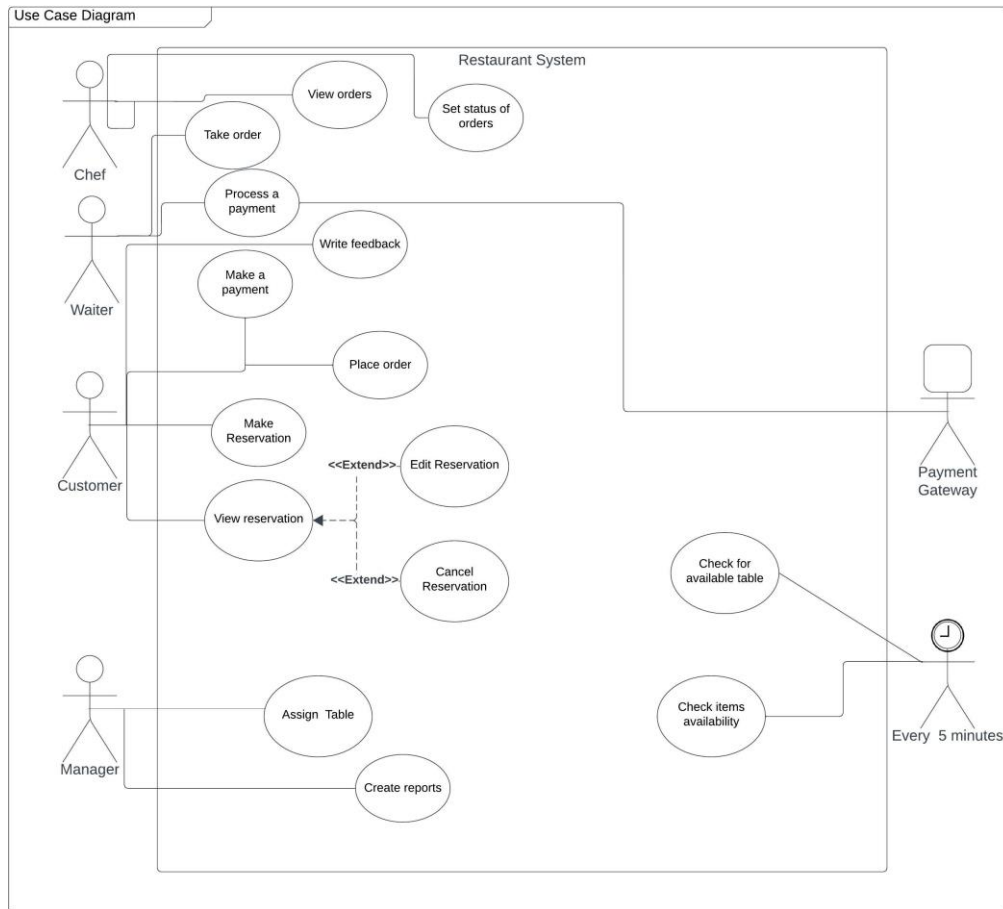# User Requirements

The Restaurant Management System(RMS) is designed to perform the operations of a restaurant efficiently. This system will be used by customers to make reservations, place orders, and provide feedback. Additionally, the system will be utilized by employees of the restaurant to assist customers, prepare orders, and manage the menu, inventory, and payments.

Customers are able to make reservations through the app by selecting a date, time, and number of guests. The system then checks for available tables and confirms the reservation, providing the customer with a reservation number. Customers can also edit or cancel their reservations if needed. For providing feedback, customers can rate their experience, leave comments on specific dishes, or describe their overall experience in the restaurant. This feedback is valuable for maintaining the quality of service. Customers can also place orders directly through the app, either while dining in or before arriving. They can browse the menu, select items, customize dishes, and add them to their order. Once finalized, the order is sent directly to the kitchen for preparation.

Employees, such as Managers, can assign tables if a customer walks into the restaurant without a prior reservation. The manager uses the system to see available tables in real-time and assigns them accordingly, ensuring that all reservations are accounted for and walk-ins are accommodated when possible. Waiters can take orders from customers dining in, using tablets or handheld devices to enter orders directly into the system, which automatically sends them to the kitchen for preparation. The waiters are also responsible for serving the food to customers once it's ready and ensuring all special requests are handled properly. Chefs can view the list of incoming orders in real-time through the system. The system organizes orders based on the time they were placed and the complexity of each dish, allowing chefs to prepare meals efficiently. The chefs can also mark orders as "in preparation," "ready to serve," or "completed," which updates the system and notifies the waiters.

Overall, the Restaurant Management System aims to streamline all operations, enhancing the customer experience and ensuring that the restaurant staff can manage their responsibilities effectively.

