Architecture Document for Restaurant Management System

1. Introduction

1.1 Purpose

The purpose of this document is to outline the architecture of a Restaurant Management System (RMS), which supports various functions such as menu management, order processing, reservation management, kitchen coordination, and billing. This system is intended to streamline restaurant operations by providing a centralised platform for efficient service delivery and operational management.

1.2 Scope

The RMS will serve as a comprehensive system for restaurants of all sizes, offering modules for customer interactions (e.g., placing orders, reservations), administrative functions (e.g., updating menu, managing reservations), and backend operations (e.g., kitchen management, billing, and order tracking). The system will operate across desktop PCs and tablets, and integrate seamlessly with existing hardware like printers, card readers, and cash registers.

1.3 Definitions, Acronyms, and Abbreviations

- **RMS**: Restaurant Management System
- **POS**: Point of Sale
- **UI**: User Interface
- **DBMS**: Database Management System
- **API**: Application Programming Interface
- **RFID**: Radio Frequency Identification

1.4 References

- IEEE Standard for Software Architecture Descriptions (IEEE 1471)
- Design and Development of Multi-touchable E-restaurant Management System by Soon Nyean Cheong et al.
- E-Restaurant: Online Restaurant Management System for Android by Maderla Rajesh et al.

---

2. Architectural Representation

The RMS follows a layered architecture with clearly defined subsystems for managing different functionalities such as authentication, reservations, order processing, and kitchen coordination. The architecture is divided into client-side applications for customers and staff, and server-side systems for database management and real-time communication between subsystems.

---

3. Architectural Goals and Constraints

- **Goals**:
    - Provide an intuitive interface for users to interact with the system.
    - Ensure scalability to handle varying workloads in small and large restaurants.
    - Maintain high levels of security for sensitive customer and business data.
    - Achieve seamless integration with existing restaurant hardware (e.g., printers, cash registers).
- **Constraints**:
    - Compliance with local food safety and financial regulations.
    - Compatibility with restaurant POS hardware and communication protocols.
    - Limited budget and resources for implementation in small-scale restaurants.

---

4. Use-Case View

4.1 Architecturally-Significant Use Cases

1. **Order Placement**: Customers can place orders via touchscreens or mobile apps.
2. **Reservation Management**: Customers can make and update reservations in real time.
3. **Menu Management**: Restaurant staff can update menu items, pricing, and availability.
4. **Kitchen Coordination**: Orders are sent directly to the kitchen with clear itemization and table numbers.
5. **Billing**: The system integrates with the POS for billing and payment processing.

---

5. Logical View

5.1 Architecture Overview – Package and Subsystem Layering

- **User Interface Layer**: Handles interactions with customers (ordering, reservations) and staff (menu updates, order tracking).
- **Business Logic Layer**: Contains the core functionality for managing orders, reservations, and communication with kitchen staff.
- **Data Layer**: Interacts with the database to store customer orders, menu items, and transaction details.

- **Integration Layer**: Provides APIs for connecting with external systems such as payment gateways, inventory management, and billing systems.

---

6. Process View

6.1 Processes

1. **Customer Order Process**: Customer selects menu items, which are sent to the kitchen and simultaneously stored in the database.
2. **Reservation Process**: Customers can book tables via mobile apps or restaurant kiosks. These reservations are managed in the database and synced with the system's scheduling.
3. **Order Fulfilment Process**: Orders are received by kitchen staff and marked as completed once served, updating the backend and billing system.
4. **Billing Process**: The system calculates the final bill, sends the payment request to the POS, and updates the order status after payment.

6.2 Process to Design Elements

- **Order Process**: Mapped to the User Interface and Business Logic layers.
- **Reservation Process**: Integrated into the Data Layer for tracking and updating reservations in real time.
- **Billing Process**: Tied to the Integration Layer for interaction with external payment systems.

6.3 Process Model to Design

The processes for order management, reservation handling, and billing are mapped to their respective modules in the architecture. The client-side application interacts with the server-side DBMS to ensure data is consistent across all systems.

6.4 Model Dependencies

- **Customer Interaction**: Depends on both the User Interface Layer and Data Layer for real-time updates and accurate information.
- **Kitchen Coordination**: Relies on fast communication between the Data Layer and Business Logic for seamless order fulfilment.
- **Payment Systems**: Dependent on the Integration Layer to securely process transactions.

6.5 Processes to the Implementation

The processes are implemented through REST APIs, real-time database synchronisation, and efficient communication protocols to handle multiple orders and transactions concurrently.

7. Deployment View

7.1 External Desktop PC

Used by restaurant managers to handle administrative functions such as menu updates, report generation, and system configurations.

7.2 Tablet Interface

Deployed on tables or customer devices for order placement, browsing the menu, and reservations.

7.3 Kitchen Display System

Receives real-time updates on customer orders with detailed breakdowns of each dish, including any special requests or customizations.

7.4 POS System

Handles billing and payments, connected to the system for real-time updates on completed orders and payments.

---

8. Performance

The RMS is designed to handle a high volume of orders, ensuring real-time updates across all subsystems. Average response times for order processing are under 2 seconds, and the system can handle up to 100 concurrent users without performance degradation.

---

9. Quality

The system is built with the following quality attributes:

- **Scalability**: Supports both small and large restaurants.
- **Reliability**: High availability with fault-tolerant mechanisms.
- **Security**: End-to-end encryption for data transmission and secure user authentication.
- **Usability**: Intuitive design tailored for both customers and restaurant staff.
