DOKUZ EYLÜL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME 2210

Object Oriented Analysis and Design

RESTAURANT MENANGMENT SYSTEM

by

2021510010 Çağrı AYDIN

CHAPTER ONE

INTRODUCTION

In a fast-food world where every customer seek for a quick service and best performance for their money a restourant menagment system is a crucial for restourants. Our restourant menagment system are designed to fulfill all needs for a restourant. From managing supplys to store al the information needed for a good restourant.

Table Management System

For our staff a good table management system is crucial for faster operations which is what we seek most. A better and readable inferface for staff willcut time off our service time and there will be no confision along the staff.

Ordering and Kitchen Communication

Smooth communication between the front-of-residence body of workers and kitchen teams is essential for making sure activate and correct order achievement. Our gadget helps efficient order-taking with customizable menus and modifiers, whilst also streamlining conversation among servers and kitchen team of workers to reduce errors and delays.

Menu Engineering and Analytics

Creating and optimizing menus is a crucial aspect of eating place control. Our device offers menu engineering tools to investigate income records, become aware of popular gadgets, and optimize menu services primarily based on purchaser choices and profitability. With actionable insights and analytics, restaurants could make knowledgeable selections to maximise revenue and purchaser satisfaction.

Staff Training and Performance Management

Investing in staff schooling and improvement is key to preserving high carrier requirements and worker satisfaction. Our system includes features for creating schooling modules, monitoring worker progress, and carrying out overall performance reviews. By

empowering body of workers with the necessary skills and know-how, eating places can deliver exceptional carrier and foster a subculture of continuous improvement.

Inventory Optimization and Supplier Management

Efficient inventory control and procurement methods are essential for controlling prices and minimizing waste. Our system automates inventory tracking, generates real-time reviews on stock degrees.

In end, our Comprehensive Restaurant Management System gives a holistic approach to restaurant operations, combining superior capabilities with consumer-friendly interfaces to supply unheard of performance and purchaser delight. Whether it is optimizing desk management, enhancing menu services, or engaging customers.

CHAPTER TWO

REQUIREMENTS

Specific Requirements

2.1 External Interfaces

None.

2.2 Functions

1. Table Management and Reservation: Allow hosts to manage table reservations, assign tables to guests, and track table availability.

- Method: manageTableReservation

- Parameters:

- reservationID (type: integer) - unique identifier for the reservation

- tableNumber (type: integer) - number of the table reserved

- reservationTime (type: datetime) - date and time of the reservation

- guestName (type: string) - name of the guest

- guestCount (type: integer) - number of guests in the reservation

- Return Value:

- success (type: boolean) - indicates whether the reservation was successfully managed

2. Ordering and Kitchen Communication: Facilitate order-taking, customization of orders, and communication between front-of-house staff and kitchen teams.

- Method: placeOrder

- Parameters:

- orderID (type: integer) - unique identifier for the order

- tableNumber (type: integer) - number of the table where the order was placed

- items (type: list) - list of ordered items, each item containing name (type: string), quantity (type: integer), and special requests (type: string)

- Return Value:

- orderStatus (type: string) - current status of the order (e.g., "received", "in preparation", "ready")

3. Menu Engineering and Analytics: Provide tools for menu creation, modification, and analysis of sales data to optimize menu offerings.

- Method: analyzeSalesData

- Parameters:

- startDate (type: datetime) - start date for analyzing sales data

- endDate (type: datetime) - end date for analyzing sales data

- Return Value:

- menuOptimizationSuggestions (type: list) - list of suggestions for optimizing the menu based on sales data analysis

4. Staff Training and Performance Management: Enable the creation of training modules, tracking of employee progress, and conducting performance evaluations.

- Method: evaluateEmployeePerformance

- Parameters:

- employeeID (type: integer) - unique identifier for the employee

- performanceMetrics (type: dictionary) - dictionary containing performance metrics such as punctuality (type: float), customer satisfaction ratings (type: float), and sales performance (type: float)

- Return Value:

- performanceScore (type: float) - overall performance score calculated based on provided metrics

5. Customer Engagement and Loyalty Programs: Implement loyalty programs, personalized promotions, and targeted marketing campaigns to engage customers and drive repeat business.

- Method: enrollInLoyaltyProgram

- Parameters:

- customerID (type: integer) - unique identifier for the customer

- loyaltyPointsEarned (type: integer) - number of loyalty points earned by the customer

- Return Value:

- enrollmentStatus (type: boolean) - indicates whether the customer was successfully enrolled in the loyalty program

6. Inventory Optimization and Supplier Management: Automate inventory tracking, generate real-time reports on stock levels, and streamline supplier management processes.

- Method: updateStockLevels

- Parameters:

- itemID (type: integer) - unique identifier for the item in inventory

- quantityAdded (type: integer) - quantity of items added to the inventory

- Return Value:

- inventoryStatus (type: string) - current status of the inventory after updating stock levels

7. Tableside Payment and Feedback: Support tableside payment processing and real-time feedback mechanisms to enhance the dining experience.

- Method: processTablesidePayment

- Parameters:

- tableNumber (type: integer) - number of the table where payment is being processed

- totalAmount (type: float) - total amount to be paid by the customer

- Return Value:

- paymentStatus (type: string) - status of the payment transaction (e.g., "successful", "pending", "declined")

8. Marketing Integration and Campaign Management: Integrate with social media channels, email marketing tools, and CRM software to streamline marketing efforts and analyze customer data.

- Method: createTargetedCampaign

- Parameters:

- campaignID (type: integer) - unique identifier for the marketing campaign

- targetAudience (type: list) - list of target audience segments based on demographics, past behavior, or preferences

- Return Value:

- campaignCreationStatus (type: boolean) - indicates whether the targeted campaign was successfully created

2.3 Performance Requirements

1. Response Time: The system shall respond to user actions within 5 seconds under normal load conditions.

2. Scalability: The system shall support a minimum of 30 concurrent users without degradation in performance.

3. Availability: The system shall have an uptime of at least 90% to ensure continuous availability for users.

2.4 Logical Database/File System Requirements

1. Relational Database: The system shall use a relational database management system (RDBMS) to store data related to orders, reservations, menus, and customer profiles.

2. File Storage: The system shall utilize file storage for storing multimedia content such as images and videos related to menu items and promotional materials.

2.5 Design Constraints

1. Compatibility: The system shall be compatible with all Windows Computers .

2.6 Software System Quality Attributes

1. Reliability: The system shall be highly reliable, with minimal downtime and error-free operation.

2. Usability: The system shall be intuitive and easy to use, requiring minimal training for restaurant staff and customers.

3. Performance: The system shall be responsive and performant, even under high load conditions, to ensure a smooth user experience.

4. Scalability: The system shall be scalable to accommodate growth in the number of users, transactions, and data volume over time.

2.7 Object Oriented Models

- 2.7.1 Analysis Class Model (Static Model)

The system shall include classes representing entities such as tables, orders, menu items, employees, and customers, along with their attributes and relationships.

- 2.7.2 Analysis Collaborations (Dynamic Model)

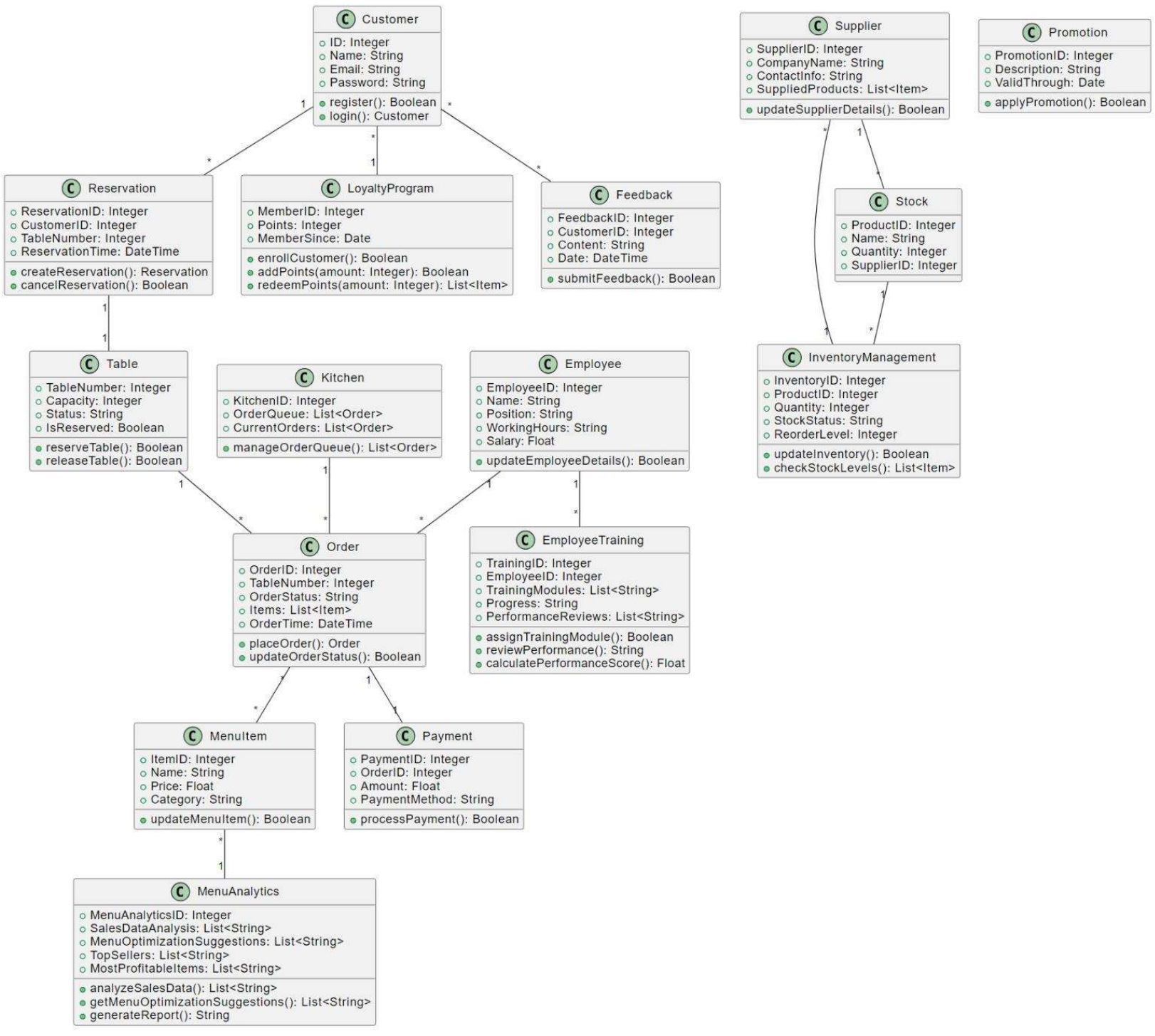
The system shall model interactions between classes to represent the flow of operations such as order placement, payment processing, and inventory management.

