Load Dummy Data

Add dummy data to reduce time wastage

GET New Request

http://127.0.0.1:5000/loaddummy

This endpoint makes an HTTP GET request to retrieve dummy data. The request does not include a request body.

Response

- Status: 200
- { "message": "Data has been Added" }

Login and Logout APi

Logina and Logout api

POST Login

http://127.0.0.1:5000/user_login

This endpoint allows users to log in with their credentials. The request should be sent as an HTTP POST to the specified URL. The request body should be in raw format and include the user's email address and password. Upon successful execution, the server will respond with a status code of 200 along with a JSON object containing his detials like auth_token.

Body raw (json)

json

```
{
    "email_address": "Postman@email.com" ,
    "password": "Postman"
}
```

POST Logout

http://127.0.0.1:5000/user_logout

User Logout

This endpoint is used to log out a user from the system.

Request Body

- No request body is required for this endpoint.

Response

- Status: 200
- **message**: A message indicating the result of the logout operation.

POST Register

http://127.0.0.1:5000/user

This endpoint allows you to create a new user by sending a POST request to the specified URL. The request should include a JSON payload in the raw request body with the following parameters:

- username (string): The username of the new user.
- email_address (string): The email address of the new user.
- password (string): The password for the new user.
- contact_number (string): The contact number of the new user.
- home_address (string): The home address of the new user.

Response

Upon successful execution, the endpoint will return a status code of 200 along with a JSON response containing a message indicating the success of the operation.

Body raw (json)

json

```
{
    "username": "Postman",
    "email_address": "Postman@email.com" ,
    "password": "Postman",
    "contact_number": "9865142",
    "home_address": "112/34, Postman Street"
}
```

Orders

GET get order history

http://127.0.0.1:5000/orderhistory/1

This endpoint retrieves the order history for a specific user by their ID.

The request does not require a request body.

Response

- Status: 200
- Body:

```
Plain Text

[
    {
        "id": 0,
        "user_id": 0,
        "product_id": 0,
        "item_name": "",
        "item_quantity": 0,
        "item_total": 0,
        "purchase_date": ""
    }
]
```

The response returns an array of order history objects, each containing the ID, user ID, product ID, item name, item quantity, item total, and purchase date.

PRODUCT API

Managers can make a request to CRUD products to the admin through these apis

GET Get all the products

http://127.0.0.1:5000/search

This endpoint makes an HTTP GET request to retrieve search results. The request does not include a request body.

The response returns a status code of 200, along with an array of objects containing search results. Each object includes a category ID, category name, and an array of linked products. Each linked product object contains a product ID, product name, product price, image URL, stock quantity, manufacture date, and expiry date.

AUTHORIZATION Bearer Token

Token	<token>

POST add a new product

http://127.0.0.1:5000/products

This API endpoint allows you to add a new product to the database. The request should be sent as an HTTP POST to the specified URL with the following payload in the raw request body type:

json

```
{
    "product_name": "kitkat",
    "product_price": 3,
    "stock_quantity": 6,
    "category_id": 3,
    "imageUrl" : "static/KitKat.png"
}
```

You can also include the optional fields "manufacture_date" and "expiry_date" in the payload in the format of "YYYY-MM-DD HH:MM:SS". However, when sending the manufacturing date from the Vue.js, ensure that it is sent in the DateTime format itself, as it is only accepted as a string here.

The response to the request will have a status code of 200 and will include a message indicating the success of the operation.

Body raw (json)

Body raw (json)

```
json

{
    "product_name": "kitkat",
    "product_price": 3,
    "stock_quantity": 6,
    "category_id": 3,
    "imageUrl" : "static/KitKat.png"}


    // "manufacture_date" : "2023-12-11 00:00:00",
    // "expiry_date": "2024-12-12 00:00:00"


//when sending the manufacturing data from the vue js then send it in the Date time Format its
```

PUT update a product

http://127.0.0.1:5000/products

This endpoint allows you to update product information using an HTTP PUT request to the specified URL. The request should include a JSON payload with the product details such as product_id, product_name, and stock_quantity.

Request Body

- `product_id` (number): The unique identifier of the product.
- `product_name` (string): The name of the product.
- `stock_quantity` (number): The available quantity of the product.

The response will include a status code of 200, along with a JSON object containing a message confirming the successful update of the product information.

Body raw (json)

```
json

{
    "product_id": 2,
    "product_name": "h2O",
    "stock_quantity": 1000
    // "product_price": 552,
    // "category_id": 1
}
```

DELETE delete a product

DELETE delete a product

http://127.0.0.1:5000/products/2

This endpoint sends an HTTP DELETE request to delete the product with ID 2.

The request does not contain a request body.

Response

- Status: 200
- Body:

```json
{
    "message": ""
}
```

Search

One can Search by Items , price, category
I also user it to doplay the prodicuts in the home page.

