Restaurant Management System

GitHub Repository: https://github.com/Anees02/Restaurant-Management-System

Project Overview:

This project is a **Restaurant Management System** built in **Java (JDBC + OOP principles)** with a modular structure. It simulates the core operations of a restaurant, including:

- Customer Management → registration, login, placing orders
- Employee Management → managers, waiters, chefs, and admins
- Table Management → reserving, releasing, and tracking table status
- Order Management → creating orders, adding food items, updating quantities, bill generation
- Menu Management → adding and updating food items by admin

Logic Walkthrough:

1. Core Entities

- **Customer** → stores personal and login details
- **Employee** → identified by role (MANAGER, CHEF, WAITER, ADMIN)
- **FoodItem** → menu items with name, price, availability
- **RestaurantTable** → each table has a status (AVAILABLE, RESERVED, OCCUPIED)
- **Order & OrderItem** → link customers, tables, and food items

2. Repositories (Data Access Layer)

- Singleton pattern ensures only one DB connection is active.
- Each repository (`CustomerRepository`, `OrderRepository`, etc.) handles CRUD operations for its entity.
- Example: **`OrderRepository`** → `createOrder()`, `updateQuantity()`, `getOrdersByCustomer()`.

3. Service Layer (Business Logic)

- **AdminService** → add/remove food items, view sales
- **CustomerService** → registration, login, place order
- **OrderService** → handle new orders, update order items, compute bill
- **TableService** → manage reservations with threading (e.g., auto-cancel table if not occupied within 20 mins).

4. Threading Logic

- When a customer books a table → a timer thread starts.
- If not occupied within 20 minutes → reservation is automatically canceled → table set back to `AVAILABLE`.

5. Execution Flow Example

- Customer logs in → books a table → places an order → waiter updates status → chef prepares → order completed → bill generated.

Features

- Customer registration and login with email validation.
- Table booking and automatic release if the customer does not attend within 20 minutes.
- Order creation and item management (quantity updates, status updates).
- Pending item tracking for chefs.
- Bill generation and payment processing.
- CRUD operations for food items by admin.
- Daily sales reporting.

Logic Flow

1. **Customer Operations:**
   - Registration and login with validation.
   - Place orders after booking a table**.**
2. **Table Management:**
   - Tables can be booked if available.
   - Automatically released if not attended within 20 minutes.
3. **Order Management:**
   - Waiters create orders, add/update items.
   - Chefs process pending items and update item status.
4. **Billing and Payments:**
   - Manager generates bills for prepared items.
   - Payments marked as completed and table is freed.
5. **Admin Operations:**
   - Add, update, or remove food items.
   - Generate daily sales report from orders.

Technical Features

- **Multithreading & Concurrency**
  - ScheduledExecutorService auto-releases tables if customers don't show up within 20 minutes.
  - synchronized booking prevents multiple customers from booking the same table at once.
- **Design Patterns**
  - **Singleton:** Ensures single instances of repository classes and database connections.
  - **DAO Pattern:**Repositories handle all SQL interactions separately from business logic.
- **Robust Architecture**
  - Layered structure with Model → Repository → Service → Command Interface.
  - Loose coupling and separation of concerns for maintainability.
- **Validation & Error Handling**
  - EmailValidator ensures correct email format using regex.
  - Custom exceptions (CustomerNotFoundException, PasswordIncorrectException, etc.) for clear error messages.
  - Logging – Integrated with Slf4j for debugging and error tracking.

Prerequisites

Before running the application, ensure the following are installed on your system:

- **Java JDK 17 or above** – for compiling and running Java files.
- **PostgreSQL** – for database storage.
- **Git** – to clone the repository.
- **IDE**: prefer IntelliJ IDEA (Other options Eclipse or VS Code)

Instructions to Run the Project:

1.Clone Github Repository

Open your terminal/command prompt and run:

```
git clone https://github.com/Anees02/Restaurant-Management-System.git
cd Restaurant-Management-System
```

2. Database Setup

1. Navigate to the `src/main/resources/` folder.
2. Copy and execute the `db_script.sql` file in your PostgreSQL Terminal or PG Admin.
    - This will create all the required tables and schema.

3. Configure Database Connection

In the same `resources` folder, open the `application.properties` file.
Update the following properties with your PostgreSQL credentials:

```
database_url = jdbc:postgresql://localhost:5432/your_database_name
username = your_username
password = your_password
```

4. Run the Application

1. Open the project in your IDE (IntelliJ IDEA / Eclipse / VS Code with Java support).
2. Locate the **`Main`** class in the package:

    `tech.zeta.commandInterface.Main`

3. Run the `Main` file directly by clicking the Play Button (especially in InteliJ IDE)
4. The command-line interface will launch and allow you to interact with the system.