Of course. Here is the API documentation in a copy-pasteable Markdown format.

~~~
# API Documentation: Food Listing App

### **Version 1.1**

This document provides documentation for the Food Listing App's backend API. The
API uses session-based cookie authentication for admin routes.

---

## **Base URL**

The base URL for all endpoints is:
`http://localhost:3000`

---

## **Authentication**

Admin-only endpoints are protected by a session cookie. To access them, the admin
must first log in via the `/api/admin/login` endpoint. Subsequent requests from
the client will automatically include the `connect.sid` cookie, granting access to
protected routes.

---

## **Admin Endpoints**

These endpoints require an active admin session.

### **Admin Login**

`POST /api/admin/login`

Authenticates an administrator and creates a session cookie.

**Request Body:** (`application/json`)
| Field | Type | Description |
| :--- | :--- | :--- |
| `username` | String | The admin's username. |
| `password` | String | The admin's password. |

**Example Request:**
```json
{
    "username": "admin",
    "password": "password"
}
~~~

Responses:

- **200 OK:** `Admin login successful`
- **401 Unauthorized:** `Invalid admin credentials`

Admin Logout

`POST /api/admin/logout`

Destroys the current admin session and clears the session cookie.

Responses:

- **200 OK:** `Admin logout successful`
- **500 Internal Server Error:** `Failed to log out.`

Check Admin Status

`GET /api/admin/status`

Checks if the current session belongs to a logged-in admin. This is useful for front-end routing.

Responses:

- **200 OK:** Returns a JSON object confirming admin status.

  ```
  {
      "isAdmin": true
  }
  ```

- **401 Unauthorized:**

  ```
  {
      "isAdmin": false
  }
  ```

Add Food Item

`POST /api/food`

Creates a new food item. Requires admin session.

Request Body: (`multipart/form-data`)

Field	Type	Description
`name`	String	The name of the food item.
`recipe`	String	The recipe or description.

Field	Type	Description
`image`	File	The image file for the item.

Responses:

- **201 Created:** Returns the newly created food item object.
- **400 Bad Request:** `Name, recipe, and image are required.`
- **401 Unauthorized:** `Unauthorized. Please login as admin first.`

Update Food Item

`PUT /api/food/:id`

Updates an existing food item. Requires admin session.

URL Parameters:

Parameter	Type	Description
`id`	Number	The unique ID of the food item to update.

Request Body: (`multipart/form-data`) - All fields are optional.

Field	Type	Description
`name`	String	The new name for the item.
`recipe`	String	The new recipe for the item.
`image`	File	A new image to replace the existing one.

Responses:

- **200 OK:** Returns the fully updated food item object.
- **401 Unauthorized:** `Unauthorized. Please login as admin first.`
- **404 Not Found:** `Food item not found.`

Delete Food Item

`DELETE /api/food/:id`

Deletes a food item. Requires admin session.

URL Parameters:

Parameter	Type	Description
`id`	Number	The unique ID of the food item to delete.

Responses:

- **200 OK:**

  ```
  {
      "message": "Food item deleted successfully."
  }
  ```

- **401 Unauthorized:** `Unauthorized. Please login as admin first.`
- **404 Not Found:** `Food item not found.`

Public User Endpoints

These endpoints are public and do not require authentication.

Get All Food Items

`GET /api/food`

Retrieves a list of all available food items.

Success Response (200 OK): Returns an array of food item objects.

```
[
    {
        "id": 1672531200000,
        "name": "Spaghetti Bolognese",
        "recipe": "A classic Italian pasta dish...",
        "image": "/public/uploads/1672531200000.jpg"
    }
]
```

Get Single Food Item

`GET /api/food/:id`

Retrieves a specific food item by its ID.

URL Parameters:

Parameter	Type	Description
`id`	Number	The unique ID of the food item.

Responses:

- **200 OK:** Returns the requested food item object.
- **404 Not Found:** `Food item not found.`