GET show all the categories and their products

http://127.0.0.1:5000/search

This endpoint makes an HTTP GET request to retrieve search results. The request does not include a request body.

Response

- Status: 200
- The response body contains an array of objects, each representing a category. Each category object includes:
  - category_id (number): The unique identifier for the category.
  - category_name (string): The name of the category.
  - linked_products (array): An array of products linked to the category, where each product object includes:
    - product_id (number): The unique identifier for the product.
    - product_name (string): The name of the product.
    - product_price (number): The price of the product.
    - imageUrl (string): The URL of the product image.
    - stock_quantity (number): The available stock quantity of the product.
    - manufacture_date (string): The date of manufacture of the product.
    - expiry_date (string): The expiry date of the product.

GET Search for a particular item

http://127.0.0.1:5000/search/fruits?searchneighbourbutton=Category

Search Fruits by Category

This endpoint makes an HTTP GET request to search for fruits based on the category specified.

Request

Endpoint

`GET http://127.0.0.1:5000/search/fruits`

Query Parameters

- `searchneighbourbutton` : Category to search by

Response

Status: 200 OK

The response will be a JSON array containing objects with the following properties:

- `category_id` : The ID of the category
- `category_name` : The name of the category
- `linked_products` : An array of objects containing details of products linked to the category, including:
  - `product_id` : The ID of the product
  - `product_name` : The name of the product
  - `product_price` : The price of the product
  - `imageUrl` : The URL of the product image
  - `stock_quantity` : The quantity of the product in stock
  - `manufacture_date` : The date of manufacture
  - `expiry_date` : The expiry date of the product

PARAMS

searchneighbourbutton	Category

GET Search for item

http://127.0.0.1:5000/search/apple

This endpoint makes an HTTP GET request to retrieve information about the search term "apple". The response will include a list of categories and linked products related to the search term.

The response will have a status code of 200, indicating a successful request. The response body will contain an array of objects, each representing a category. Each category object will include a category ID, category name, and an array of linked products. Each linked product object will contain a product ID, product name, product price, image URL, stock quantity, manufacture date, and expiry date.

USER API

This Collection allows the user to register, change his password , and to delete his account .
He can also apply for becoming a store manager !

GET see all users

http://127.0.0.1:5000/user

This endpoint allows you to retrieve user information.

Request

The request does not require any parameters.

Response

Upon a successful request, the server will respond with a status code of 200 and a JSON array containing user information. Each user object includes the following fields:

- `username`: The username of the user.
- `email_address`: The email address of the user.
- `active`: A boolean indicating whether the user is active.
- `roles`: An array of roles associated with the user, where each role object includes the following field:
  - `name`: The name of the role.

POST add a new user

http://127.0.0.1:5000/user

This endpoint allows you to create a new user by sending a POST request to the specified URL. The request should include a JSON payload in the raw request body with the following parameters:

- username (string): The username of the new user.
- email_address (string): The email address of the new user.
- password (string): The password for the new user.
- contact_number (string): The contact number of the new user.
- home_address (string): The home address of the new user.

Response

Upon successful execution, the endpoint will return a status code of 200 along with a JSON response containing a message indicating the success of the operation.

Body raw (json)

```json
{
"username": "harry",
"email_address": "Harry@gmail.com",
"password":"harry",
"contact_number": "12323123",
"home_address": "Harryone-Street "
}
```

PUT change user password

http://127.0.0.1:5000/user

This endpoint allows the user to update their account information, including the username, email address, and password. The HTTP PUT request should be sent to http://127.0.0.1:5000/user with a JSON payload containing the user's email address, old password, and new password.

The request should have the following structure:

```json
{
    "email_address": "harry@gmail.com",
    "new_password": "harrynew",
    "old_password": "harry"
}
```

Upon successful execution, the endpoint returns a status code of 401 with a JSON response containing meta information and any potential errors.

Example response:

```json
{
    "meta": {
        "code": 0
    },
    "response": {
        "errors": [""]
    }
}
```

Body raw (json)

json

```
// //Unknwon person
// {
//     "username": "UnknwonPerson",
//     "email_address": "UnknownPerson@gmail.com",
//     "new_password": "Harry",
//     "old_password": "Harry"
// }
// // old_pass != old_pass_in_db
// {
//     "username": "Harry",
//     "email_address": "Harry@gmail.com",
//     "new_password": "NewHarry",
//     "old_password": "Not the password in database"
// }
// // old_pass == new_pass
// {
//     "username": "Harry",
//     "email_address": "Harry@gmail.com",
//     "new_password": "Harry",
//     "old_password": "Harry"
// }
// Fianlly changing the password
// {
//     "jibbersih": "jibberish",
//     "email_address": "Harryone@gmail.com",
//     "new_password": "1234",
//     "old_password": "1234"
//}
{

    "email_address": "harry@gmail.com",
    "new_password": "harrynew",
    "old_password": "harry"
}
```

DELETE delete a user

http://127.0.0.1:5000/user

Delete User

This endpoint is used to delete a user from the system.