Use case diagram

Use Case Diagram

Restaurant System

Chef — View orders, Set status of orders, Take order

Waiter — Process a payment, Write feedback

Make a payment

Place order

Customer — Make Reservation

<<Extend>> Edit Reservation

View reservation

<<Extend>> Cancel Reservation

Check for available table

Check items availability

Manager — Assign Table, Create reports

Payment Gateway

Every 5 minutes

Use case Scenario: Make Reservation

Use Case Scenario: Make Reservation

Actor: Customer

Purpose and Context: A customer wants to reserve a table for a specific date and time.

Assumptions:
1. The customer has already decided on the date and time for the reservation.

Preconditions:
1. The system is operational and table availability is up-to-date.

Basic Flow of Events:

1.The customer accesses the "Make Reservation" option on the system.
2.The system prompts the customer to input the reservation date, time, and number of guests.
3. Customer inputs info.
4.The system checks table availability based on the provided details.
5.The system displays available tables.
6.The customer selects a table from the list of available options.
7.The customer confirms the reservation.
8.The system saves the reservation and provides a confirmation message, including reservation details.

Allternative Flow:

1. No Tables Available:
1.1 The system informs the customer that no tables are available for the selected time and date.
1.2 The system suggests alternative time/date for the reservation.
1.3 The customer choose suggested time/date and continue from point 5 of basic flow.

2. Incomplete Personal Details:
2.1 The system prompts the customer to fill in missing mandatory information.
2.2 The customer provides the missing details and continue.

3. Customer doesn't confirm the reservation:
3.1 The reservation is not made.
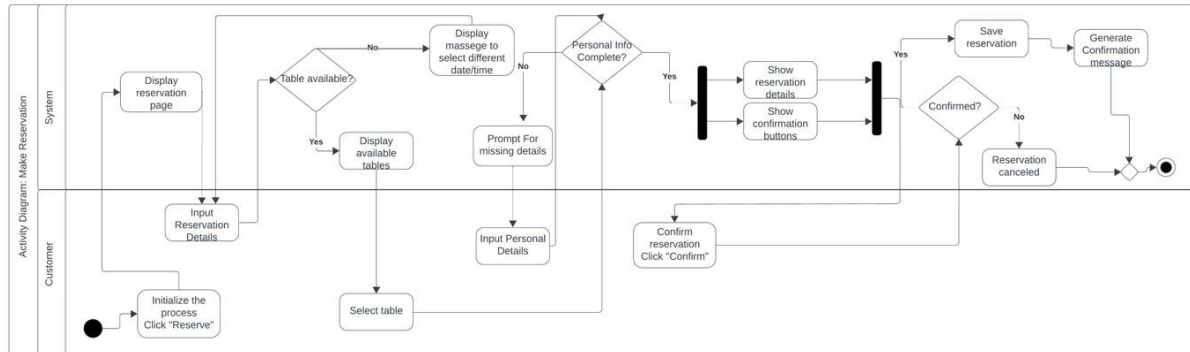
Postconditions:
Basic:
The reservation is made and saved in the system.
The customer recieves a confirmation
Alternative:
Reservation is not made.

Activity diagram: Make Reservation

The process of making a reservation in the restaurant system is simple and intuitive for the customer. When a customer decides they want to reserve a table, they open the restaurant app and click on the "Reserve" button. They are asked to provide details like the date and time they want to visit and the number of people coming. After inputting these details, the system automatically checks if there are available tables for the requested date and time.

If a table is available, it shows the customer a list of available options, and they can choose their preferred one. Sometimes, if a table isn't available at the selected time, the system suggests other possible times or dates that could work. Once the customer selects a table and fills out any necessary information (like personal details for identification), they proceed to confirm the reservation. After confirmation, the system generates a summary of the reservation, and the customer receives a confirmation message, either on their app or by email. If the customer decides not to confirm, the reservation is simply not made, and they can try again later.

Use case scenario: Place an order

Use Case Scenario: Place an Order

Actor: Customer

Purpose and Context: A customer wants to place order of meals and drinks

Assumptions:
1. The menu is up-to-date with current items and availability.

Preconditions:
1. The system is operational and menu is accessible.

Basic Flow of Events:

1.The customer accesses the "Place order" option on the system.
2. The system shows menu of available menu items.
3. The customer browses the menu.
4. The customer selects item to add to the order, specifying quantity and optionaly notes (e.g. , allergies).
5. The system adds the selected items to the order summary and calculates subtotal.
6. The customer reviews the order summary.
7. The customer confirms the order or correct it.
8. The order sends the order to the kitchen for preparation.

