tracks the selected category in the dropdown menu.

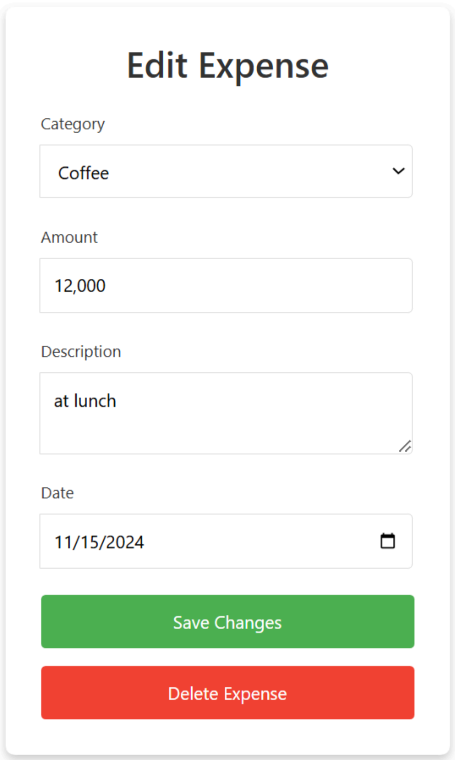
isAddingNewCategory (boolean): Indicates whether the user is adding a new category.

newCategoryName (string): Holds the name of a new category entered by the user.

successMessage (string): Temporarily displays a success message when the expense is successfully updated.

+

5. **Delete expense**
- **Description:**
- **Screenshots:**



Edit Expense

Category: Coffee

Amount: 12,000

Description: at lunch

Date: 11/15/2024

- **Implementation details:**

  1. **Any libraries other than the libraries introduced in the class?(None)**

  2. **Which server-side APIs does this feature use?**

     DELETE /api/v1/expenses/:id:Deletes the expense identified by the given ID from the database.

  3. **Does this feature read/store data? In which table?**

     Store:

     Expenses Table: Deletes the specific expense entry via the API (DELETE /api/v1/expenses/:id).

  4. **Which client-side states are needed to implement this feature?**

  The following client-side states are used for the delete functionality:

successMessage (string): Temporarily displays a success message when the expense is successfully deleted.