**Restaurant Reservation System API Requirements Document**

**1. Introduction**

This document outlines the requirements for a REST API server designed for a restaurant reservation system. The system will manage reservations, user accounts, restaurant information, and menu details.

**2. General Requirements**

* The API will be RESTful and use JSON as the data exchange format.
* The API will be secured with authentication and permissions.
* The API will provide endpoints for CRUD operations on all resources.

**3. Functional Requirements**

**3.1 User Management**

* **Registration**: Users can register by providing their name, email, and password.
* **Authentication**: Users can log in and out of the system.
* **Profile Management**: Users can view and edit their profiles.

**3.2 Restaurant Information**

* **List Restaurants**: Users can list all available restaurants with pagination.
* **View Restaurant Details**: Users can view detailed information about a restaurant, including location, menu, and availability.
* **Search**: Users can search for restaurants by name or location.

**3.3 Reservations**

* **Make a Reservation**: Users can make a reservation by specifying the date, time, number of people, and any special requests.
* **Modify Reservation**: Users can change or cancel their existing reservations.
* **View Reservations**: Users can view their upcoming and past reservations.

**3.4 Menu Management**

* **View Menu**: Users can view the current menu items for each restaurant.
* **Add/Update Menu Items** (Admin): Admin users can add or update menu items for a restaurant.

**3.5 Table Management (Admin)**

* **Manage Tables**: Admin users can add, remove, or modify tables within the restaurant, including the number of seats.

**3.6 Reporting (Admin)**

* **Reservation Reports**: Admin users can generate reports on reservation patterns, peak times, and customer preferences.

**4. Non-Functional Requirements**

* **Performance**: The API should handle a high number of concurrent users and reservations.
* **Scalability**: The system should be able to scale to accommodate growth.
* **Reliability**: The system should have a high uptime and provide accurate information.
* **Security**: User data should be encrypted, and all transactions should be secure.

**5. Data Model**

A conceptual data model that identifies the main entities and relationships:

* **User**: Stores user information.
* **Restaurant**: Stores restaurant details.
* **Reservation**: Links a user to a restaurant with specific reservation details.
* **Menu Item**: Stores details about the food items available.
* **Table**: Represents the tables within a restaurant.

**6. Security Requirements**

* **Authentication**: Users must be authenticated to make reservations.
* **Authorization**: Certain actions should be restricted to admin users, such as managing the menu and tables.

**7. Technical Requirements**

* **Technology Stack**: Java with Spring Boot, Spring Data JPA, and a relational database like PostgreSQL.
* **API Documentation**: Swagger or OpenAPI for API documentation.
* **Testing**: Unit and integration tests covering all endpoints.

**8. Compliance**

* The system will comply with relevant data protection and privacy laws.

**9. Deployment**

* The API will be containerized using Docker for easy deployment.