System Restaurant Management Application

Introduction

The **System Restaurant Management Application** is designed to streamline and optimize the management of restaurant operations, client relationships, and order tracking. This web-based application provides an intuitive interface for managing key aspects of a restaurant business, including:

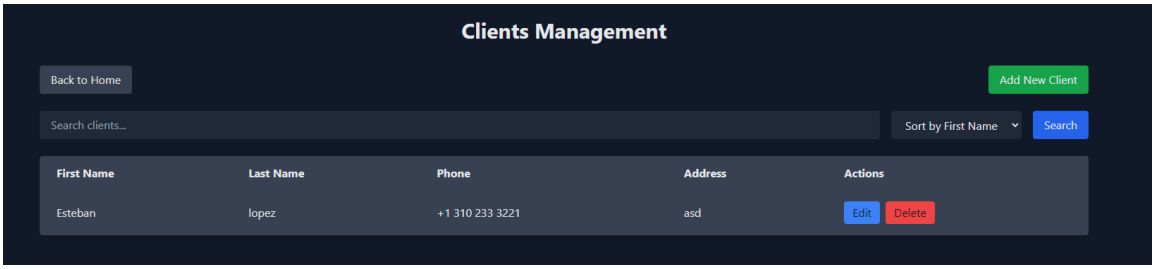
1. **Client Management**
    - Maintain a comprehensive record of client details.
    - Enable efficient client search, addition, editing, and deletion.
2. **Restaurant Management**
    - Organize restaurant details and related information.
    - Provide seamless functionality to add, update, and remove restaurants.
3. **Order Tracking and Management**
    - Simplify order creation and monitoring.
    - Track order statuses and handle related items efficiently.

By offering these features, the application ensures improved operational efficiency, better customer service, and a centralized system for managing critical data. Its user-friendly design and integration with a PostgreSQL

database make it a robust tool for small and medium-sized restaurant businesses.

Features

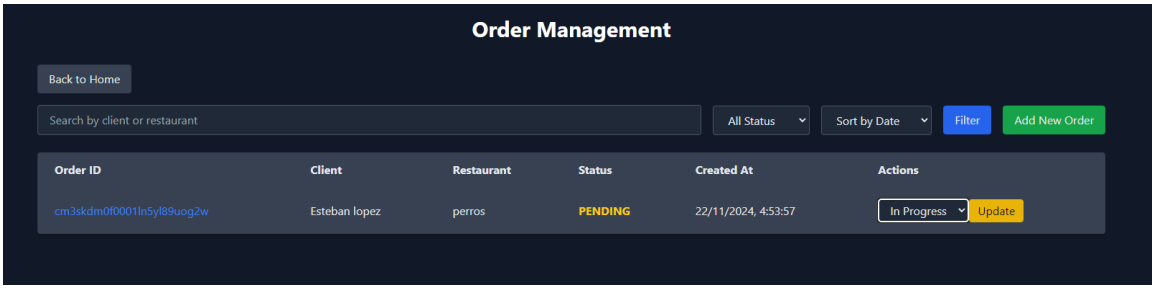
- **Client Management**
  - View and manage a list of clients.
  - Add new clients with complete details like name, phone, and address.
  - Edit existing client information.
  - Delete clients and their associated orders.



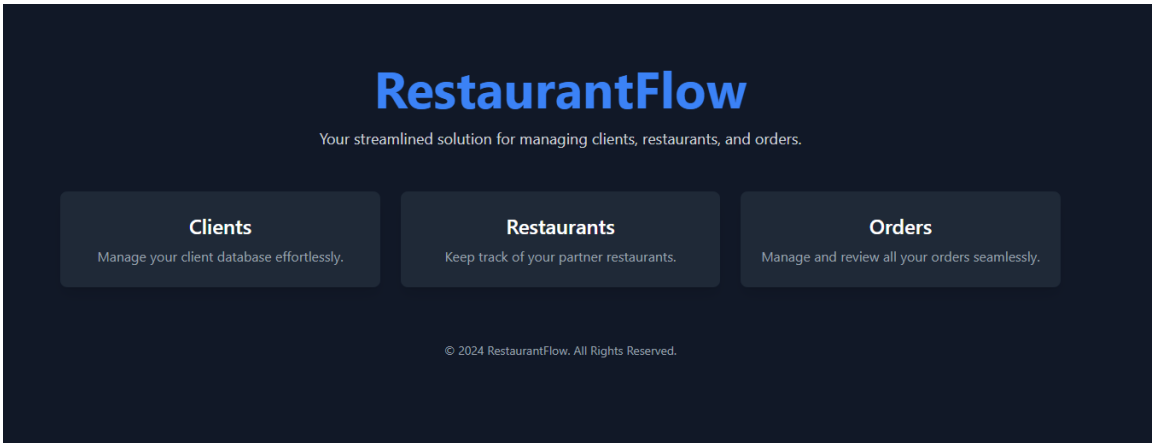
- **Restaurant Management**
  - Maintain a list of restaurants.
  - Add new restaurants with information like name, phone, and address.
  - Update restaurant details.
  - Delete restaurants, cascading the deletion of associated orders.

