RESTAURANT MANAGEMENT SERVICE

📌 Project Logic Walkthrough: Restaurant Management Service

This **Restaurant Management Service** project is developed to streamline the internal workflow of a restaurant and ensure efficient handling of core operations including table reservations, menu item management, order processing, and payment handling. The system is structured in a modular and maintainable way, using well-defined entities and service layers to separate responsibilities.

🧩 Core Entities and Their Responsibilities

1. **Booking**
    - Represents a reservation made by a customer.
    - Fields: bookingId, customerId, bookingTime, numberOfPeople.
    - Purpose: To allocate tables efficiently and prevent overbooking.
2. **MenuItem**
    - Represents a dish or item available in the restaurant.
    - Fields: menuItemId, name, description, price, availability.
    - Purpose: To manage the menu and serve as a reference in orders.
3. **Order**
    - Represents a customer's order placed during or after booking.
    - Fields: orderId, bookingId, waiterId, status, createdAt.
    - Purpose: To record and track the progress of customer orders.

4. **OrderItem**
    - Represents individual menu items within an order.
    - Fields: orderItemId, orderId, menuItemId, quantity, totalPrice.
    - Purpose: To detail out what exactly is ordered within each order.
5. **Payment**
    - Handles payment information for a particular order.
    - Fields: paymentId, orderId, amount, paymentMethod, paymentTime.
    - Purpose: To ensure each order has a verifiable payment record.

🔄 Key Functional Flows

1. Booking a Table

- Customers provide booking details (number of people, time).
- The system checks for table availability and creates a Booking entry.
- This step prevents crowding and helps kitchen & waiter staff prepare in advance.

2. Placing an Order

- Once the customer is seated, a waiter initiates an Order linked to the Booking.
- The order contains multiple OrderItem entries, each referencing a MenuItem.
- Quantities and prices are stored to calculate the final bill.

3. Processing Payment

- After completing the meal, the customer proceeds to payment.
- The waiter or manager records payment using the Payment entity.
- Includes validation for accepted payment methods (e.g., Cash, Card).

- Ensures order-wise traceability and financial reporting.

4. Updating Order Status

- Orders have statuses such as: Pending, Preparing, Ready, Served, Completed.
- Kitchen staff and waiters can update status as the order progresses.
- This enables real-time order tracking and reduces miscommunication.

5. Generating Reports

- Admins or managers can generate analytical reports.
- Reports can include:
    - Total number of orders per day/week/month.
    - Income reports from Payment data.
    - Most ordered menu items.
    - Booking statistics and customer traffic trends.

⚙️ Implementation Logic

- Each entity (Booking, MenuItem, Order, OrderItem, Payment) is implemented as a Java class.
- Fields are encapsulated using **private access modifiers**, and accessed via **getters and setters**.
- Constructors (default and parameterized) help in object initialization.
- Relationships are managed through foreign keys using IDs (e.g., bookingId, orderId, menuItemId), ensuring referential integrity.

The service layer handles the core business logic like validation, state transitions, and calculating totals.

The DAO layer interacts with the database (via JDBC or JPA) to persist entities.

🧠 Design Decisions

- **Modularity**: Each operation is encapsulated into its own class/service making the system scalable and easier to test.
- **Data Integrity**: Use of constraints (like foreign keys) ensures clean relationships.
- **Auditing**: Timestamps and status fields help in tracking lifecycle events.
- **Extensibility**: Easy to integrate with a frontend or mobile interface for real-time updates.

📊 Conclusion

The Restaurant Management Service is designed to handle real-world restaurant operations efficiently by providing a robust backend architecture. It streamlines reservation management, optimizes kitchen coordination, tracks orders and payments reliably, and equips restaurant management with data-driven insights through reporting. Each entity and flow is carefully designed to reflect real-life use cases, and the Java-based implementation ensures cross-platform support and maintainability.

2. GitHub Repository Link

https://github.com/kruthikab_Zeta/RestaurantManagementService

3. Instructions to Run the Project

Prerequisites

- Java JDK 8 or above
- IntelliJ IDEA (or any Java IDE)
- Database setup

Steps

- Clone the repository:
- git clone git@github.com:kruthikab_Zeta/RestaurantManagementService.git
- Open the project in IntelliJ IDEA.
- Build the project using the IDE's build tools.
- Run the main application class (if provided) or execute test cases.