CHAPTER THREE

UML DIAGRAMS

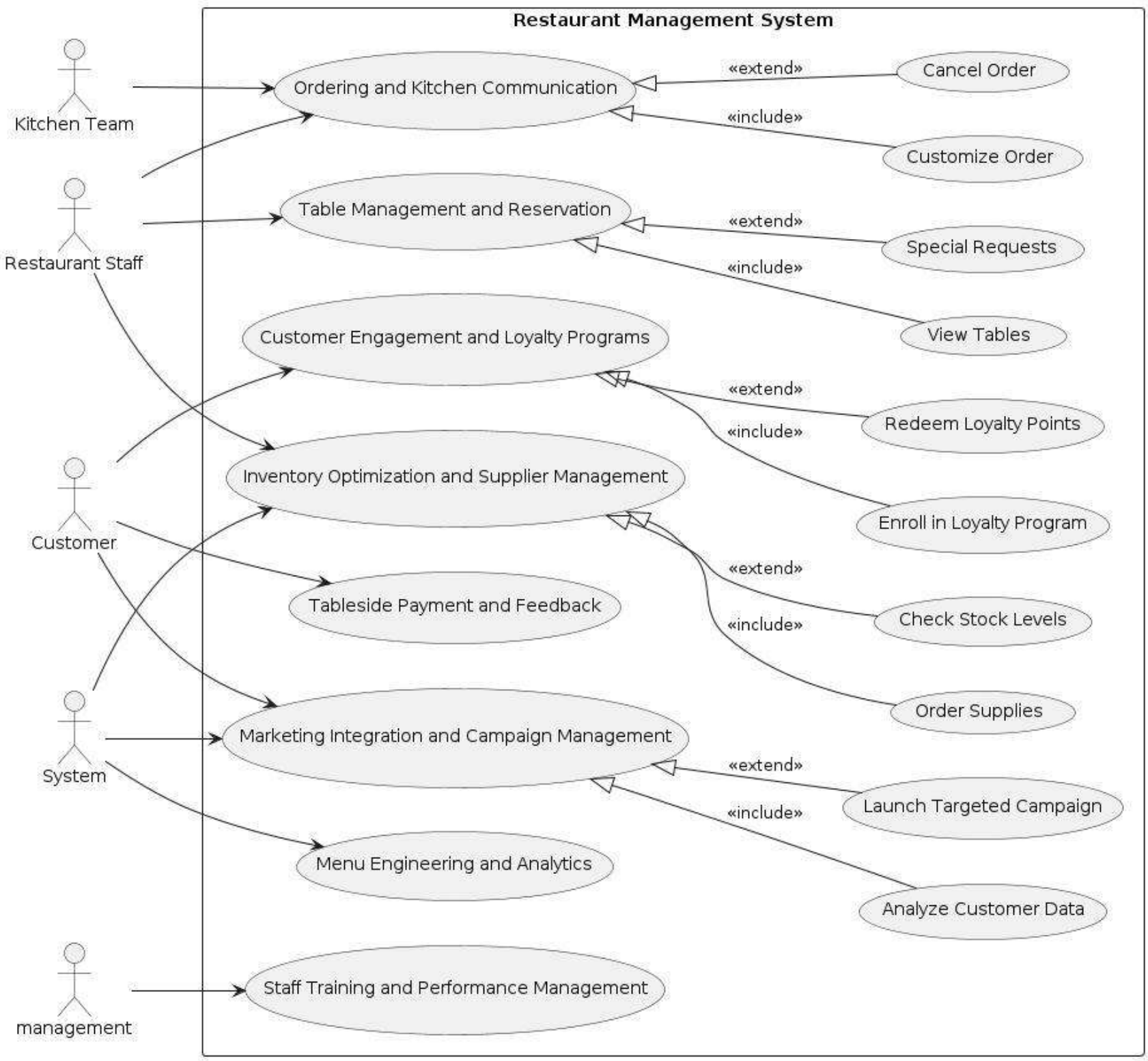
CLASS DIAGRAM

This is the class structure representing relations between classes like Customer, Reservation, Table, and Employee with respect to a restaurant management software. Some of the important operations are managing the reservation and the loyalty program, and processing orders with respect to the inventory system. Class attributes and operations of these classes help to manage restaurant activities effectively, starting from table booking, stock control, to employee management. These classes help to show how data in the system is organized and manipulated for the smooth operation of the restaurant.



USE CASE DIAGRAMS

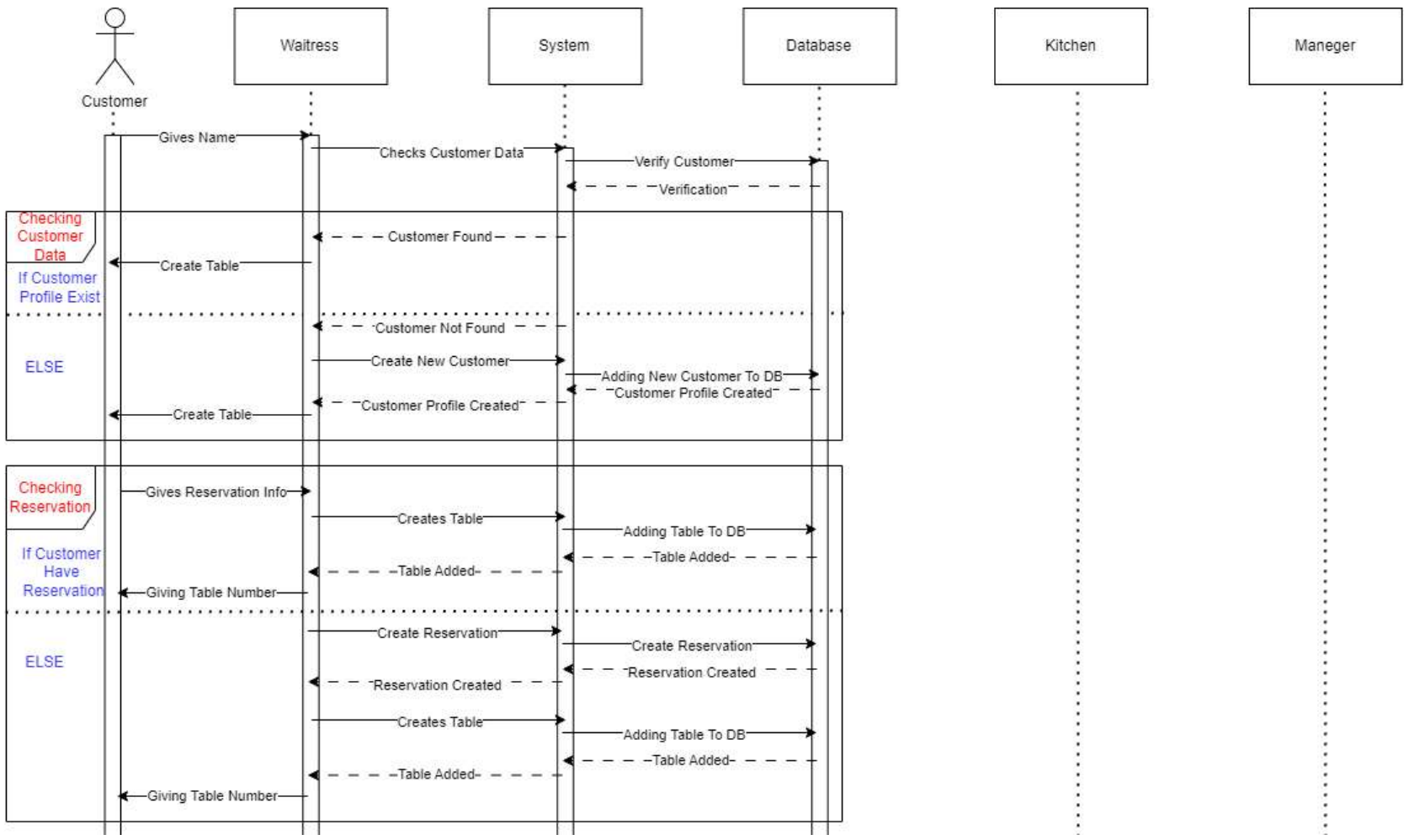
This is a complex diagram of different functionalities in a restaurant management system. It gives details of the different user touchpoints in the system, such as the kitchen team dealing with orders and communications, staff taking bookings and dealing with customers, and administration dealing with stocks and marketing information. It supports functionalities that will help to enhance customer service, such as loyalty programs, feedback mechanisms, all to ensure the smoothness in operations and, most importantly, improved customer experience.