Allternative Flow:

1. **Incomplete Order Details:**
1.1 The system prompts the customer to fill in missing mandatory details (e.g., item quantity).
1.2 The customer provides the missing information and continues from point 5 of the basic flow.
2. **Customer Cancels Order:**
2.1 The customer exits the order process or cancels before confirming.
2.2 The order is not sent to the kitchen, and the table's order status remains unchanged.

Postconditions:
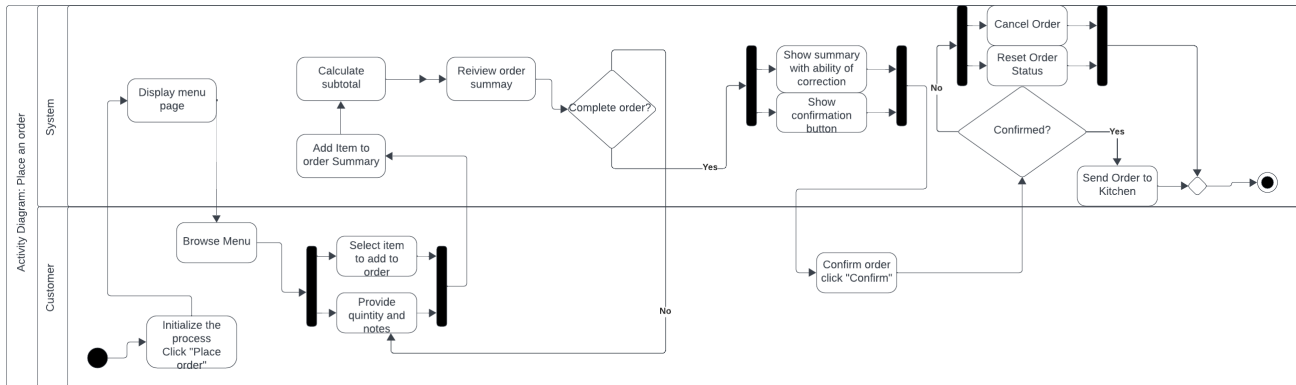**Basic:**
 • The order is successfully saved in the system and sent to the kitchen for preparation.
 • The customer receives an order confirmation.
Alternative:
 • The order is not placed, and the customer is prompted to try again.

Activity diagram: Place an Order

Placing an order at the restaurant using the system is designed to be just as seamless. When customers are ready to order, whether they are dining in or ordering ahead, they go to the "Place Order" section. The system presents them with the current menu, allowing them to browse through the available dishes.

The customer can click on any item they want to order. If they have specific preferences—like wanting a dish with less spice or avoiding certain ingredients—they can add notes before finalizing the item. Once they've selected an item, they provide the quantity they'd like and add it to the order summary. The system then calculates the subtotal if all the necessary iformation is provided the system continues to the next step. If not all the information was provided the system asks customer to correct the order.

After all items are chosen, the customer is shown an order summary. Here, they can check everything they've added, make corrections, or remove items. If everything looks good, they click "Confirm" to finalize the order, and the system sends the order to the kitchen for preparation. If at any point the customer changes their mind, they can choose to cancel the order, and it will simply not be processed. This process gives customers full control of their meal selection, ensuring they can tailor the experience to their taste and preferences, while also making it easy for the kitchen to receive clear instructions.

Use case scenario: Process Payment

Use Case Scenario: Process Payment

Actor: Waiter

Purpose and Context: A waiter needs to process the payment for a customer's order after they have finished dining.

Assumptions:
1. The customer has finished their meal and requested the bill.
2. The order has been fully prepared, served, and is recorded in the system.

Preconditions:
1. The system is operational and connected to the payment processing service.
2. The customer's order exists in the system and is marked as ready for payment.

Basic Flow of Events:
1. The waiter accesses the "Process Payment" option on the system.
2. The system displays the customer's order summary, including all items and the total amount due.
3. The waiter presents the bill to the customer.
4. The customer reviews the bill and provides a payment method (e.g., cash, credit card, mobile payment).
5. The waiter selects the payment method in the system.
6. The waiter inputs any necessary payment details (e.g., entering cash amount received or initiating card payment).
7. The system processes the payment transaction.
8. The system confirms that the payment was successful.
9. The system updates the order status to "Paid."
10. The system generates a receipt.
11. The waiter provides the receipt to the customer and thanks them.

