**CMPSC221 Section (001)**

Restaurant Management System

Prepared By

(Mohammad Abu Shariha, Jaiyoung Cho)

**Instructor: Shahid Hussain**

**Date: (12/09/2024)**

**Table of Contents**

[1 Project Description 3](#_Toc184688493)

[2 List of Concepts and Functions 3](#_Toc184688494)

[3 Association between concepts 7](#_Toc184688495)

[4 Use Case Diagram 8](#_Toc184688496)

[5 Activity Diagram 11](#_Toc184688497)

[6 Class Diagram 12](#_Toc184688498)

[7 Finite State Machine 12](#_Toc184688499)

[8 User Manual 13](#_Toc184688500)

# 1 Project Description

A restaurant management system is essential software designed to streamline and automate various restaurant operations. It enables restaurant staff; such as waiters, managers, and chefs to efficiently handle tasks like taking orders, processing payments, managing reservations, and tracking inventory. Additionally, the system enhances customer experience by allowing them to place orders either in person or online, with the orders being directly communicated to the kitchen, ensuring accuracy and timeliness. This results in improved customer service, better resource management, and increased profitability for the restaurant. It also includes customer feedback and generating reports on sales, customer preferences, and employee performance.

# 2 List of Concepts and Functions

**Concepts:**

* Staff (waiters, managers, and chefs)
* Customers
* Orders
* Payments
* Reservations
* customer Feedback
* Reports (Sales, Customer Preferences, Employee Performance)

**Functions:**

* **Process and manage customer orders:** Allow staff to take and fulfill orders, with automatic updates to the kitchen.
* **Handle table reservations and seating:** Manage reservations, assign tables, and track seating arrangements.
* **Track inventory levels and manage supplies:** Monitor stock levels, alert staff when supplies are low, and manage reordering.
* **Schedule and manage restaurant staff:** Create work schedules, assign shifts, and manage staff availability.
* **Generate sales and inventory reports:** Provide detailed reports on sales, inventory usage, and financial performance.
* **Process payments:** Handle payments from customers, both online and in-person, including the management of different payment methods.
* **Manage customer information and preferences:** Store customer data, track dining preferences, and manage feedback for improved service.

**Entity Description:**

| **Entity** | **Attributes** | **Methods** |
| --- | --- | --- |
| Staff | 1. staffId 2. name 3. position 4. shift 5. contactInformation | 1. setStaffId() 2. getStaffId() 3. setName() 4. getName() 5. setPosition() 6. getPosition() 7. setShift() 8. getShift() 9. setContactInformation() 10. getContactInformation() 11. assignShift() 12. updatePosition() |
| Customer | 1. customerId 2. name 3. contactInformation 4. preferences (e.g., dietary restrictions, favorite dishes) 5. history (e.g., past orders, feedback) | 1. setCustomerId() 2. getCustomerId() 3. setName() 4. getName() 5. setContactInformation() 6. getContactInformation() 7. setPreferences() 8. getPreferences() 9. setHistory() 10. getHistory() 11. updatePreferences() 12. viewOrderHistory() |
| Order | 1. orderId 2. customerName 3. orderDate 4. orderItems (List of items) 5. totalAmount | 1. setOrderId() 2. getOrderId() 3. setCustomerName() 4. getCustomerName() 5. setOrderDate() 6. getOrderDate() 7. setOrderItems() 8. getOrderItems() 9. setTotalAmount() 10. getTotalAmount() 11. addItem() 12. calculateTotalAmount() 13. processOrder() |
| Payment | 1. paymentId 2. orderId 3. amount 4. paymentMethod (e.g., cash, credit card) 5. paymentDate | 1. setPaymentId() 2. getPaymentId() 3. setOrderId() 4. getOrderId() 5. setAmount() 6. getAmount() 7. setPaymentMethod() 8. getPaymentMethod() 9. setPaymentDate() 10. getPaymentDate() 11. processPayment() 12. refundPayment() |
| Reservation | 1. reservationId 2. customerName 3. reservationDate 4. table (Table object) | 1. setReservationId() 2. getReservationId() 3. setCustomerName() 4. getCustomerName() 5. setReservationDate() 6. getReservationDate() 7. setTable() 8. getTable() 9. confirmReservation() 10. cancelReservation() |
| Inventory | 1. itemId 2. itemName 3. quantity 4. unitPrice | 1. setItemId() 2. getItemId() 3. setItemName() 4. getItemName() 5. setQuantity() 6. getQuantity() 7. setUnitPrice() 8. getUnitPrice() 9. updateQuantity() 10. checkStockLevel() |
| CustomerFeedback | 1. feedbackId 2. customerId 3. feedbackDate 4. feedbackText 5. rating (e.g., 1-5 stars) | 1. setFeedbackId() 2. getFeedbackId() 3. setCustomerId() 4. getCustomerId() 5. setFeedbackDate() 6. getFeedbackDate() 7. setFeedbackText() 8. getFeedbackText() 9. setRating() 10. getRating() 11. submitFeedback() |
| Report | 1. reportId 2. reportType (e.g., Sales, Customer Preferences, Employee Performance) 3. reportDate 4. reportData (e.g., list of data points, graphs) | 1. setReportId() 2. getReportId() 3. setReportType() 4. getReportType() 5. setReportDate() 6. getReportDate() 7. setReportData() 8. getReportData() 9. generateReport() |