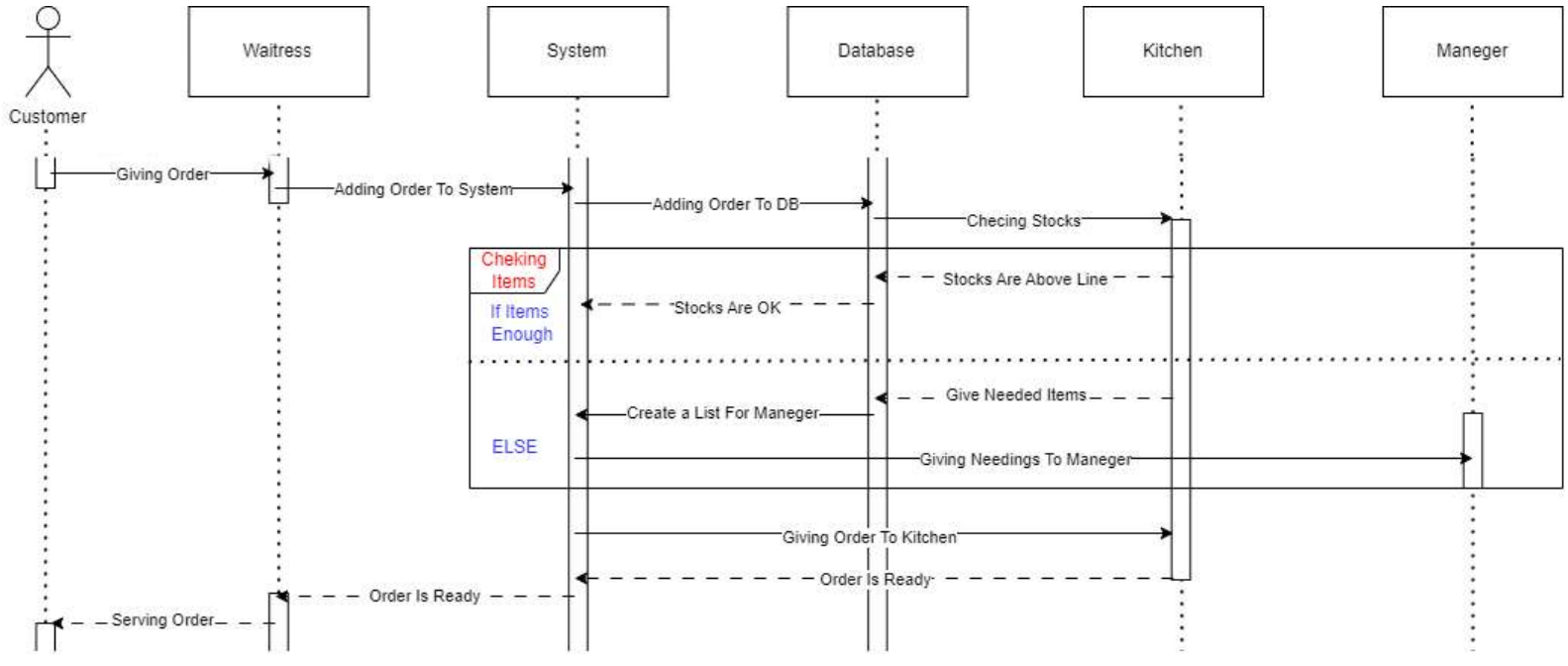
SEQUENCE DIAGRAMS

These three diagrams represent the complex interactions in the restaurant business process, especially the customer care process, order processing, closing activities and customer arrivals and table management that generates customer information true and check the saved order so that the tables are allocated accordingly. The second figure shows the order control system where orders are received, stock availability is checked and repaired, establishing the interaction between waitress, system and kitchen in managing inventory stock and filling orders is confirmed Finally, this final diagram supports billing and claims, including payment processing, claims. It also allows tables in the system to be locked. Together, these diagrams detail the processes required to improve customer service, data management and operational efficiency in a restaurant environment.

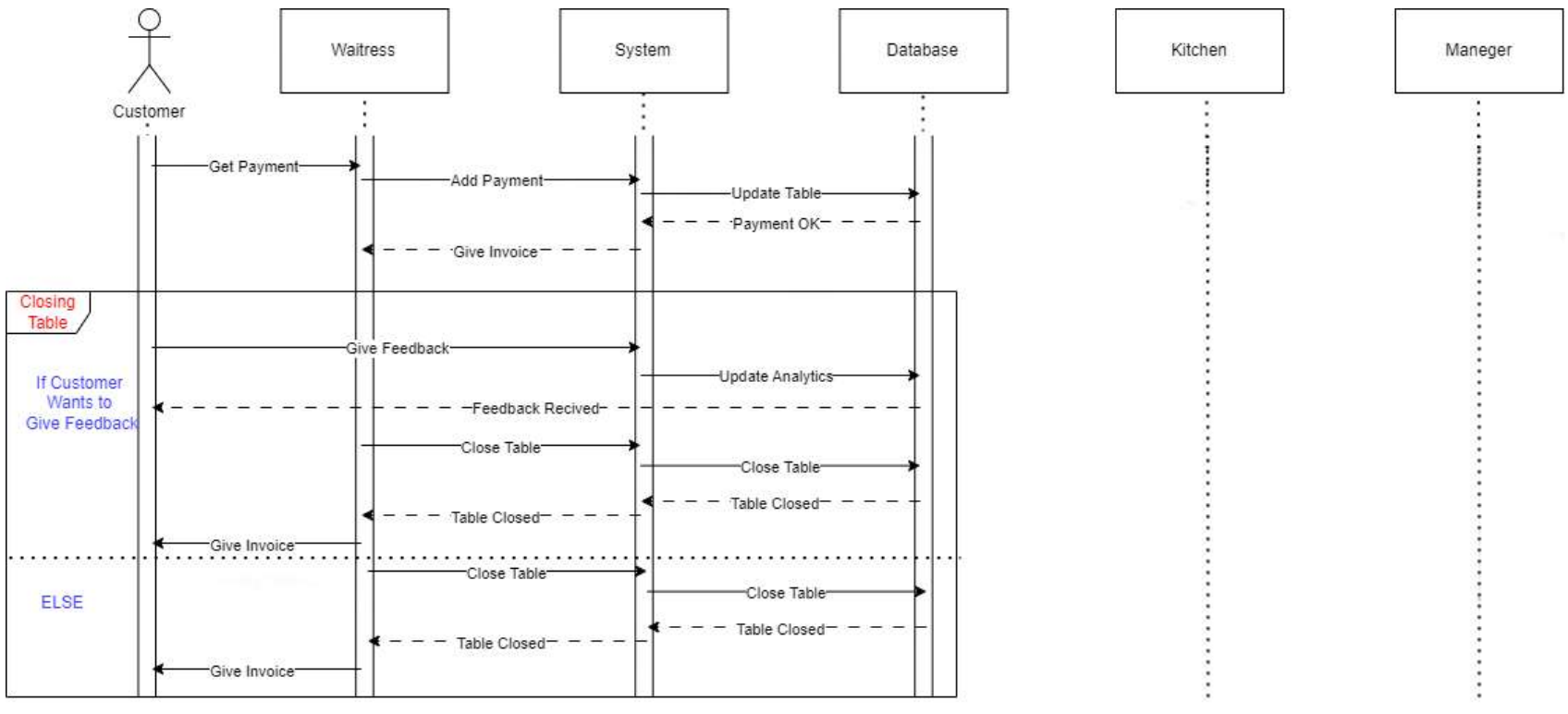
Example 1 :



