Restaurant Management System

Project Overview

This application is built using **React**, **Express.js**, **MongoDB**, and **Bootstrap**. It is designed to manage and handle CRUD operations for resturants, allowing to perform tasks such as viewing, adding, editing, and deleting resturants records.

Features

1. **Admin Operations**:
    - CRUD operations for resturants (Create, Read, Update, Delete)
    - View and manage the list of Resturants
    - Edit and delete individual Resturants entries
2. **User Operations**:
    - View the list of Resturants
3. **Resturant Management**:
    - Add new Resturant information
    - Update existing Resturants records
    - Delete a Resturant records

Tech Stack

- **Frontend**: React.js (with functional components, React hooks for state and effects)
- **Backend**: Express.js (for handling API requests)
- **Database**: MongoDB (for storing resturant and admin data)
- **Styling**: Bootstrap (for responsive design)
- **API Client**: Axios (for making HTTP requests to the backend)

Key Files and Structure

1. Route.js

This is the main entry point of the application where routes are defined using React Router. It includes:

- Routes for doctor management (/Resturant/add, /Resturant/getAll/Resturant/getOne /Resturant/edit/:id).

2. Index.js

This file contains the base configuration for making API calls to the server using **Axios**. It sets up the base URL and common headers for API requests.

3. ResturantController.js

This service file defines methods for interacting with the backend APIs related to User. It contains CRUD methods for handling user data.

- **Methods**:
  - **create**: To add a new Resturant.
  - **readAll**: To retrieve a list of all Resturants.
  - **readOne**: To retrieve information for a specific resturant by ID.
  - **update**: To update a resturant's information.
  - **delete**: To delete a resturant record.

4. Restaurant.jsx

This component displays a list of resturants. It uses useEffect to call the readAll API and fetch the data. The list includes options for:

- Viewing resturant details.
- Editing resturants information.
- Deleting a resturant.

5. Edit.jsx

This component allows admins to edit the details of an existing resturant. It uses the useParams hook to get the resturant ID from the URL and then fetches the corresponding resturant information using readOne API. The form allows for updating name, Cuisines, phone number, and location.

6.Add.jsx

This component provides a form for adding new resturants to the system. It allows admins to input details such as the resturants's name, location, cuisines, and phone number.

API Endpoints

1. GET /resturant

- **Purpose**: Retrieves a list of all resturants.
- **Response**: Returns a JSON array of resturant objects.

2. GET /resturant/{id}

- **Purpose**: Retrieves information for a specific resturant by ID.
- **Response**: Returns a JSON object with resturants details.

3. POST /resturants

- **Purpose**: Adds a new resturant to the system.
- **Request Body**: A JSON object containing the resturant's name, Cuisine, phone number, and location.

4. PUT /resturants/{id}

- **Purpose**: Updates information for a specific resturant by ID.
- **Request Body**: A JSON object containing the updated resturant details.

5. DELETE /resturant/{id}

- **Purpose**: Deletes a resturant from the system.
- **Response**: Returns a message indicating the success or failure of the deletion.

Restaurant Data Model

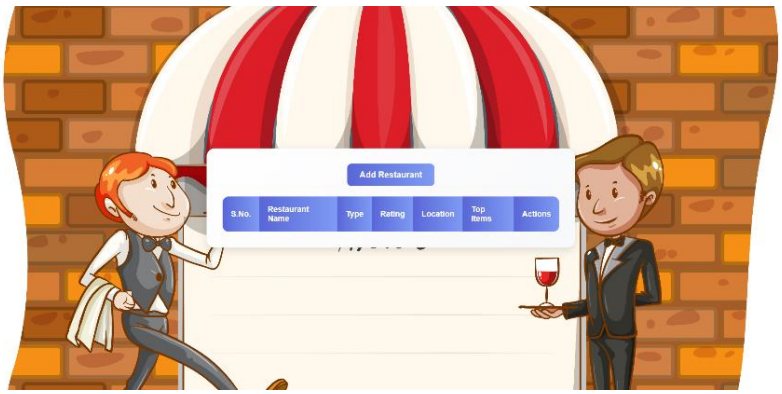
Restaurant

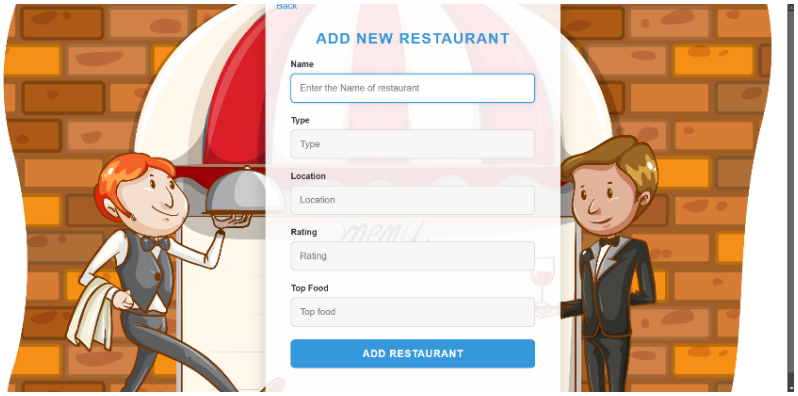
- **id** (String): Unique identifier for the restaurant (MongoDB _id).
- **name** (String): Name of the restaurant.
- **type** (String): Cuisines.
- **top_food** (String): Restaurant’s top food
- **location** (String): Location where the restaurant present.
- **rating** (Number): Rating of the restaurant