# 3 Association between concepts

**1. One to One**

* **Staff** ↔ **ContactInformation** Each staff member has one set of contact information, and each set of contact information belongs to only one staff member. This relationship ensures that a staff member is uniquely associated with their contact details.
  + **Association**: One to One
* **Order** ↔ **Payment** Each order can have exactly one payment, and each payment is tied to only one order. This reflects the typical one-to-one relationship between an order and its corresponding payment transaction.
  + **Association**: One to One

**2. One to Many**

* **Customer** ↔ **Order** One customer can place multiple orders, but each order is placed by a single customer. This reflects the relationship between a customer and their purchasing activity.
  + **Association**: One to Many
* **Staff** ↔ **Shift** A staff member can be assigned to multiple shifts, but each shift belongs to only one staff member. This models the typical scheduling setup in a restaurant where one staff member works various shifts.
  + **Association**: One to Many
* **Order** ↔ **OrderItems** An order consists of multiple items, but each order item belongs to only one specific order. This represents a customer’s selection of multiple menu items in one transaction.
  + **Association**: One to Many

**3. Many to Many**

* **Customer** ↔ **OrderItems** A customer can order multiple items in different orders, and each item can be ordered by many different customers. This relationship is common in restaurant management where many customers order overlapping items.
  + **Association**: Many to Many
* **Customer** ↔ **Reservation** A customer can make multiple reservations, and a reservation can involve multiple customers, especially in group dining situations. This models the flexibility of reservation systems in restaurants.
  + **Association**: Many to Many

**4. Aggregation**

* **Order** aggregates **OrderItems** An order is an aggregate of multiple items. Although the order groups these items, the items exist independently of the order. For example, the menu items are defined regardless of whether an order is made.
  + **Association**: Aggregation
* **CustomerFeedback** aggregates **Customer** and **Order** Customer feedback relates to both the customer and the order but exists as a separate entity. Feedback is not dependent on a specific customer or order to exist, and it can be reviewed independently.
  + **Association**: Aggregation

**5. Composition**

* **Reservation** ↔ **Table** A reservation cannot exist without a table assignment, as each reservation requires a specific table. If a reservation is deleted, the association with the table also ends.
  + **Association**: Composition
* **Order** ↔ **OrderItems** An order is composed of multiple order items. If the order is deleted, the associated order items are also deleted, as they are part of the order’s composition.
  + **Association**: Composition

**6. Inheritance**

* **NA** There are no specific inheritance relationships in the defined entities of the Restaurant Management System. In a more extended system, you could introduce inheritance between different types of employees (e.g., Chef, Waiter inheriting from Staff), but in this scope, no inheritance is present.
  + **Association**: NA

# 4 Use Case Diagram

A diagram of a diagram

Description automatically generated with medium confidence

This use case diagram represents a restaurant management system and shows the interactions between various actors (e.g., Restaurant Owner/Manager, Supplier, Staff, Chef, and Customer) and the associated use cases. Below is a brief description of each use case:

**1. Check Availability**

* **Actors**: Customer, Staff
* **Purpose**: Allows customers and staff to verify the availability of tables, menu items, or resources.
* **Goal**: Ensure smooth operations and prevent overbooking or stock shortages.
* **Example**: A customer checks if a table for two is available for dinner, or staff checks if a specific ingredient is in stock.

**2. Reservation**

* **Actors**: Customer
* **Purpose**: Enables customers to reserve tables in advance using the system.
* **Goal**: Simplify the booking process and ensure proper table management.
* **Example**: A customer reserves a table for a family dinner at 7 PM.

**3. Order**

* **Actors**: Customer, Staff
* **Purpose**: Facilitates placing orders for food and beverages by customers or staff on their behalf.
* **Goal**: Track all orders for preparation and payment processes.
* **Example**: A customer places an order for pizza and drinks at the restaurant or through a mobile app.

**4. Inventory**

* **Actors**: Staff, Chef, Supplier, Restaurant Owner/Manager
* **Purpose**: Allows management of stock levels and tracking of supplies.
* **Goal**: Ensure adequate resources are available for food preparation.
* **Example**: Staff updates inventory when a shipment arrives from the supplier, or the chef checks for ingredient availability.

**5. Cook**

* **Actors**: Chef
* **Purpose**: Represents the food preparation process based on customer orders.
* **Goal**: Ensure efficient cooking and timely delivery of orders.
* **Example**: The chef prepares a steak after receiving an order from staff.