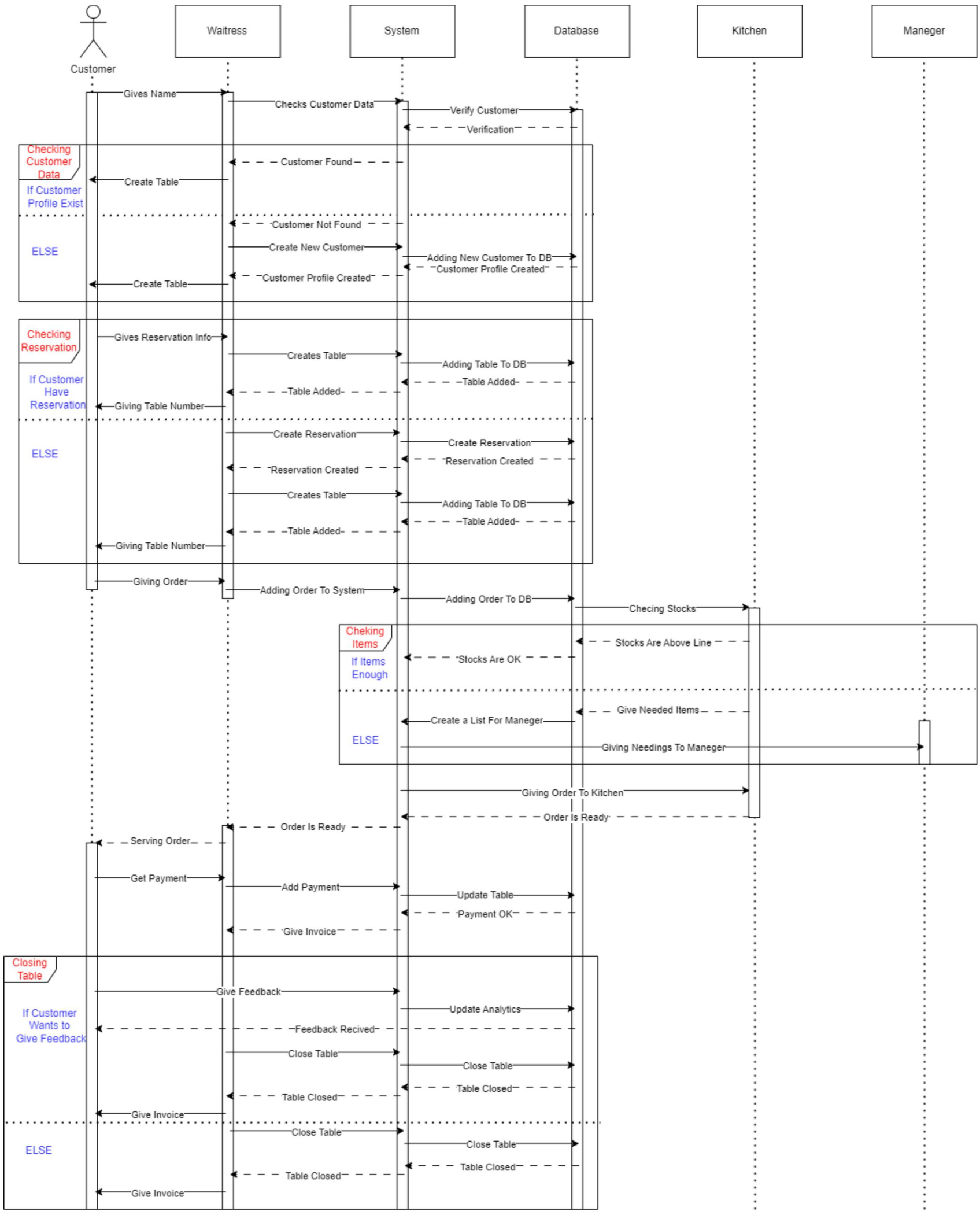
Example 2:



Example 3:



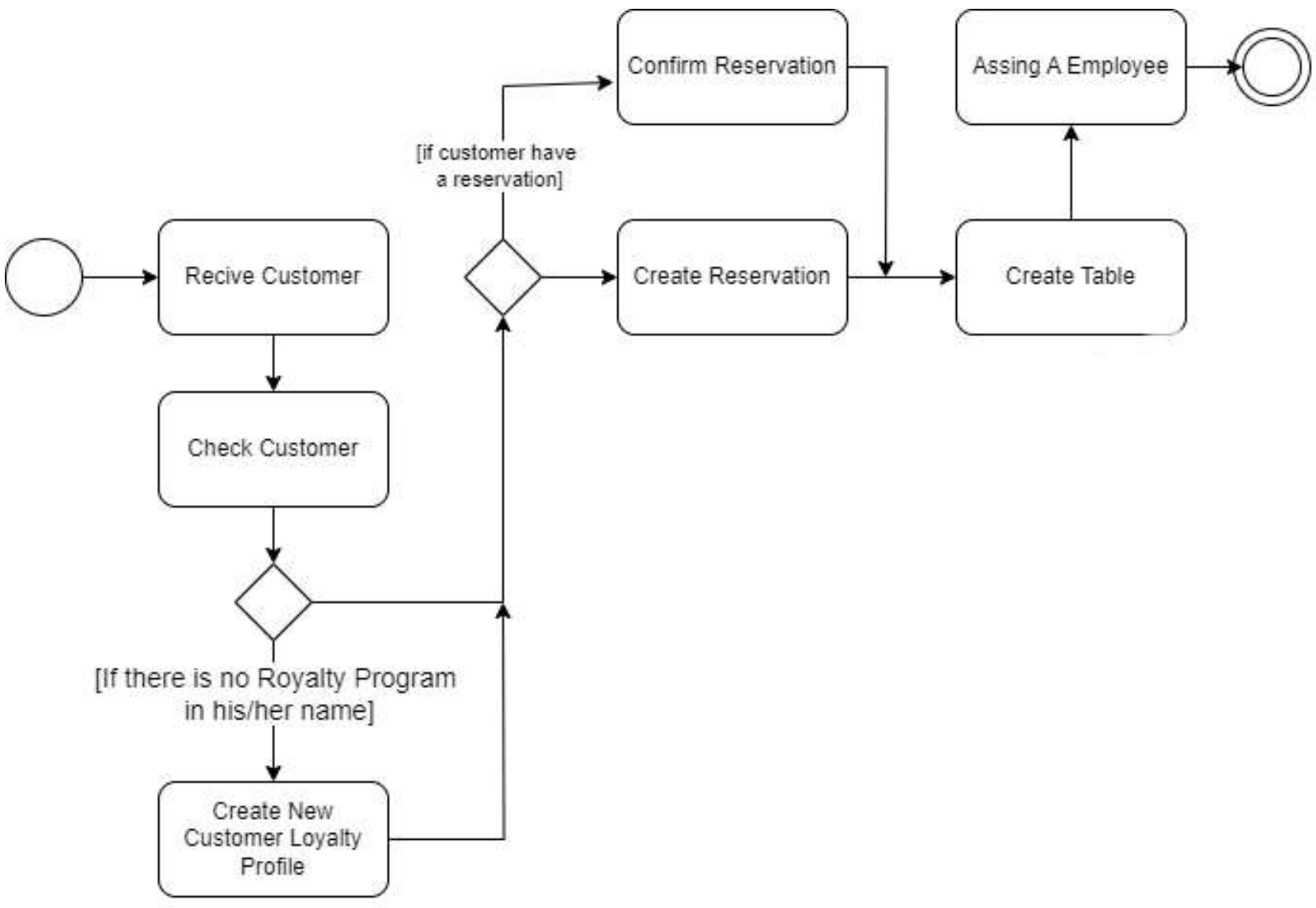
All Wiew On Sequence Diagram


Customer
Waitress
System
Database
Kitchen
Maneger
Gives Name
Checks Customer Data
Verify Customer
Verification
Checking Customer Data
Customer Found
Create Table
If Customer Profile Exist
Customer Not Found
ELSE
Create New Customer
Adding New Customer To DB
Customer Profile Created
Customer Profile Created
Create Table
Checking Reservation
Gives Reservation Info
Creates Table
Adding Table To DB
Table Added
If Customer Have Reservation
Giving Table Number
Create Reservation
Create Reservation
ELSE
Reservation Created
Reservation Created
Creates Table
Adding Table To DB
Table Added
Table Added
Giving Table Number
Giving Order
Adding Order To System
Adding Order To DB
Checing Stocks
Cheking Items
Stocks Are Above Line
Stocks Are OK
If Items Enough
Give Needed Items
Create a List For Maneger
ELSE
Giving Needings To Maneger
Giving Order To Kitchen
Order Is Ready
Order Is Ready
Serving Order
Get Payment
Add Payment
Update Table
Payment OK
Give Invoice
Closing Table
Give Feedback
Update Analytics
If Customer Wants to Give Feedback
Feedback Recived
Close Table
Close Table
Table Closed
Table Closed
Give Invoice
Close Table
Close Table
ELSE
Table Closed
Give Invoice