Alternative Flow:
1. **Payment Declined**:
   - **1.1** The system indicates that the payment was declined or failed.
   - **1.2** The waiter informs the customer of the declined payment.
   - **1.3** The customer provides an alternative payment method.
   - **1.4** The waiter processes the new payment method, returning to step 5 of the basic flow.
2. **Partial Payment / Split Bill**:
   - **2.1** The customer requests to split the bill among multiple payment methods or guests.
   - **2.2** The waiter selects the "Split Payment" option in the system.
   - **2.3** The waiter enters the amount or items each person will pay for.
   - **2.4** The waiter processes each payment separately, repeating steps 5 to 8 for each.
   - **2.5** The system confirms that the total amount has been fully paid.
   - **2.6** The system generates individual receipts for each payment.
   - **2.7** The waiter hands out the receipts to the respective customers.
3. **Payment System Error**:
   - **3.1** The system encounters an error during payment processing (e.g., network failure).
   - **3.2** The system displays an error message to the waiter.
   - **3.3** The waiter informs the customer of the issue and apologizes.
   - **3.4** The waiter retries the transaction or, if unsuccessful, contacts a manager.
   - **3.5** The manager may choose to process the payment using an alternative method (e.g., manual imprint).
4. **Customer Disputes Bill**:
   - **4.1** The customer notices an error on the bill (e.g., incorrect item or amount).
   - **4.2** The customer informs the waiter about the discrepancy.
   - **4.3** The waiter reviews the order details and verifies the issue.
   - **4.4** The waiter corrects the order in the system or contacts a manager if necessary.
   - **4.5** The system updates the order summary and recalculates the total.
   - **4.6** The waiter presents the corrected bill to the customer.
   - **4.7** The customer proceeds with the payment, returning to step 4 of the basic flow.

Postconditions:
- **Basic**:
   - The payment is successfully recorded in the system.
   - The order status is updated to "Paid."
   - The customer's bill is settled, and they receive a receipt.
   - The system reflects accurate sales and inventory data.
- **Alternative**:
   - If payment is unsuccessful, the order remains marked as "Unpaid."
   - The system logs the failed payment attempt.
   - The waiter or manager takes appropriate actions to resolve the issue (e.g., offering alternative payment methods, correcting billing errors).
   - Any adjustments to the order are saved in the system for accurate record-keeping.

Exceptions:
- **System Crash During Transaction**:
   - The system crashes or freezes during the payment process.
   - The waiter restarts the system and verifies whether the payment was processed.
   - If uncertain, the waiter may contact the payment processor or manager before retrying to prevent double charging.
- **Insufficient Funds (Cash Payment)**:
   - The customer does not have enough cash to cover the bill.
   - The waiter politely informs the customer of the shortage.
   - The customer provides an alternative payment method or removes items if possible, subject to restaurant policy.

Activity diagram: Process Payment
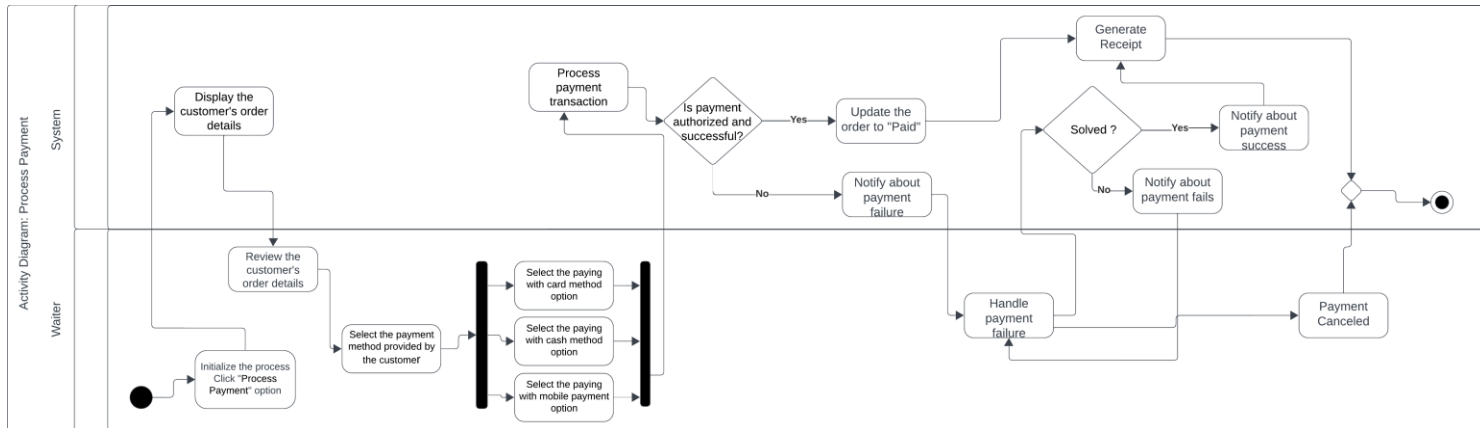
The **Process Payment** activity is designed to ensure a seamless and flexible payment experience for the customer. Once the waiter selects the "Process Payment" option, the system displays the customer's order details, allowing the waiter to confirm the order's accuracy before proceeding.

