Thinkbridge – FoodieDelight Report

Github: https://github.com/ashgole/Thinkbridge_Foodiedelight_MERN

Title: Hotel Management System

Description:

The Hotel Management System is a comprehensive web application designed to streamline the management of hotel operations. Built with React on the frontend and Express.js on the backend, this system provides an efficient and user-friendly interface for managing hotel data, including restaurant details and menu items.

Key features include:

- **State Management with Redux**: Centralized state management using Redux for seamless handling of restaurant data and search terms.
- **Routing with React Router**: Smooth navigation between different components such as the Restaurant List and Add Restaurant forms.
- **Forms and Input Handling**: Controlled components with validation to ensure accurate data entry.
- **Styling with Tailwind CSS**: Modern and responsive user interface using Tailwind CSS.
- **API Integration**: Utility functions for interacting with backend APIs and handling CRUD operations.
- **Pagination and Search**: Efficient handling of large datasets with pagination and search functionality.
- **Image Uploading Feature**: Middleware for handling image uploads, ensuring images are processed and stored correctly on the server.

Process to run:

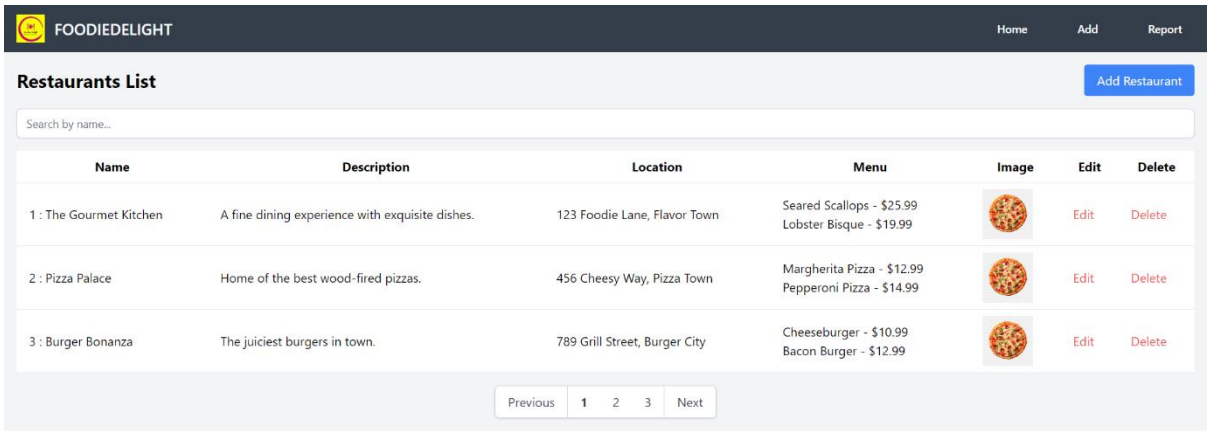
- Extract zip file
- Open in vscode , code .