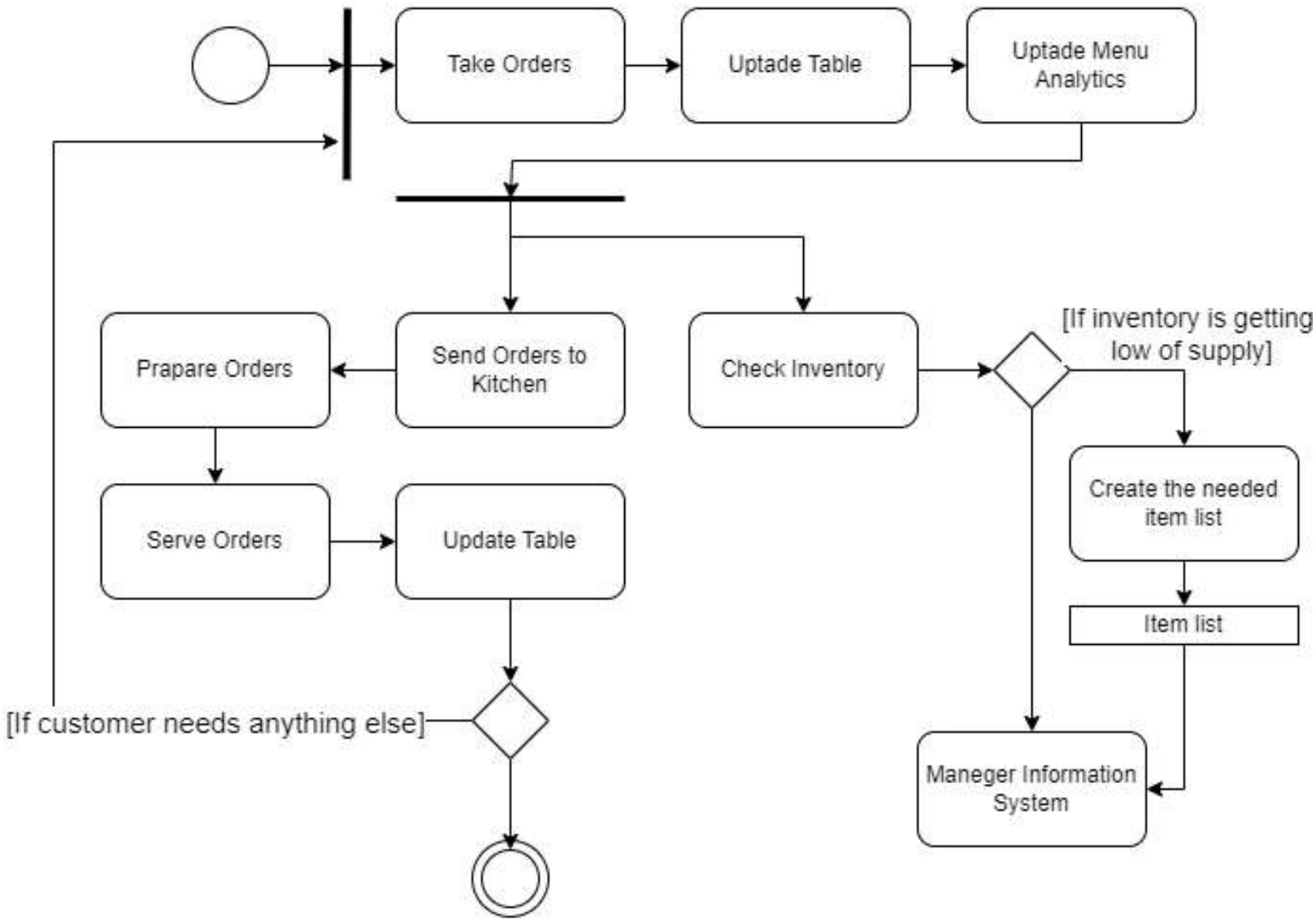
ACTIVITY DIAGRAMS

The three activity diagrams are a comprehensive overview of the operational process involved in managing the interaction of the restaurant customers. The first captures how customers are welcomed in, their bookings and loyalty profiles are checked, and may be created, and making every customer feel recognized and valued upon arrival at the restaurant. The second diagram captures the kitchen, the very heart of the restaurant, where orders are taken, prepared, and served quickly and with care, and where the smart management system keeps a close eye on inventory and menu trends. The third diagram captures the end-to-end dining experience—where, for each interaction with a customer, there is gathering of quite valuable feedback, processing invoices efficiently, making payments, and updating staff performance records as well as loyalty program features, to ensure that every encounter with a customer is satisfying as possible.

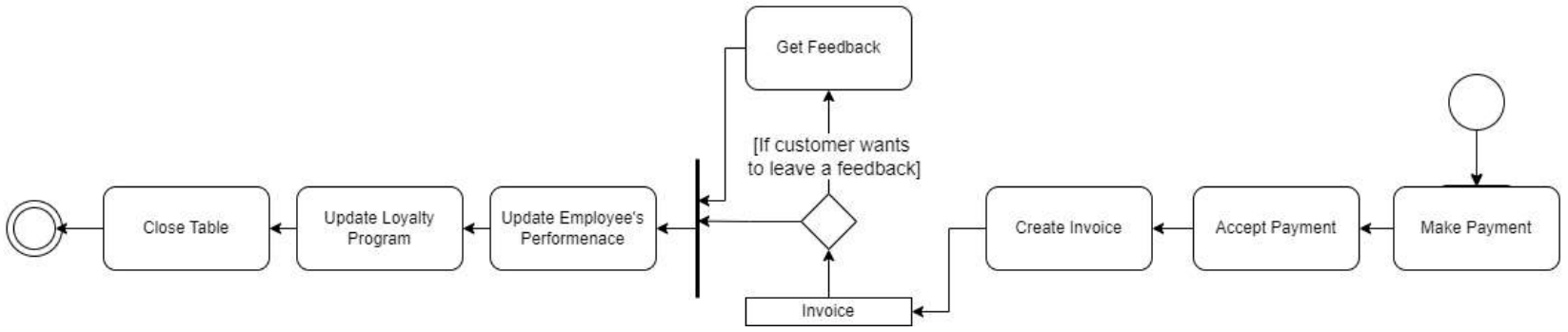
Example 1:



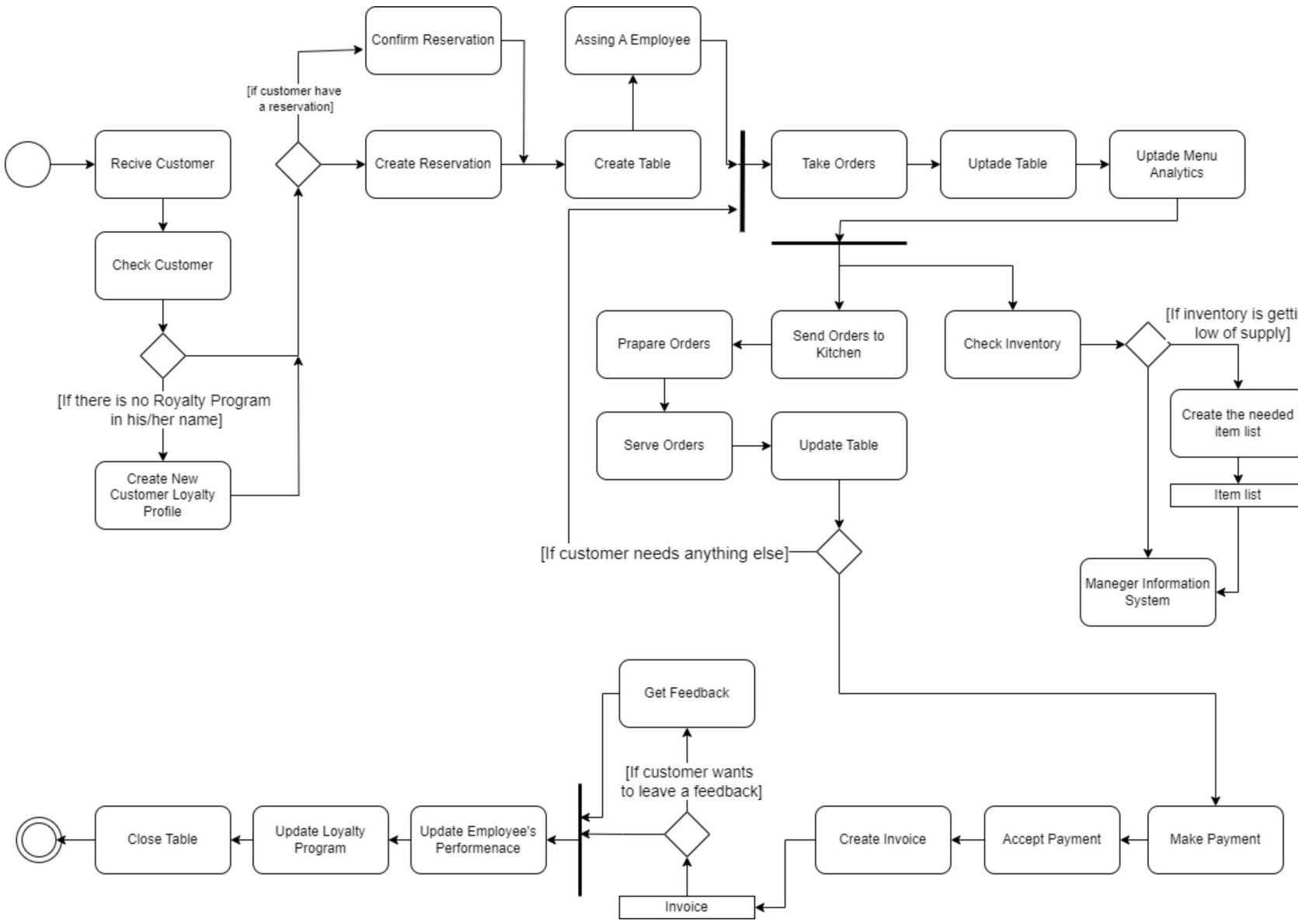
Example 2:



