Full Stack Contribution Report

Overview

The report details the front-end and back-end contributions to the restaurant management system project. Tasks included enhancing user interface elements, adding new features, fixing navigation issues and ensuring a working backend for restaurant/table management.

Frontend Contributions

1. User Interface Enhancements

The CSS of the navigation bar was styled differently to maintain streamlined appearance.

Fixed Navigation Issues:

A broken "Continue Now" button was fixed to redirect users to the login page.

That button that says Reserve Now was fixed so it redirects people to the right page.

2. New Pages Added

A new Page for customers to gain information related to the webapp and what it offers. Style centered around the homepage with it being concise and to the point to retain attention.

A screenshot of a website

AI-generated content may be incorrect.

Dedicated business portal: A Portal was created for potential business organizations to assess the capabilities of table management.

A screenshot of a restaurant

AI-generated content may be incorrect.

Screens screenshot of a computer

AI-generated content may be incorrect.

3. Manager/Admin Features

Admins and managers can edit table details.

Reservation Management Managers view existing/past reservations and manually add new reservations to tables.

A screenshot of a phone

AI-generated content may be incorrect.

4. Customer Features

Reservation System: Customers make reservations for a restaurant by specifying the number of people in their party.

A screenshot of a computer

AI-generated content may be incorrect.

**Backend Contributions**

**1. Completion of Restaurant and Table Routes**

The API routes for managing restaurant and table data were developed and completed. These routes enable the creation and retrieval of restaurant and table records in the system.

* **Restaurant Routes:**
  + A route was implemented to retrieve all reservations for a given restaurant.
  + A route was created to allow for the addition of new reservations for a restaurant.
  + A GET handler was developed to retrieve all restaurants currently listed in the database.
  + A GET handler was created to fetch a specific restaurant based on the provided ID in the URL.
  + A POST method was implemented to add new restaurant entries to the database, ensuring uniqueness.
  + A DELETE method was implemented to remove a restaurant entry from the database.
* **Table Routes:**
  + A route was implemented to fetch all tables for a specific restaurant.
  + A route was created to enable the addition of new tables to a restaurant.
  + A GET handler was created to retrieve all tables associated with a specific restaurant ID.
  + A POST method was implemented to insert a new table linked to a restaurant while maintaining uniqueness constraints.
  + A DELETE method was developed to remove a table from the database.

**2. Unit Testing**

Unit tests were developed for the restaurant and table routes to ensure that they function as expected. These tests validate the correct behavior of the API endpoints, ensuring data is retrieved and created successfully. The handlers were modified to avoid direct interaction with Supabase during testing, utilizing a mocked version of Supabase to ensure accurate test results.

**3. Backend Integration Issue Resolution**

Several issues were identified and resolved within the backend integration, particularly in the restaurant service layer. These changes ensured the smooth operation of the backend and improved overall system performance.

**4. Database Schema Creation and Connection**

New schemas for restaurants and tables were created in Supabase. These schemas were designed to structure the data efficiently and ensure proper relationships between restaurant and table records.

**5. API Documentation**

The necessary references to the API were made, ensuring that the routes and endpoints are well-documented and easily accessible for future development or integration tasks.

**6. Postman Demonstrations**

The implemented handlers were tested using Postman to verify their functionality:

* Retrieving all restaurants from the database.
* Fetching a specific restaurant by ID.
* Adding a new restaurant entry and verifying its creation.
* Deleting an existing restaurant and confirming its removal.
* Retrieving all tables for a given restaurant.
* Adding a new table while enforcing uniqueness constraints.
* Deleting a table and confirming its removal.