  Screenshot: *(Drag and drop the screenshot of the "Restaurant Management" page)*
- **Order Management**
  - View a list of all orders with status and details.
  - Filter and sort orders by status or date.
  - Create new orders by selecting a client, restaurant, and adding item details.
  - Update order statuses (e.g., from pending to completed).
  - Delete completed orders directly from the list.

- **Dynamic Item Management in Orders**
  - Add multiple items to an order dynamically.
  - Specify quantity, description, and unit price for each item.
  - Real-time calculation of item totals and validations.



- **Responsive and Styled with Tailwind CSS**
  - Tailored design for an elegant and user-friendly experience.
  - Fully responsive for different devices.



- **Data Integrity with PostgreSQL**
  - Data persistence through a PostgreSQL database.
  - Robust relations for cascading deletions.

Design Choices

1. **Tech Stack**

Technology	Purpose
Remix Framework	Used for building a full-stack web application with efficient routing and server-side rendering.
Tailwind CSS	Simplifies the process of designing and styling with a utility-first approach.
PostgreSQL	Provides a reliable and scalable database for managing the application’s data.
Prisma	Serves as the ORM for connecting the application to the PostgreSQL database with a clear schema.
React	Powers the user interface for dynamic and interactive features.
Vite	Optimizes development and builds with fast bundling.

2. Design Decisions

- **Remix Framework**: Chosen for its server-side rendering capabilities, which provide faster load times and improved SEO.
- **PostgreSQL**: Selected for its robust relational database system, ensuring data integrity and supporting complex queries.
- **Tailwind CSS**: Adopted for quick, consistent, and scalable design patterns, reducing the need to write custom CSS.
- **Component-Based Design**: Follows React's modular philosophy, allowing for reusable components across the application.
- **Cascading Deletions**: Implemented in Prisma to ensure referential integrity when clients or restaurants are removed.

3. Code Snippets

- **Dynamic Client Filtering in Prisma**

```
const clients = await prisma.client.findMany({
  where: {
    firstName: {
      contains: searchQuery,
    },
  },
```