The waiter then chooses the payment method based on the customer's preference. Options include payment by card, cash, or mobile payment methods. The system processes the transaction using the chosen method, ensuring secure and efficient handling of the payment.

If the payment is authorized and successful, the system updates the order status to "Paid" and notifies the waiter about the successful payment. A receipt is generated for the customer as a record of the transaction.

In cases where the payment fails, the system immediately notifies the waiter of the issue. The waiter can attempt to resolve the problem, such as retrying the transaction or choosing an alternative payment method. If the issue cannot be resolved, the payment is canceled, and the waiter informs the customer accordingly.

This structured process ensures that the payment workflow is transparent, secure, and adaptable to various scenarios, minimizing errors and enhancing customer satisfaction.

Class Diagram - Analytical

Restaurant Management System - Class Diagram - Analytical

Note Colors:
Green = Association Examples
Yellow = Derived Calculations
Blue = Tagged Value/ Enum

Feedback
feedbackID
customer
rating
/averageRating
comments
date

Average rating is calculated from the average of all ratings.

< given by

Customer
customerID
name
contactNumber
email [0..1]

placed by >

< made by

Table
tableNumber
maxSeats
isReserved
location
seatingArrangement

assigned to >

Reservation
reservationID
customerId
table
reservationDateTime
numberOfGuests
status

reservationdDateTime

Qualified association

Payment
PaymentID
amount
date

quantity

Total amount is calculated from sum of all orderItems.

PaymentMethod
methodID
methodName
description

{BLIK, Cash, Credit Card, Apple Pay}

belongs to >

Order
orderID
table
/totalAmount
status
timestamp

< belongs to

Composition association

Association with attribute

OrderItem
orderItemID
notes

refers to >

< served by

{litres, kilograms, ...}

Inventory
inventoryID
lastUpdated

contains >

Composition association

Ingredient
IngredientID
Name:String
quantityInStock:int
unit:String
isPerishable:bool

< uses

< manages

Menu
menuID

Tax percentage calculated from multiplication of price and tax percentage.

MenuItem
itemID
name
description
price
taxPercentage = 1.2
/priceAfterTax
isAvailable
category
preparationTime
calories
discountPrice

supervises >

Reflex association

Staff
staffID
name
contactNumber

< employs

Restaurant
restaurantID
name
address
contactNumber
openingHours

has >

Aggregation association

Composition association

contains >

Aggregation association

{ disjoint, incomplete, dynamic }

Waiter
tipsCollected

Manager

Bartender

Chef
specialty

prepares >

{ complete, overlapping }

WaiterBartender
bonus

Analytical Class Diagram — Design Description

The analytical class diagram presented above visualizes the comprehensive structure of the restaurant management system. This system models the intricate operations of a restaurant, enabling smooth coordination between customers, staff, inventory, and services. The primary goal is to enhance operational efficiency, ensure customer satisfaction, and optimize resource management within the restaurant.

System Overview

- **Customers**: Customers interact with the system by making reservations, placing orders, making payments, and providing feedback. This interaction is pivotal in creating a personalized dining experience and allows the restaurant to tailor its services to meet customer preferences and expectations.

- **Staff Roles**: Staff members are hierarchically structured with specializations for positions like Chef, Bartender, Waiter, Manager, and WaiterBartender. Each role has specific responsibilities that contribute to the overall operations of the restaurant. This hierarchical structuring ensures clarity in duties and efficient workflow management.

- **Inventory and Ingredients**: The system tracks inventory levels and manages ingredients required for menu items. By connecting inventory and ingredients to menu items, the system ensures proper stock management, minimizes wastage, and supports kitchen operations. This linkage is crucial for maintaining the freshness of ingredients and ensuring that menu items are always available.

- **Tables and Seating Management**: Tables are assigned to orders and reservations to manage customer seating effectively. The system optimizes table allocation based on reservation schedules and walk-in customers, enhancing the dining experience by reducing wait times and maximizing occupancy.

Key Relationships and Interactions

- **Customers Linked to Orders and Feedback**: Customers are directly associated with their orders and feedback. This linkage creates a comprehensive customer interaction flow, allowing the restaurant to track customer preferences, order history, and satisfaction levels. It enables personalized service and targeted marketing efforts.