Example 3:



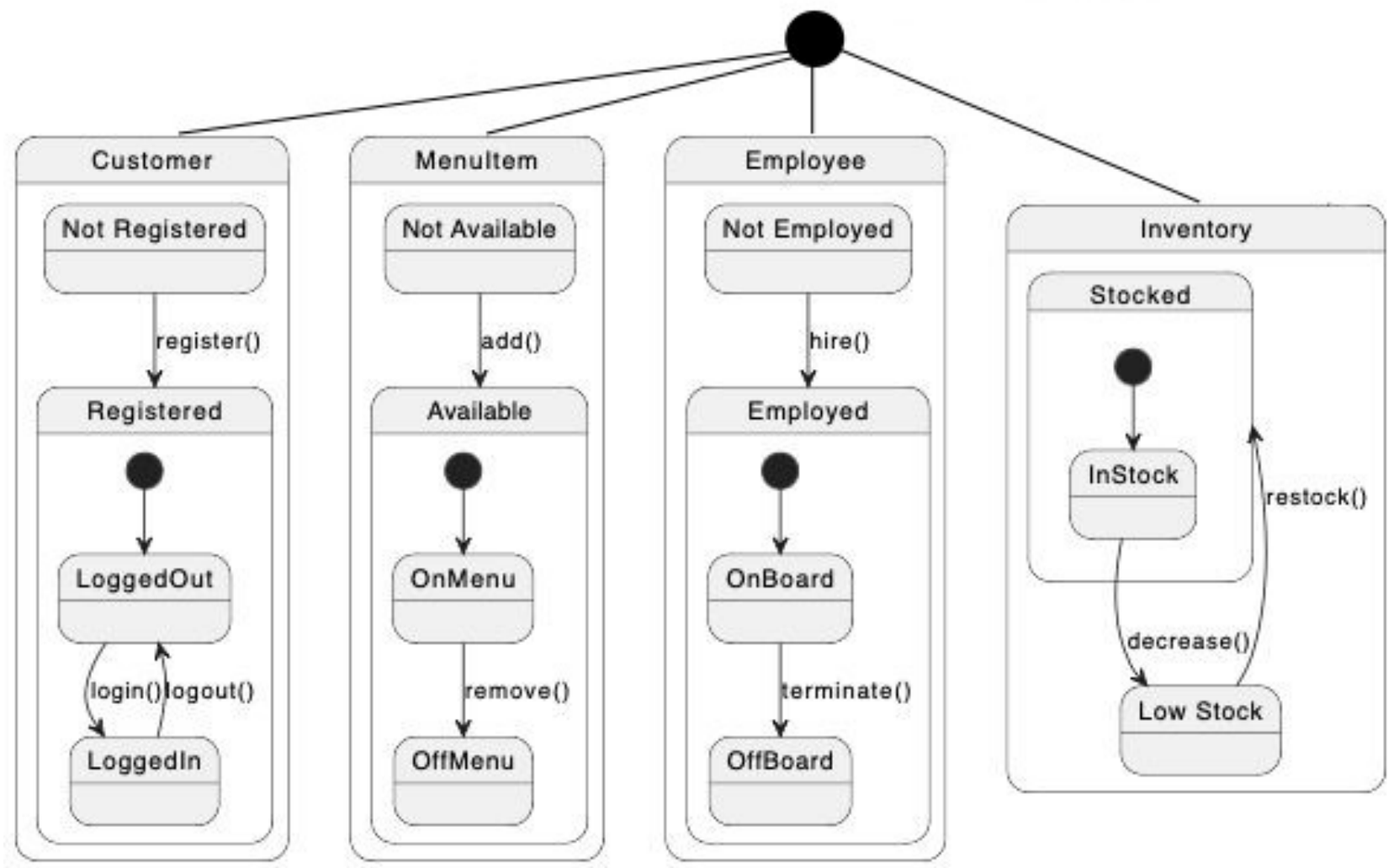
All Wiew On Activity Diagram



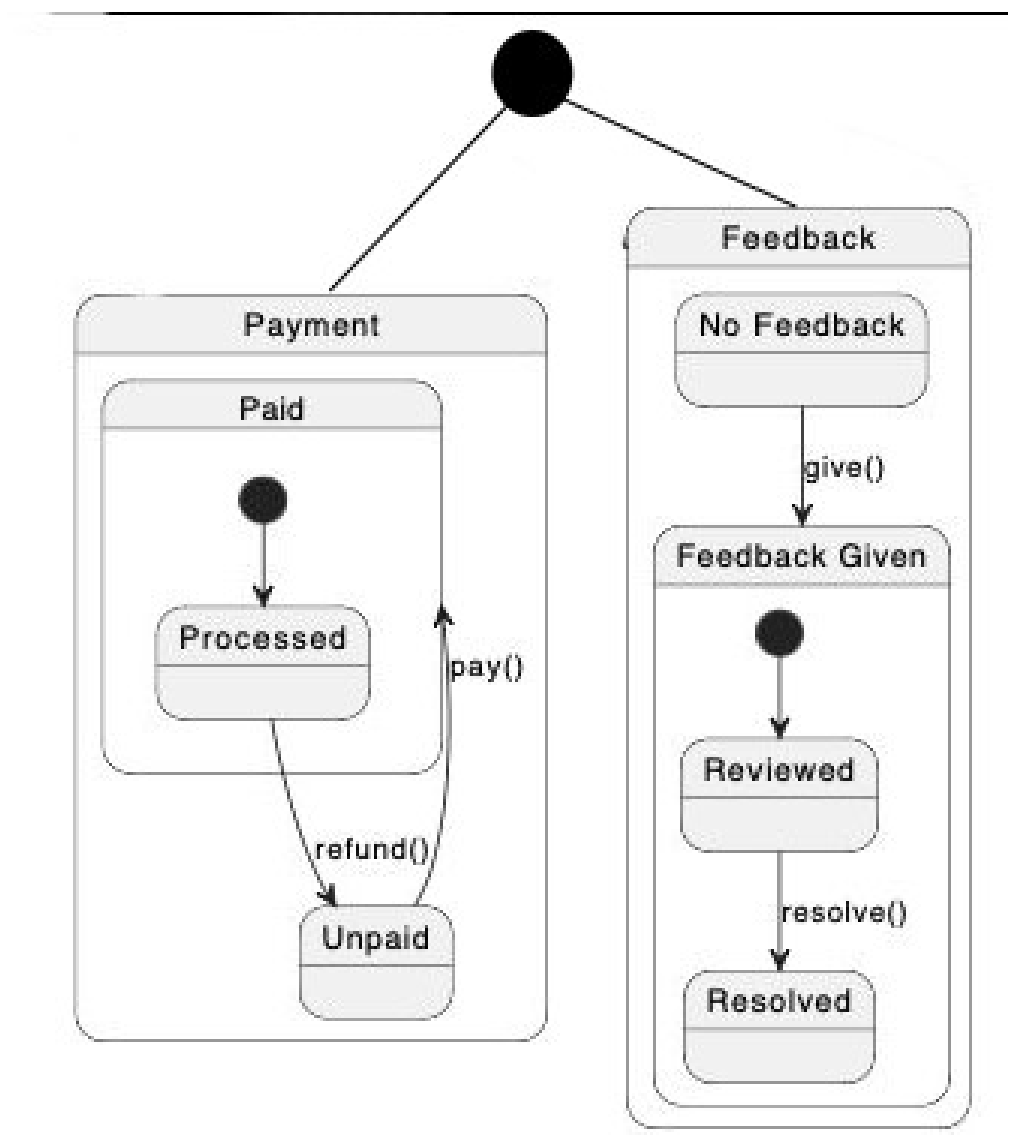
STATE DIAGRAMS

The presented state layout diagrams illustrate concepts within the restaurant location management device, providing a pure social order of customers, employees, food items, inventory, billing, comments , orders, and a clean description of inventory means To be on the menu, for personnel who are no longer provided on board or onboard, and for stock less than stock inventory, which means as each company state evolves in responding specifically to movement. The second account focuses on economic explanatory mechanisms, which show how payments evolve from unpaid to processed and the response is no longer paid and proceeds to an unambiguous decision. Figure 3 shows the sequence from installation to loading and storage from arrival to cancellation or crowning, together these diagrams encapsulate the operation of a restaurant internal flows are discussed, with systematic oversight and a variety of operational elements that are essential for effective systems and customer pride.