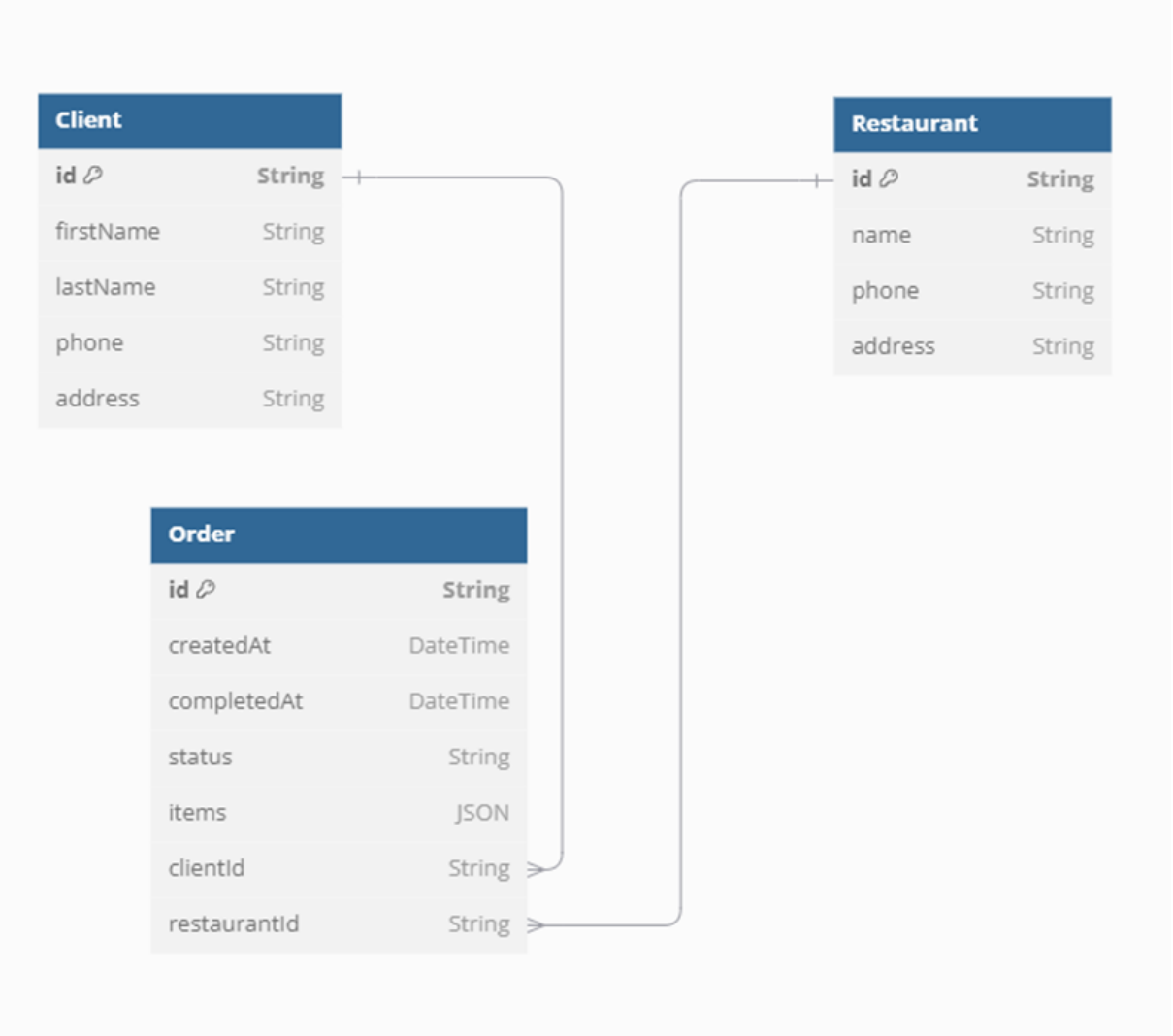
```
  orderBy: { lastName: 'asc' },
});
```

- **Cascading Deletion Example**

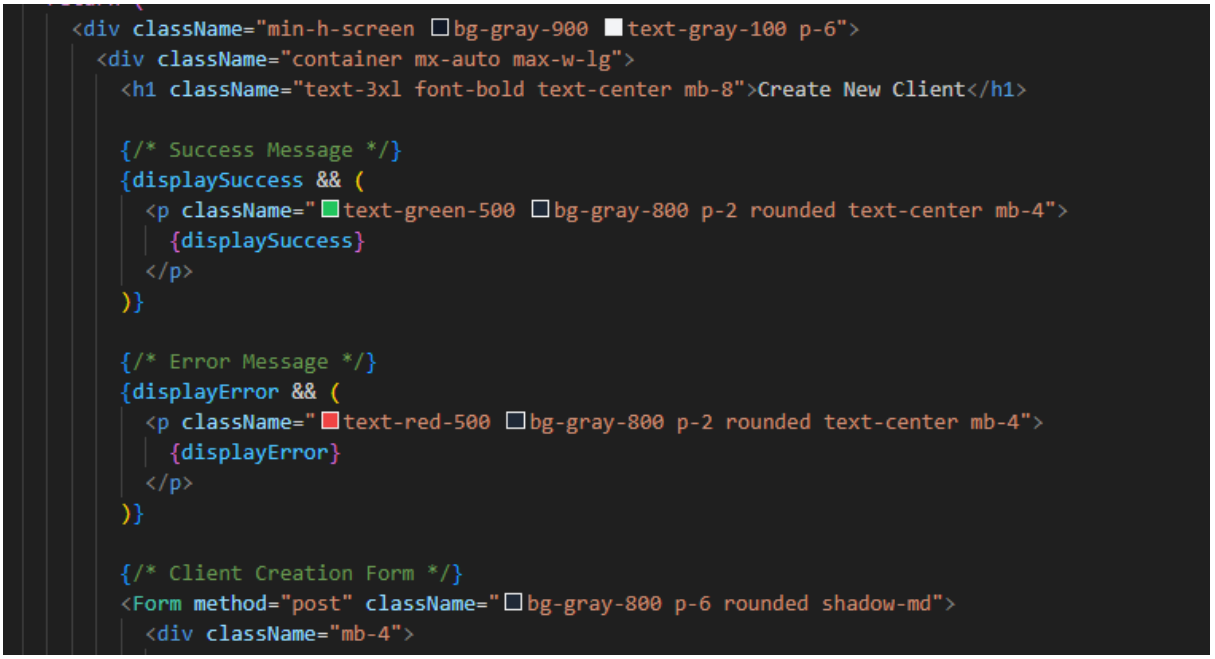
```
await prisma.client.delete({
  where: { id: clientId },
  include: {
    orders: {
      include: { items: true },
    },
  },
});
```