- **Orders Associated with Menu Items, Tables, and Payments**: Orders are central to the dining experience and are connected to several key components:

  - **Menu Items**: Each order contains one or more menu items selected by the customer. This association ensures accurate preparation and billing.
  - **Tables**: Orders are linked to specific tables, facilitating efficient service delivery by the wait staff.
  - **Payments**: Orders are associated with payments, ensuring that all transactions are accurately recorded and processed.

- **Inventory and Ingredients Connected to Menu Items**: Menu items are composed of various ingredients managed within the inventory system. This connection ensures that the availability of menu items is automatically updated based on inventory levels, preventing situations where customers order items that are out of stock.

**Staff Hierarchical Structure and Contributions**

- **Generalization of Staff Roles**: The staff hierarchy is represented using generalization in the UML diagram. The base class Staff is specialized into specific roles:

  - **Chef**: Responsible for preparing dishes, managing kitchen operations, and maintaining food quality standards.
  - **Bartender**: Specializes in preparing and serving beverages, managing the bar inventory, and creating

drink menus.

- o **Waiter**: Handles customer orders, serves food and drinks, provides customer service, and manages table assignments.
- o **Manager**: Oversees restaurant operations, manages staff schedules, handles administrative tasks, and ensures overall customer satisfaction.
- o **WaiterBartender**: A versatile role that combines the responsibilities of a waiter and a bartender, offering flexibility in staffing.

- **Supervision Relationships**: The diagram accounts for the supervision relationships between staff roles. For example, managers supervise chefs, bartenders, and waiters, coordinating their efforts to ensure seamless operations. Chefs may oversee kitchen assistants, while head waiters might coordinate the activities of junior waiters.

- **Specific Contributions**:

  - o **Chefs** contribute by crafting dishes, innovating menu items, and maintaining high culinary standards.
  - o **Bartenders** enhance the dining experience by providing a variety of beverages and engaging with customers at the bar.
  - o **Waiters** are the primary point of contact with customers, influencing their overall satisfaction through service quality.
  - o **Managers** ensure that all aspects of the restaurant function cohesively, from front-of-house service to back-of-house operations.

UML Constructs and Design Decisions

- **Generalization (Inheritance)**: The use of generalization groups staff members into specialized roles. This allows shared attributes and methods (like StaffID, Name, and ContactNumber) to be defined once in the base Staff class, promoting code reuse and consistency. Specialized classes inherit these attributes and can introduce role-specific properties and behaviors.

- **Aggregation and Composition**:

  - o **Aggregation**: Represents a whole-part relationship where the part can exist independently of the whole. For instance, a Menu aggregates multiple MenuItem objects. Menu items can exist independently and may be part of multiple menus (e.g., seasonal menus).
  - o **Composition**: A stronger form of aggregation indicating ownership and a lifecycle dependency between the whole and its parts. For example, an Order is composed of OrderItem objects. If the order is deleted, its order items cease to exist, reflecting a tight coupling.

- **Associations**:

  - o **Multiplicity**: Defines how many instances of a class can be associated with instances of another class. For example, a Customer can have multiple Reservations (one-to-many), but a Reservation is linked to a single Customer (many-to-one).
  - o **Bidirectional Associations**: Some associations are navigable in both directions. For instance, Orders and Tables are linked such that the system can identify which table an order is associated with and vice versa.

- **Notes and Explanations**: The diagram includes notes to clarify constructs and design decisions that are not immediately apparent. For example, the decision to model WaiterBartender as a subclass of Staff rather than combining Waiter and Bartender roles through multiple inheritance (which is not supported in some programming languages) is explained.

