CodeLogic.pdf

1. Project Overview

Project Title: Restaurant Management System

Objective:
Design and implement a digital system for a mid-sized restaurant to manage table bookings, orders, kitchen workflow, billing, and staff efficiently. The system supports concurrent users and real-time updates.

Key Features:
- Customers can book tables online.
- Waiters take orders via tablets and send them to the kitchen.
- Kitchen staff see live orders and mark them as prepared.
- Managers generate bills and record payments.
- Admins manage menus, prices, and daily sales reports.

2. Logic Walkthrough

A. Database Design

- PostgreSQL backend.
- Main entities: Customer, RestaurantStaff, TableInfo, Booking, MenuItem, Order, OrderItems, Billing.
- Relationships:
  - Customer -> Booking
  - Booking -> TableInfo
  - Order -> TableInfo, OrderItems, Billing
  - OrderItems -> MenuItem
  - RestaurantStaff -> Order (as waiter)

B. DAO Layer

- Each entity has a DAO interface and an implementation class (<<DAO>>).
- Methods include add, update, delete, and specialized queries.
- DAOs use thread-safe singleton DBConnection for PostgreSQL interactions.

C. Service Layer

- Handles business logic and orchestrates DAO calls.
- Main services:

  - CustomerIntegratedServices: Manages customer interactions and bookings.
  - OrderService: Takes orders, updates kitchen status, and completes orders.
  - BillingService: Generates bills and records payments.
- Services ensure transactional consistency.

D. Booking Logic

- Booking is checked for past or expired times.
- Bookings older than 30 minutes are canceled automatically.
- Otherwise, booking ID is used to link orders.

E. Order Flow Logic

1. Waiter assigned dynamically from available staff.
2. Customer places order -> OrderItems are created.
3. Kitchen prepares order -> updates status to READY.
4. Waiter serves items -> status updated to SERVED.
5. Table status updated to ACTIVE after completion.

F. Billing Logic

- Billing DAO generates a new record per order.
- Returns generated bill_id for reference.
- Supports payment status updates.

G. Utility

- DBConnection singleton manages database connections.
- Ensures a single point of connection and prevents multiple connections.

3. GitHub Repository

Link:
https://github.com/PremansuChakraborty/Restaurant-project.git

4. Instructions to Run the Project

1. Prerequisites:
   - Java JDK 17+ installed
   - PostgreSQL installed and running
   - Maven installed

2. Clone the Repository:
   git clone https://github.com/PremansuChakraborty/Restaurant-project.git

```
cd RestaurantManagementSystem
```

3. Configure Database:
   - Create database restaurant_db
   - Run SQL scripts in resources/db/schema.sql to create tables
   - Update application.properties with DB username and password

4. Build the Project:

```
mvn clean install
```

5. Run the Project:
   - From IDE: Run tech.zeta.Main class
   - From terminal:

```
mvn exec:java -Dexec.mainClass="tech.zeta.Main"
```

6. Test Features:
   - Follow console prompts for booking, ordering, and billing
   - Check PostgreSQL tables for updated records

5. Notes

- Logging via Slf4j for tracing workflow.
- Thread-safe singleton ensures connection stability.
- Enum types used for statuses: RestaurantStaffStatus, OrderItemEnum.
- Modular design allows future extension (GUI or REST API).