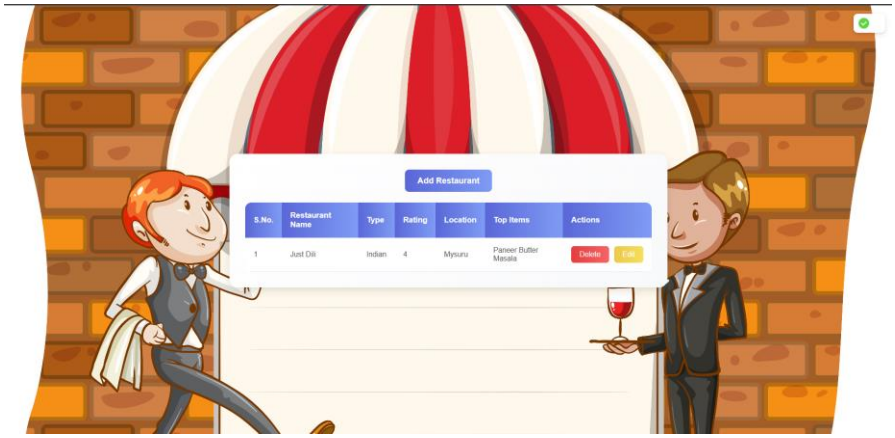
How to Run the Application

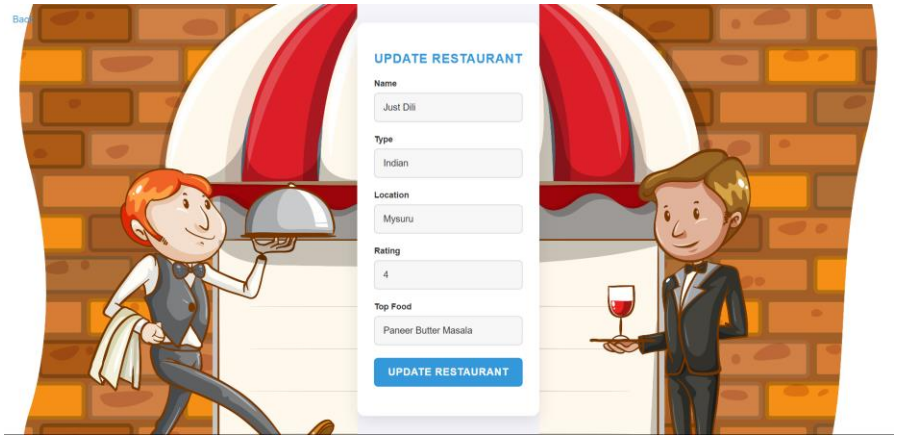
1. **Backend (Express)**:
    - Navigate to the backend directory.
    - Run npm install to install dependencies.
    - Start the backend server: node app.js.
2. **Frontend (React)**:
    - Navigate to the frontend directory.
    - Run npm install to install dependencies.
    - Start the React development server: npm start.
3. **Database (MongoDB)**:
    - Ensure MongoDB is running and the database is set up with the necessary collections (doctors and admins).

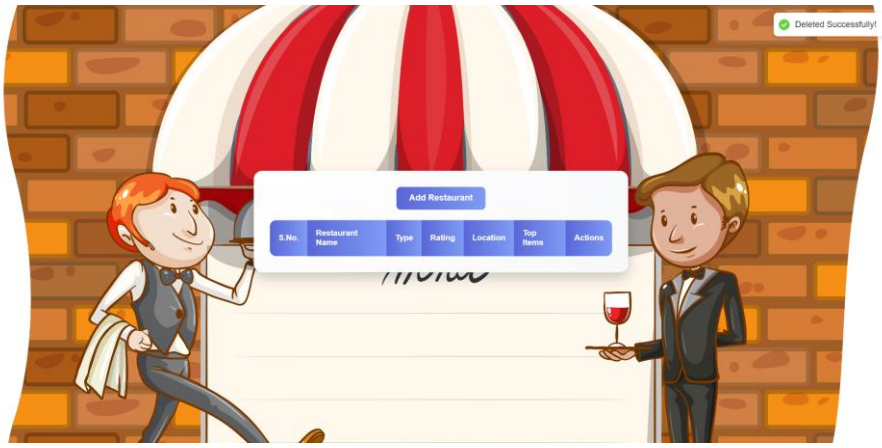
Screenshots:











Database:

Documents 1 | Aggregations | Schema | Indexes 1 | Validation

Type a query: { field: 'value' } or Generate query

Explain | Reset | Find | Options

ADD DATA | EXPORT DATA | UPDATE | DELETE

25 | 1 - 3 of 3

```
_id: ObjectId('6737228d031e5d3924f3ff7b')
name : "Just Dilli"
type : "Indian"
location : "Mysuru"
rating : 3
top_food : "Paneer Butter Masala"
__v : 0
```

```
_id: ObjectId('673722c2031e5d3924f3ff7f')
name : "The Spicy Spoon"
type : "Indian"
location : "Mumbai"
rating : 4
top_food : "Butter Chicken"
__v : 0
```

```
_id: ObjectId('673722e3031e5d3924f3ff83')
name : "Bella Italia"
type : "Italian"
location : "Italy"
rating : 3
top_food : "Pasta Carbonara"
__v : 0
```

Conclusion

This application provides a complete solution for managing restaurants records, with features such as adding, updating, and deleting restaurant information.