- cd backend
- npm i
- npm run dev

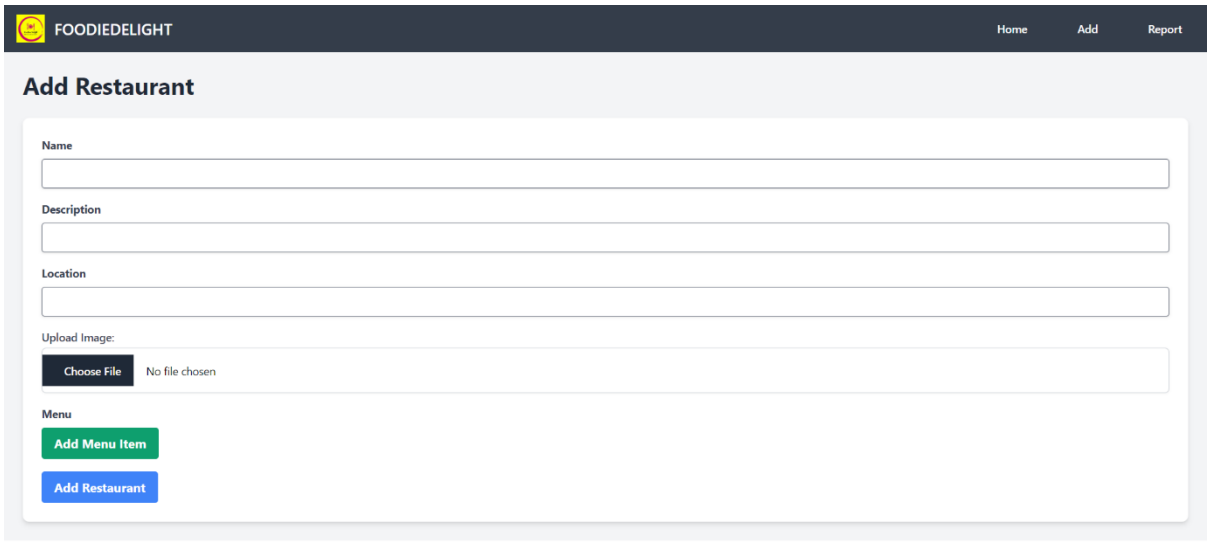
- cd frontend
- npm i
- npm run dev

Screenshots:

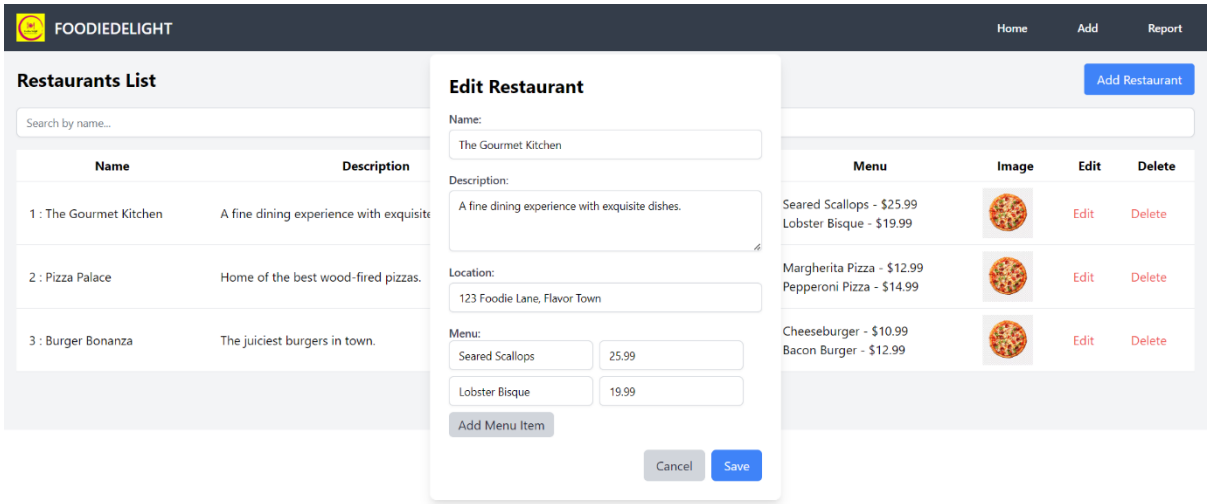
Home -



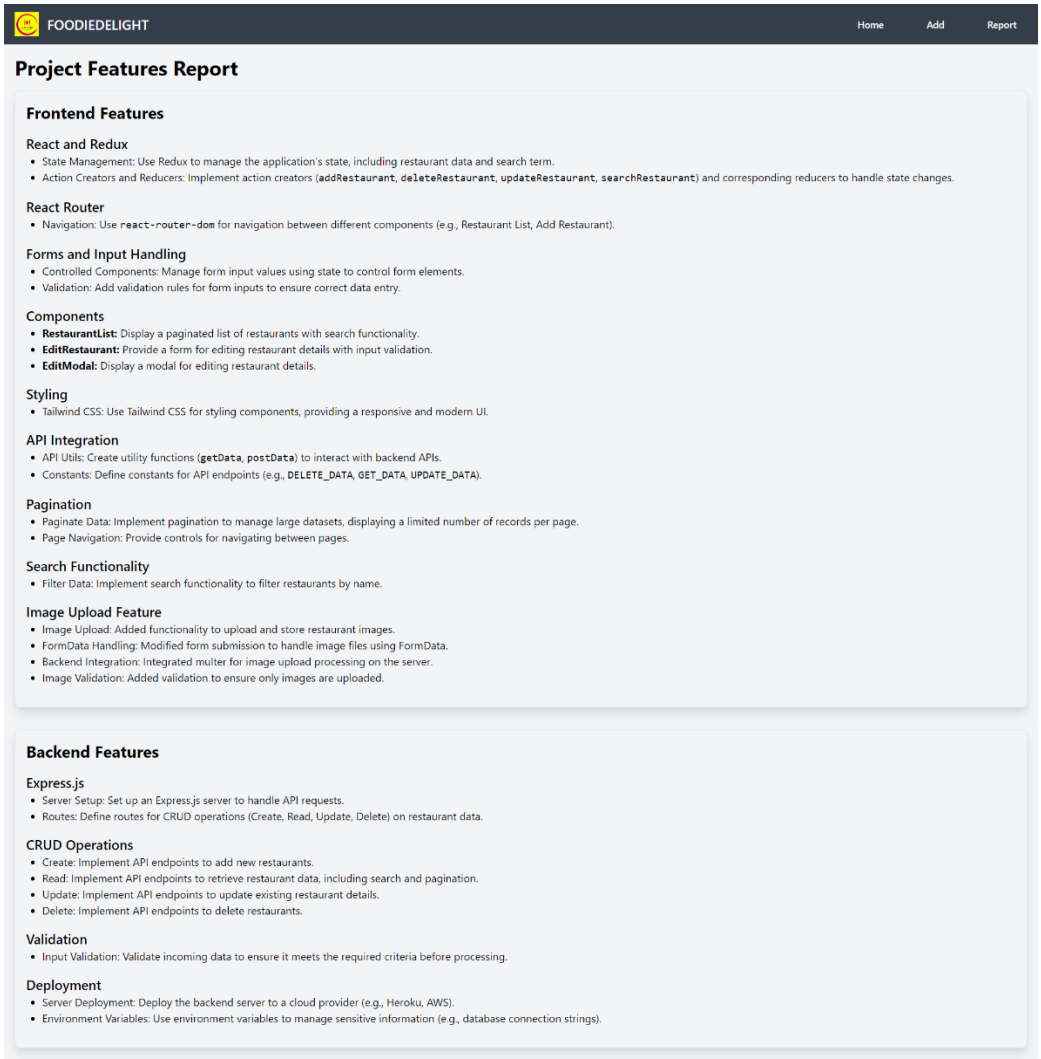
Add Restaurant Component –



Edit Restaurant Component -



Report –


FOODIEDELIGHT
Home
Add
Report
Project Features Report
Frontend Features
React and Redux
State Management: Use Redux to manage the application's state, including restaurant data and search term.
Action Creators and Reducers: Implement action creators (addRestaurant, deleteRestaurant, updateRestaurant, searchRestaurant) and corresponding reducers to handle state changes.
React Router
Navigation: Use react-router-dom for navigation between different components (e.g., Restaurant List, Add Restaurant).
Forms and Input Handling
Controlled Components: Manage form input values using state to control form elements.
Validation: Add validation rules for form inputs to ensure correct data entry.
Components
RestaurantList: Display a paginated list of restaurants with search functionality.
EditRestaurant: Provide a form for editing restaurant details with input validation.
EditModal: Display a modal for editing restaurant details.
Styling
Tailwind CSS: Use Tailwind CSS for styling components, providing a responsive and modern UI.
API Integration
API Utils: Create utility functions (getData, postData) to interact with backend APIs.
Constants: Define constants for API endpoints (e.g., DELETE_DATA, GET_DATA, UPDATE_DATA).
Pagination
Paginate Data: Implement pagination to manage large datasets, displaying a limited number of records per page.
Page Navigation: Provide controls for navigating between pages.
Search Functionality
Filter Data: Implement search functionality to filter restaurants by name.
Image Upload Feature
Image Upload: Added functionality to upload and store restaurant images.
FormData Handling: Modified form submission to handle image files using FormData.
Backend Integration: Integrated multer for image upload processing on the server.
Image Validation: Added validation to ensure only images are uploaded.
Backend Features
Express.js
Server Setup: Set up an Express.js server to handle API requests.
Routes: Define routes for CRUD operations (Create, Read, Update, Delete) on restaurant data.
CRUD Operations
Create: Implement API endpoints to add new restaurants.
Read: Implement API endpoints to retrieve restaurant data, including search and pagination.
Update: Implement API endpoints to update existing restaurant details.
Delete: Implement API endpoints to delete restaurants.
Validation
Input Validation: Validate incoming data to ensure it meets the required criteria before processing.
Deployment
Server Deployment: Deploy the backend server to a cloud provider (e.g., Heroku, AWS).
Environment Variables: Use environment variables to manage sensitive information (e.g., database connection strings).