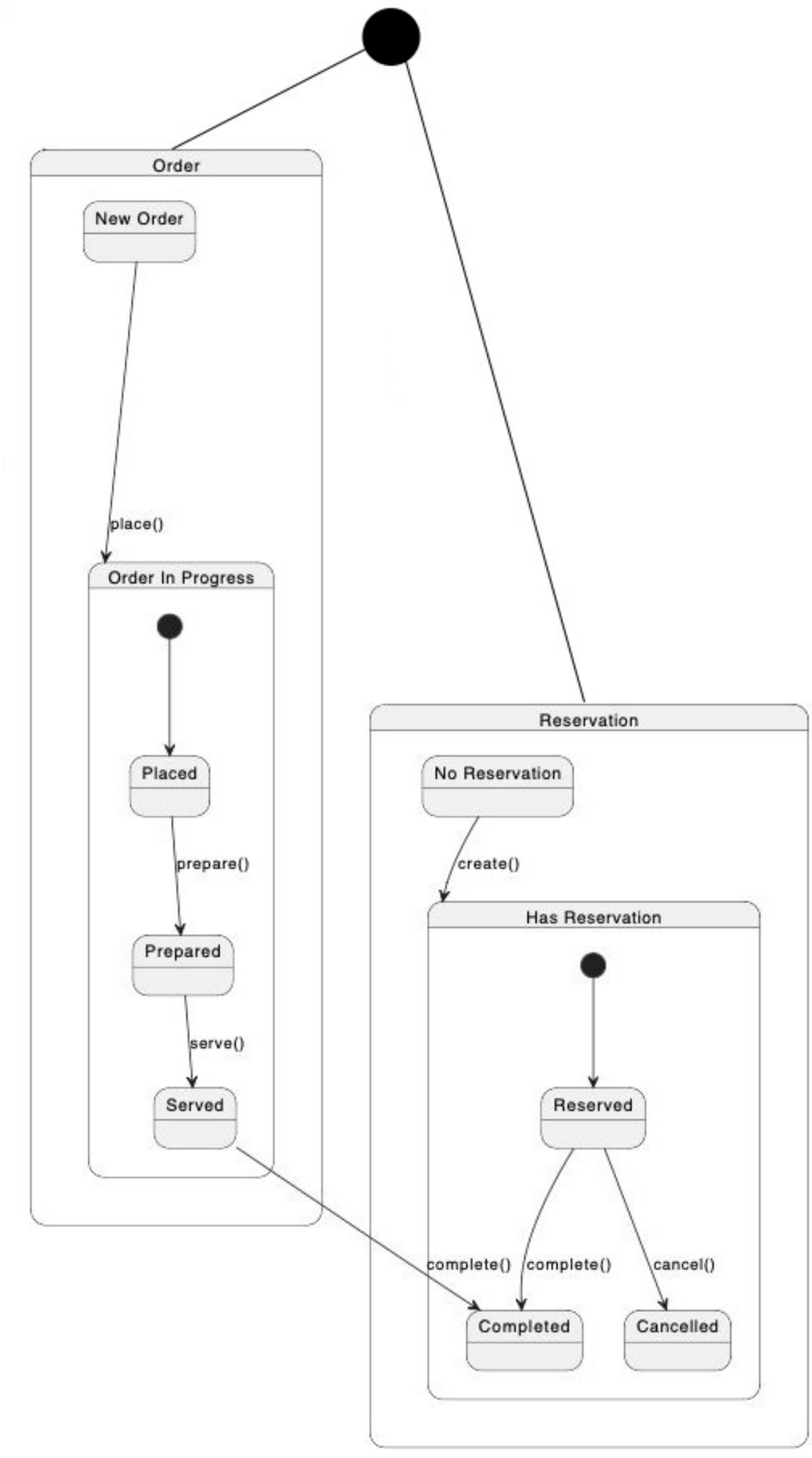
Example 1:



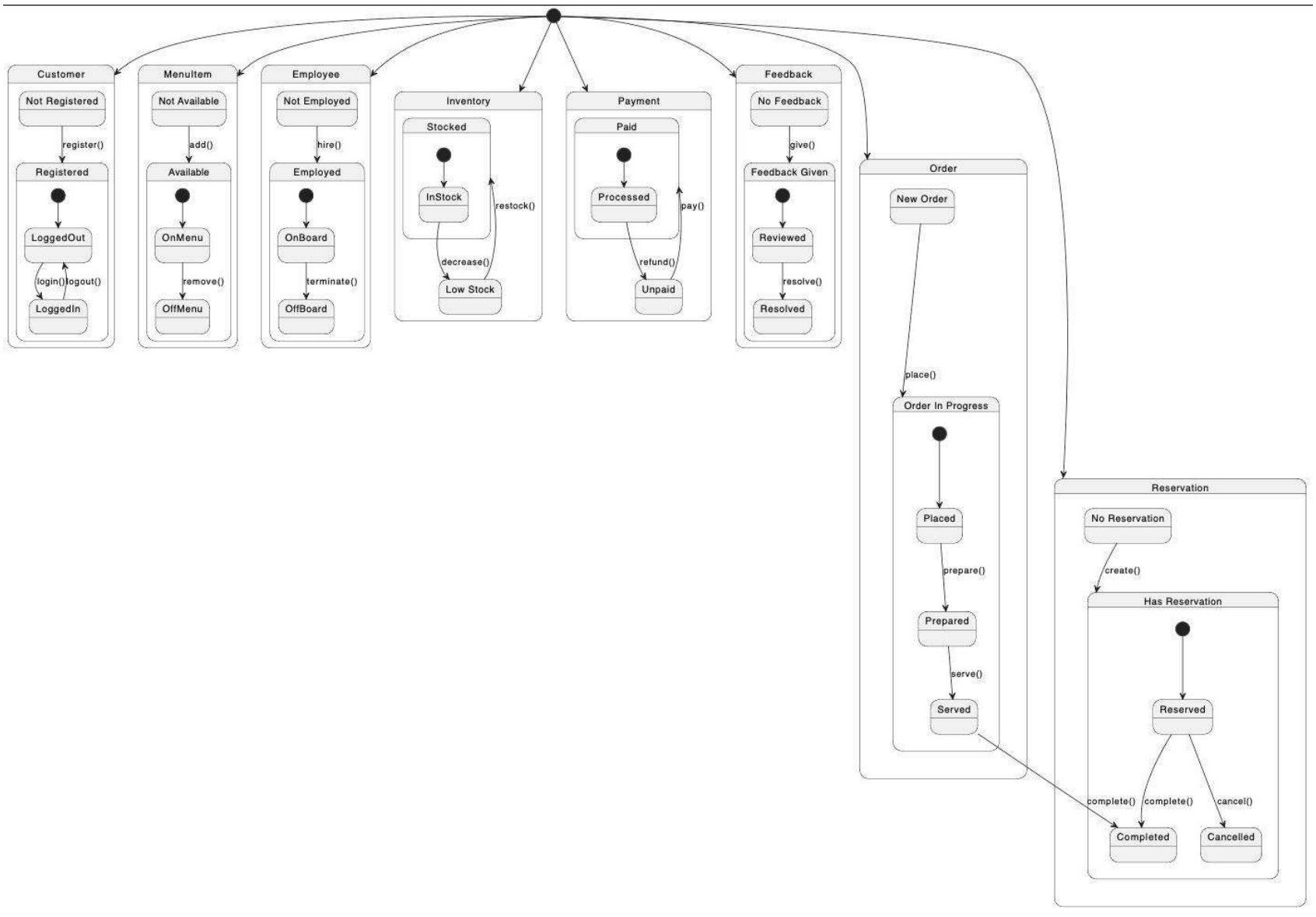
Example 2:



Example 3:


Order
New Order
place()
Order In Progress
Placed
prepare()
Prepared
serve()
Served
Reservation
No Reservation
create()
Has Reservation
Reserved
complete()
complete()
cancel()
Completed
Cancelled

All Wiew On State Diagram



CAHPTER FOUR

Main Components

1. Order Class: Represents an order in the restaurant.

2. OrderAndKitchenCommunication Class: Manages the GUI and interaction for order and kitchen communication.

Order Class

The Order class encapsulates the details of an order:

-Attributes:

- nextId: A static counter to assign unique IDs to each order.

- orderId: The ID of the order.

- tableNumber: The table number where the order is placed.

- itemName: The name of the item ordered.

- itemPrice: The price of the item.

- quantity: The quantity of the item ordered.

- **specialRequests:** Any special requests associated with the order.

- **Constructor:**

- **Initializes the attributes and assigns a unique ID to each order.**

- **Methods:**

- **Getters for all attributes.**

- **getTotalPrice():** Returns the total price of the order (price quantity).

- **toString():** Returns a string representation of the order.

OrderAndKitchenCommunication Class

The OrderAndKitchenCommunication class extends JFrame and manages the GUI:

- **Attributes:**

- **mainApp:** A reference to the main application.

- **orders:** A list to store orders.

- **textAreaOrders:** A text area to display orders**.**

- **textFieldTableNumber:** A text field to input the table number.

- **panelMenu:** A panel to display menu items.

- Static Block:

- **menuItems:** A static map containing menu items and their prices.

- Constructor:

- Initializes the attributes.

- Calls initializeUI() to set up the GUI components.