Design Diagram - Initial Version
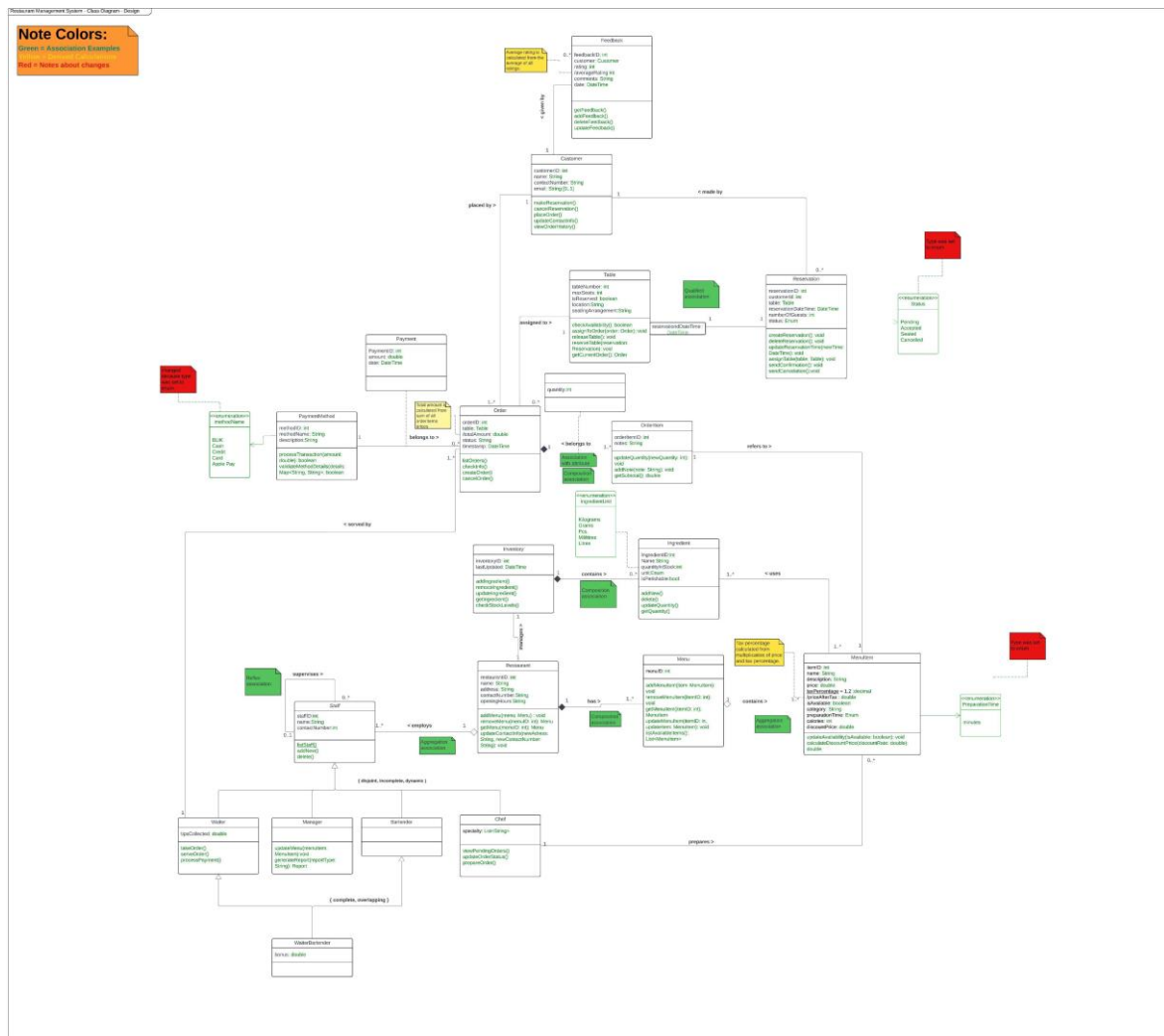
Design Decisions

The design class diagram illustrates the changes necessary to model UML constructs that do not exist in the C# language. The changes introduced are highlighted in green with some additional markups in yellow and red. Information in regards to qualified associations, composition associations, associations with attributes,overlapping, dynamic, disjoint, and abstract attributes, an so on will be described in the future.

Optional Attributes

Optional attributes are attributes that may contain no value. Optional attributes are simply C# properties that are marked as being nullable - e.g "?", which is to say they may contain null as a value. In the diagram they are marked as [0..1]

Multi-value attributes

Multi-value attributes are implemented as lists. Chefs specialty in the diagram is a multi-value attribute marked as a list of strings, to allow a chef workingin the restaurant to be responsible for more than one thing.

Derived attributes

Derived attributes utilize that get feature when declaring attributes in c#. In most instances showcased, the derived attributes would have the logics directly created on the getter and would return the final total. Ensuring an automatic creation of the derived attributes based on the other values that create the base of a derived attribute.
For example the totalAmount attribute in Order would be created as:
TotalAmount => orderItems.Sum(item => item.TotalPrice);

where orderItems is a list of all positions ordered.

Tagged value

Tagged values are implemented using the enumeration class method. Using enum types, each attribute will have their own separate class where the specific values are stored. Furthermore, due to the usage of enumerations the design class diagram includes the notations of those classes. (e.g Method Name) which transform the previously used note notation for tagged values to an actual class related to the given attribute.

Associations

- Reverse Association: Between MenuItem and Ingredient classes.
    - **From MenuItem to Ingredient**:
    We add the ingredient to the menu item's list of ingredients. Moreover, we also update the ingredient's list to include the menu item.

    - **From Ingredient to MenuItem**:

    When we remove an Ingredient from a MenuItem, you remove it from the menu item's list. In addition, we also update the ingredient's list to remove the menu item.