**6. Serve**

* **Actors**: Staff
* **Purpose**: Reflects the process of delivering prepared food to customers.
* **Goal**: Provide efficient and timely service to enhance the dining experience.
* **Example**: Waitstaff serves food to a table immediately after it’s prepared.

**7. Apply Discounts (<<extend>> from Payment)**

* **Actors**: Customer, Restaurant Owner/Manager
* **Purpose**: Apply applicable discounts to customer payments during checkout.
* **Goal**: Enhance customer satisfaction by offering promotions or rewards.
* **Example**: A customer receives a 10% discount for being a loyalty program member.

**8. Select Payment Method (<<include>> in Payment)**

* **Actors**: Customer
* **Purpose**: Allows customers to choose a preferred payment method during the checkout process.
* **Goal**: Support multiple payment options like cash, card, or digital wallets.
* **Example**: A customer selects "Credit Card" as the payment method for their order.

**9. Payment**

* **Actors**: Customer, Restaurant Owner/Manager
* **Purpose**: Handles customer payments and completes transactions.
* **Goal**: Ensure accurate billing and recordkeeping for all orders.
* **Example**: A customer pays for their meal using a mobile payment app.

**10. Report**

* **Actors**: Restaurant Owner/Manager
* **Purpose**: Generates operational reports, including sales, inventory, and performance data.
* **Goal**: Provide insights to optimize restaurant operations.
* **Example**: The manager reviews a monthly revenue report.

**11. Feedback**

* **Actors**: Customer
* **Purpose**: Allows customers to provide reviews or suggestions on their experience.
* **Goal**: Collect valuable input for service improvement.
* **Example**: A customer rates their dining experience as 4/5 stars and comments on the service.

# 5 Activity Diagram

**Activity Diagram 1: Customer Order Process**

**Description**: This diagram shows the flow of activities involved in a customer placing an order in the restaurant, from selecting items to confirming the order.

**Key Activities:**

* **Customer** checks the menu.
* **Customer** selects items to order.
* **Staff** receives the order from the customer.
* **Staff** confirms order details with the customer.
* **Order** is sent to the kitchen (Chef).
* **Chef** prepares the food.
* **Staff** serves the food to the customer.

**Activity Diagram 2: Payment and Checkout Process**

**Description**: This diagram represents the steps involved in a customer completing their payment after finishing their meal.

**Key Activities:**

* **Customer** receives the bill.
* **Customer** chooses the payment method (cash, card, etc.).
* **Staff** confirms the payment details.
* **Customer** pays the bill.
* **Staff** confirms payment and generates receipt.
* **Transaction** is complete.

# 6 Class Diagram

A diagram of a function

Description automatically generated

**Associations**

* **Customer ↔ Reservation**: A customer can make multiple reservations, but a reservation belongs to one customer.
* **Order ↔ OrderItem**: An order contains multiple order items (Composition).
* **Order ↔ Payment**: An order can have one or more payments (Association).
* **Table ↔ Reservation**: A table can be reserved by multiple reservations, but a reservation is linked to one table (Aggregation).

# 7 Finite State Machine

**States**:

1. **Order Created**: The customer places an order.
2. **Order Confirmed**: Staff confirms the order details.
3. **Order Prepared**: Chef prepares the order.
4. **Order Served**: Staff serves the food to the customer.
5. **Order Completed**: Customer pays, and the transaction ends.

**Transitions**:

* *Create Order*: Moves the order to "Order Created."
* *Confirm Order*: Staff confirms the order, transitioning it to "Order Confirmed."
* *Prepare Food*: Chef finishes preparing the food, moving it to "Order Prepared."
* *Serve Food*: Staff serves the food, transitioning it to "Order Served."
* *Complete Payment*: Customer pays the bill, transitioning the order to "Order Completed."

# 8 User Manual

**1. Start the Application**

* When the system is launched, a login screen will appear.
* Options:
  + **Admin Login**: For restaurant managers or staff to access reports, inventory, and order management.
  + **Customer Login**: For customers to browse the menu, make reservations, or place orders.

**2. Customer Workflow**

* **Browse Menu**: After logging in, the customer can check available dishes and specials.
* **Place an Order**:
  + Select desired items and add them to the cart.
  + Confirm the order by clicking "Place Order."
* **Make Payment**:
  + Choose a payment method (cash, card, or digital wallet).
  + Complete the transaction and receive a receipt.

**3. Admin/Staff Workflow**

* **Inventory Management**: Admin can add, update, or remove inventory items.
* **Order Management**:
  + View active orders.
  + Confirm or update order statuses.
* **Generate Reports**:
  + Admin can generate and download daily/weekly/monthly reports.

**4. Exit the Application**

* After completing tasks, click the "Logout" button to return to the login screen.