Request

- Method: DELETE
- URL: `http://127.0.0.1:5000/user`
- Payload (raw request body type):

```json
{
    "email_address": "Harry@gmail.com",
    "password": "harry"
}
```

Response

- Status: 401
- Body:

```json
{
    "meta": {
        "code": 0
    },
    "response": {
        "errors": [""]
    }
}
```

Body raw (json)

```json
//// Wrong Username
// {
//     "username": "Harry",
//     "email_address": "UnknownPerson@gmail.com",
//     "password": "NEWHarry"
// }

////correct usecase
{
"email_address": "Harry@gmail.com",
"password":"harry",
//below things are just extra, dont need them
"username": "hasdarry",
"contact_number": "12323123",
"home_address": "Harryone"
}
```

POST request to become a Stroe manager

http://127.0.0.1:5000/BecomeStoreManger

Sends a request to admin to become a storemanager

HEADERS

Authentication-Token	Authentication-Token

AdminResource

Admin can finalize the requests and then approve them or reject them.
Finally CRUD on products and categories through this only!

GET show all requests

http://127.0.0.1:5000/admin

This endpoint makes an HTTP GET request to retrieve admin information.

Request

The request does not require any parameters or body.

Response

- Status: 200
- Body: null

POST New Request

http://127.0.0.1:5000/admin

The `POST /admin` endpoint is used to add documentation. The request should include a JSON payload with the `request_id` and `what_action` fields. The `request_id` should be a number, and the `what_action` should be a string indicating the action to be performed.

Example Request:

```json
{
    "request_id": 1,
    "what_action": "Add"
}
```

PARAMS

request_id	1

what_action	Add

Body raw (json)

json

```
// {
//     "request_id": 4,
//     "what_action": "Add"
// }
{
    "request_id": 1,
    "what_action": "Add"
}
```

CART CHECKOUTAPI

Add to order and pay as cash on delivery

POST next step, Pay!

http://127.0.0.1:5000/checkout

This API endpoint is used to initiate the checkout process. Upon making a POST request to this endpoint, the server will process the request and return a response.

The last call to this request returned a 404 status code with an empty description in the response body.

No specific request payload information is available for this request.

CART ITEM API

Add to cart and remove from cart

GET Get all cart items of current_user

http://127.0.0.1:5000/cart

This endpoint makes an HTTP GET request to retrieve the contents of the cart. The request does not require any parameters or payload.

It fetches the current uses cart y his id

POST add to cart!

http://127.0.0.1:5000/cart

Body raw (json)

```json
{
    "quantity": 1,
    "product_id": 3
}
```

DELETE remove item from cart

http://127.0.0.1:5000/cart

Body raw (json)

```json
{
```

```
    "product_id" : 4,


    "quantity": 1
}
```

CATEGORY API

Managers can make a request to CRUD categories to the admin through these apis

POST add a new category

http://127.0.0.1:5000/categories

This endpoint allows you to create a new category by making an HTTP POST request to the specified URL. The request should include a JSON payload in the raw request body with the "category_name" field.

Request Body

- category_name (string, required): The name of the category to be created.

Response

- Status: 200 OK
- message (string): A message indicating the result of the operation.

Body raw (json)

json

```
{
    "category_name": "Sea Food"
}
```

PUT update a category with json

http://127.0.0.1:5000/categories

This endpoint allows you to update the category name by sending an HTTP PUT request to the specified URL. The request should include a JSON payload with the new category name, old category name, and category ID. If the category ID is sent, the system will automatically retrieve the ID of the specified category.

Request Body

- category_name (string): The new name of the category.
- old_category_name (string): The current name of the category

- old_category_name (string): The current name of the category.
- category_id (integer): The ID of the category to be updated.

Response

Upon successful execution, the endpoint will return a status code of 200 with an empty message in the response body.

Note: Ensure that the category ID is provided to update the category properly.

Body raw (json)

```json
{
    "category_name": "Phal",
    "old_category_name": "Fruits",
    "category_id": 1
}
//if we are sending this category_id then we have to implement a function that automatically g
```

DELETE delete a category

http://127.0.0.1:5000/categories/1

This endpoint makes an HTTP DELETE request to delete the category with ID 1.

Request

The request does not require a request body as it is a simple DELETE request.

- Endpoint: `http://127.0.0.1:5000/categories/1`
- Method: `DELETE`

Response

The response will have a status code of 200 indicating a successful deletion. The response body will contain a message confirming the deletion.

- Status: 200
- Response Body:

```json
{
    "message": ""
}
```