        - Shortly, it allows us to access related objects from either side of the relationship easily.
            1. We can find all ingredients used in a menu item.
            2. We can find all menu items that use a specific ingredient.

STATE DIAGRAM
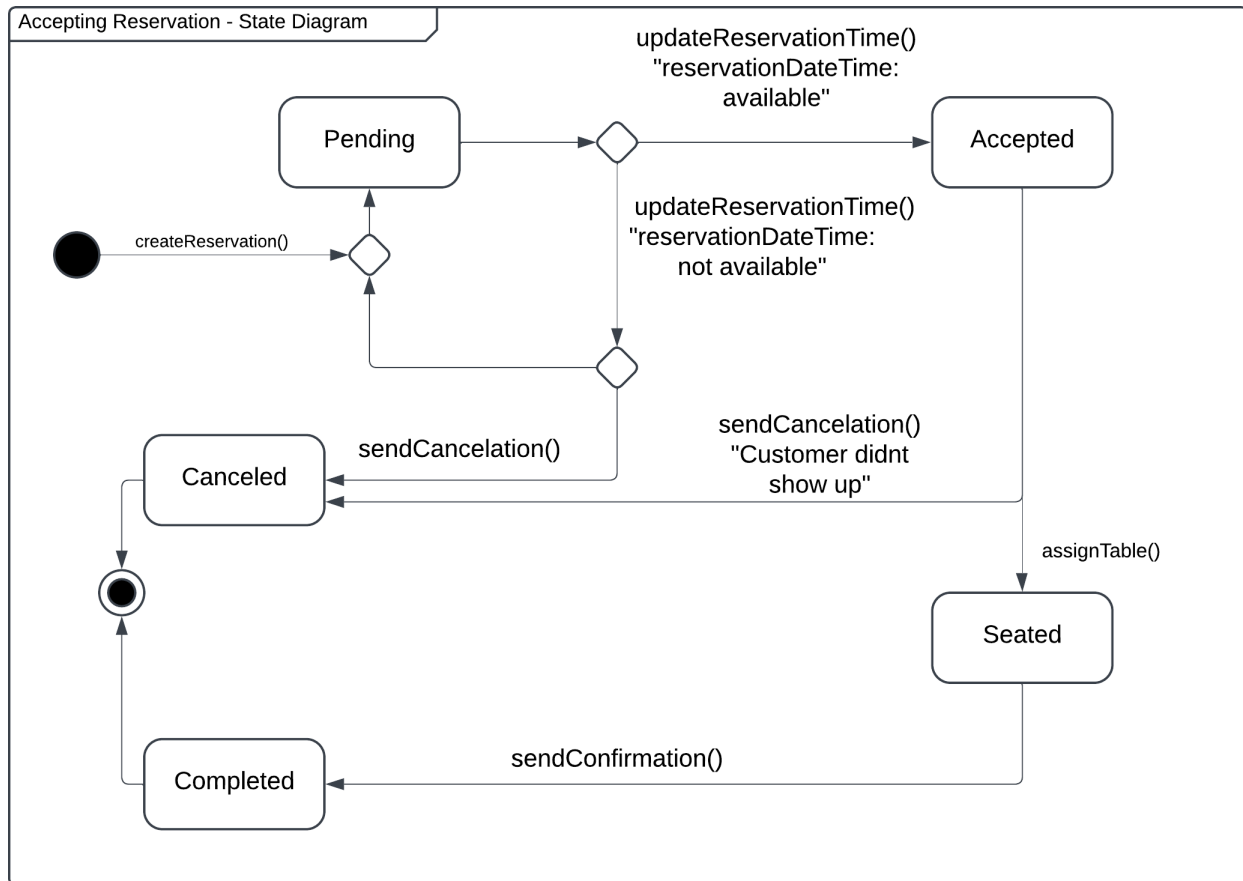
State Diagram - Design Description

The state diagram describes the state of accepting the reservation in the system. When using createReservervation() method, a new Reservation object is created with the status of "Pending". Depending whether there are dates/times available for the reservation and whether the client decides to change the date there are 3 possible states that the Object can be assigned to.

1."**Accepted**" - In case if there is suitable DateTime
2."**Cancelled**" - If there are no suitable dates
3."**Pending**" - If the client decides to reschedule the reservation

Once the reservation enters the accepted state, further modifications are done to it using either assignTable() method, if the client showed up, or sendCancelation(), if the client didn't show up for his scheduled time:
1."**Seated**" - If the client showed up
2."**Cancelled**" - If the client didn't show up
3."**Completed**" - By invoking sendConfirmation() method

The state of the object can end in two ways. Either as "Complete" or "Cancelled". If canceled the object should be removed thus the reservationDateTime tied to it, should be available again.