4. Visual Aids

- **Database Schema**:



- **Architecture Diagram**:
  - Client Browser → Remix Framework → Database (PostgreSQL)
- **Code Preview**:



```
<div className="min-h-screen bg-gray-900 text-gray-100 p-6">
  <div className="container mx-auto max-w-lg">
    <h1 className="text-3xl font-bold text-center mb-8">Create New Client</h1>

    {/* Success Message */}
    {displaySuccess && (
      <p className="text-green-500 bg-gray-800 p-2 rounded text-center mb-4">
        {displaySuccess}
      </p>
    )}

    {/* Error Message */}
    {displayError && (
      <p className="text-red-500 bg-gray-800 p-2 rounded text-center mb-4">
        {displayError}
      </p>
    )}

    {/* Client Creation Form */}
    <Form method="post" className="bg-gray-800 p-6 rounded shadow-md">
      <div className="mb-4">
```

5. Best Practices

☑ ~~Follow **DRY principles** (Don't Repeat Yourself) by reusing React components.~~

☑ ~~Use environment variables to manage sensitive information like database credentials.~~

☑ ~~Validate all user inputs to prevent SQL injection and other vulnerabilities.~~

Usage Guide

. Access and Main Navigation

1. **Open the application**: Visit the project URL or launch the local server and open http://localhost:5173.
2. **Main screen**: From the main screen, select one of the following options:
   - **View Clients**: Manage client data.
   - **View Restaurants**: View and manage restaurants.
   - **View Orders**: Manage orders and their associated details.

2. Managing Clients

1. **View the client list**:
   - Navigate to the **Clients** page from the main screen.

- Use the search bar to filter clients by name or phone.

2. **Add a new client**:
   - Click the **"Add New Client"** button.
   - Fill out the form with the client details and click **"Create Client"**.
3. **Edit an existing client**:
   - In the client table, click **"Edit"** next to the client you want to modify.
   - Make the necessary changes and save.
4. **Delete a client**:
   - Click **"Delete"** and confirm. All associated orders will be deleted as well.

3. Managing Restaurants

1. **View the restaurant list**:
   - Go to the **Restaurants** page.
   - Use the search bar to filter by name or phone.
2. **Add a new restaurant**:
   - Click the **"Add New Restaurant"** button.
   - Complete the form and submit the data.
3. **Edit a restaurant**:
   - Select **"Edit"** for the restaurant you want to update.
4. **Delete a restaurant**:
   - Use the **"Delete"** button in the table. Associated orders will also be removed.

4. Managing Orders

1. **View the orders**:
   - Open the **Orders** tab.
   - Filter orders by status (e.g., "Pending," "In Progress," "Completed").
2. **Change the status of an order**:

- Click on the current status of the order and select a new status from the dropdown menu.
- The **"Completed"** status will make the order immutable.

3. **View order details**:
   - Click on the **Order ID** to see full details.
4. **Delete an order**:
   - When the order status is **"Completed"**, use the **"Delete"** button.

5. Creating a New Order

1. Go to the **Orders** tab and click **"Create New Order"**.
2. Select a client and a restaurant from the dropdown menus.
3. Add one or more items to the order:
   - Specify the quantity, description, and unit price.
   - Use the **"Add Item"** button to include more items.
4. Review the data and click **"Create Order"**.

GitHub Repository

```
https://github.com/David-lpz28/System_Restaurant
```