Food Order App API Guideline

Introduction

The Food Order API is a web-based application programming interface that provides access to various resources related to restaurants, cuisines, categories, and user details. This guideline document aims to provide developers with an overview of the API's endpoints, request/response formats, authentication requirements, and usage guidelines.

Base URL

The base URL for accessing the Food Order API is - https://foodorder-api-elti.onrender.com/v1/

Endpoints

1. Fetch Restaurants

- URL: /restaurants
- Method: GET
- Description: Retrieve a list of restaurants.
- Response:
    - If restaurants data retrieved successfully then,
        - Status: 200 OK
        - Body: JSON array of restaurant objects.
    - If restaurants data not available then,
        - Status: 400
        - Body : String message

2. Fetch Restaurant Details

- URL: /restaurants/{restaurant_id}
- Method: GET
- Description: Retrieve detailed information about a specific restaurant.
- Parameters:
    - restaurant_id: The unique identifier of the restaurant.
- Response:
    - If details of the specific restaurant are retrieved then,
        - Status: 200 OK
        - Body: JSON object representing the restaurant details.
    - If details of the specific restaurant are not available then,

        - Status: 400
        - Body: String message

3. Fetch Cuisines

- URL: /cuisines
- Method: GET
- Description: Retrieve a list of available cuisines.
- Response:
    - If cuisines data retrieved successfully then,
        - Status: 200 OK
        - Body: JSON array of cuisine objects.
    - If cuisines data not available then,
        - Status: 400
    - Body : String message

4. Fetch Cuisine Details

- URL: /cuisines/{cuisine_id}
- Method: GET
- Description: Retrieve detailed information about a specific cuisine.
- Parameters:
    - cuisine_id: The unique identifier of the cuisine.
- Response:
    - If details of the specific cuisine are retrieved then,
        - Status: 200 OK
        - Body: JSON object representing the cuisine details.
    - If details of the specific cuisine are not available then,
        - Status: 400
        - Body: String message

5. Fetch Categories

- URL: /categories
- Method: GET
- Description: Retrieve a list of available categories.
- Response:
    - If categories data retrieved successfully then,
        - Status: 200 OK
        - Body: JSON array of category objects.
    - If categories data not available then,

        - Status: 400
    - Body : String message

6. Fetch Category Details

- URL: /categories/{category_id}
- Method: GET
- Description: Retrieve detailed information about a specific category.
- Parameters:
    - category_id: The unique identifier of the category.
- Response:
    - If details of the specific category are retrieved then,
        - Status: 200 OK
        - Body: JSON object representing the category details.
    - If details of the specific cuisine are not available then,
        - Status: 400
        - Body: String message

7. Fetch Users

- URL: /users
- Method: GET
- Description: Retrieve a list of available users.
- Response:
    - If user's data retrieved successfully then,
        - Status: 200 OK
        - Body: JSON array of user objects.
    - If user's data not available then,
        - Status: 400
    - Body : String message

8. Fetch User Details

- URL: /users/{user_id}
- Method: GET
- Description: Retrieve detailed information about a specific user.
- Parameters:
    - user_id: The unique identifier of the user.
- Response:
    - If details of the specific user are retrieved then,
        - Status: 200 OK

- Body: JSON object representing the user details.
- If details of the specific user are not available then,
  - Status: 400
  - Body: String message

9. Login

- URL: /login
- Method: POST
- Body:

  Example=>{"username":"john35@d", "password":"johnd"}
- Description: Retrieve detailed information about a specific user.
- Response:
  - If login is successful, it returns the user data in JSON format with a status code of 200. For example:

```
{
  "data": {
    "email": "john@gmail.com",
    "fullname": "John Doe",
    "id": 1,
    "isAdmin": true,
    "password": "johnd",
    "username": "john35@d"
  }
}
```

  - If login fails due to invalid credentials, returns a JSON response with an error message and a status code of 403.
  - If an exception occurs during the login process, returns a JSON response with an error message and a status code of 400.

Error Handling

The API follows standard HTTP status codes for indicating the success or failure of a request. In case of an error, the response body will contain an error message along with the appropriate status code.