- initializeUI():

- Sets the frame title, size, and default close operation.

- Sets the layout to BorderLayout.

- Creates and adds the top panel for table number input.

- Creates and adds the center panel for menu items.

- Adds buttons for each menu item to the panelMenu.

- Adds an action listener to each button to handle adding orders.

- Creates and adds the text area for displaying orders.

- addOrder():

- Checks if the table number is entered.

- Creates an Order object and adds it to the orders list.

- Updates the order list display.

- Saves the order to a file.

- updateOrderList():

- Builds a string representation of all orders.

- Updates the text area with the order list.

- saveOrderToFile():

- Appends the order details to the orders.txt file.

- Also saves item details to menu_items.txt for analytics.

Explanation of Interfaces and Background Code

1. JFrame: The main window for the GUI.

2. JPanel: Used for grouping UI components.

3. JTextArea: A text area for displaying multiple lines of text.

4. JTextField: A single-line text field for user input.

5. JButton: A button that triggers actions when clicked.

6. GridLayout: A layout manager that arranges components in a grid.

7. BorderLayout: A layout manager that arranges components in five regions: north, south, east, west, and center.

8. JScrollPane: A scrollable view of a component.

Event Handling

- ActionListener: An interface for receiving action events.

- Implemented using anonymous inner classes for handling button clicks.

File I/O

- BufferedWriter: Used for writing text to files efficiently.

- FileWriter: A convenience class for writing character files.

Example Usage

When a user enters a table number and clicks on a menu item button, the addOrder() method:

- Reads the table number from the text field.

- Creates a new Order object with the specified item and price.

- Adds the order to the orders list.

- Updates the display with the current list of orders.

- Saves the order to orders.txt and menu_items.txt.

RestaurantManagementSystem Class

The RestaurantManagementSystem class extends JFrame and manages the GUI for table management and reservations.

Attributes

- textFieldReservationID: Text field for reservation ID (not editable, automatically assigned).

- textFieldTableNumber: Text field for table number input.

- textFieldGuestName: Text field for guest name input.

- textFieldGuestCount: Text field for guest count input.

- buttonAddReservation: Button to add a new reservation.

- textAreaReservations: Text area to display current reservations.

- mainApp: Reference to the main application.

Constructor

The constructor initializes the frame and UI components and sets up the background panel.

1. Frame Setup:

- Sets the frame title, size, default close operation, and location.

2. Background Panel:

- Adds a BackgroundPanel with an image as the background.

3. Content Panel:

- Creates a contentPanel with a BorderLayout and sets it as non-opaque to allow the background to be visible.

4. Top Panel:

- Creates a panelTop with a GridLayout for input fields and the add button.

- Adds labels and text fields for reservation details.

5. Add Reservation Button:

- Adds an action listener to handle adding reservations and updating the display and file.

6. Reservation Display Area:

- Creates a textAreaReservations to display current reservations and sets it as non-opaque.

- Adds the text area to a scroll pane in the center of the contentPanel.

7. Content Pane:

- Uses a JLayeredPane to manage the background and content panels.

- Adds the backgroundPanel at layer 0 and contentPanel at layer 1.

8. Load Reservations:

- Calls loadReservations() to populate the text area with existing reservations from the file.

Helper Methods

1. createLabel(String text):

- Creates a JLabel with the specified text and sets its color to white.

2. getReservationID():

- Reads the reservation.txt file to determine the next reservation ID.

- Returns the next available reservation ID.

3. writeToDataBaseTxt(JLabel ID):

- Appends the new reservation details to reservation.txt.

4. addReservation(JLabel ID):

- Adds the reservation details to the text area.

- Updates the active tables in the main application.

- Clears the input fields and updates the reservation ID.

5. clearFields(JLabel ID):

- Clears the input fields and updates the reservation ID field.

6. loadReservations():

- Reads reservation.txt and populates the text area with existing reservations.

- Updates the active tables in the main application.

7. isNumeric(String str):

- Checks if a string is numeric.

BackgroundPanel Inner Class

This inner class extends JPanel and handles custom painting of the background image.

1. Attributes:

- backgroundImage: The image to be used as the background.

2. Constructor:

- Initializes the backgroundImage.

3. paintComponent(Graphics g):

- Overrides the paintComponent method to draw the background image.

Usage and Flow

1. Initialization:

- When an instance of RestaurantManagementSystem is created, the constructor sets up the frame and loads existing reservations.

2. Adding Reservations:

- When the "Rezervasyon Ekle" button is clicked, the reservation details are appended to reservation.txt, and the text area is updated to display the new reservation.

3. Loading Reservations:

- On startup, the loadReservations() method reads the existing reservations from reservation.txt and displays them in the text area.

Overview

The code represents a simple Java Swing application for managing inventory items and suppliers. It includes adding new inventory items, displaying the inventory list, and reviewing the inventory through a graphical user interface (GUI).

Components and Structure

1. InventoryOptimizationAndSupplierManagement Class: This is the main class that extends JFrame and represents the GUI for managing the inventory and suppliers. It includes fields for user inputs and displays the inventory list.

2. InventoryItem Class: This is a simple class representing an inventory item with attributes such as name, quantity, and supplier. It also overrides the toString method to provide a readable format for inventory items.

Detailed Explanation

InventoryOptimizationAndSupplierManagement Class

Fields and Constructor:

- MainApplication mainApp: This field holds a reference to the main application, which allows interaction with other parts of the program.

- List<InventoryItem> inventoryItems: This list stores the inventory items.

- JTextArea textAreaInventory, JTextField textFieldItemName, JTextField textFieldItemQuantity, JTextField textFieldSupplier: These fields represent the text areas and text fields used for displaying and inputting data.

The constructor InventoryOptimizationAndSupplierManagement(MainApplication app) initializes the main application reference and inventory list, and calls initializeUI() to set up the GUI components.

initializeUI Method:

- Title and Size: Sets the title of the window and its size.

- Close Operation: Specifies the default close operation to dispose of the window when closed.

- Layout: Sets the layout manager for the frame to BorderLayout.

Panel Top:

- Uses a GridLayout with 4 rows and 2 columns to organize the input fields and the "Add Item" button.

- Adds labels and text fields for "Item Name", "Quantity", and "Supplier".

- Adds the "Add Item" button which, when clicked, will add a new inventory item.

Center Area:

- Adds a JTextArea wrapped in a JScrollPane to display the inventory list.

Bottom Area:

- Adds a "Review Inventory" button to trigger the review of the inventory.

Event Handling:

- buttonAddItem ActionListener: When the "Add Item" button is clicked, it calls the addInventoryItem method.

- buttonReviewInventory ActionListener: When the "Review Inventory" button is clicked, it calls the reviewInventory method.

addInventoryItem Method:

- Reads input from the text fields for item name, quantity, and supplier.

- Creates a new InventoryItem object and adds it to the inventory list.

- Updates the inventory list display and clears the input fields.

updateInventoryList Method:

- Builds a string representation of the inventory list by iterating over the inventoryItems list and appending each item to a StringBuilder.

- Sets the text of textAreaInventory to the built string.

clearFields Method:

- Clears the text fields for item name, quantity, and supplier.

reviewInventory Method:

- Builds a string report of the inventory by iterating over the inventoryItems list.

- Displays the report in a JOptionPane dialog.

InventoryItem Class

Fields and Constructor:

- String name, int quantity, String supplier: These fields store the name, quantity, and supplier of the inventory item.

- The constructor initializes these fields.

toString Method:

- Provides a readable string representation of the inventory item, including its name, quantity, and supplier.

Summary

The application provides a straightforward interface for managing an inventory list. Users can input item details, add them to the inventory, and review the current inventory list. The InventoryOptimizationAndSupplierManagement class handles the GUI and user interactions, while the InventoryItem class encapsulates the data for each inventory item. The use of JTextField for input, JTextArea for display, and JOptionPane for dialogs demonstrates basic Swing components and event handling in Java.

Let's break down the provided code and explain each part in detail.

StaffTrainingAndPerformanceManagement Class

The StaffTrainingAndPerformanceManagement class extends JFrame and manages the GUI for staff training and performance management.

Attributes

- mainApp: A reference to the main application.

- staffMembers: A list to store StaffMember objects.

- textAreaStaffMembers: A text area to display staff member information.

- textFieldStaffName: A text field for entering the staff member's name.

- textFieldTrainingModule: A text field for entering the training module.

- textFieldProgress: A text field for entering the progress percentage.

- textFieldPerformanceScore: A text field for entering the performance score.

Constructor

The constructor initializes the frame and UI components and sets up the background panel.

1. Frame Setup:

- Sets the frame title, size, default close operation, and layout.

2. Top Panel:

- Creates a panelTop with a GridLayout for input fields and the add button.

- Adds labels and text fields for staff member details.

3. Add Staff Button:

- Adds a button to add a new staff member and attaches an action listener.

4. Staff Member Display Area:

- Creates a textAreaStaffMembers to display current staff members and sets it as non-editable.

- Adds the text area to a scroll pane in the center of the frame.

5. Review Performance Button:

- Adds a button to review performance and attaches an action listener.

6. Adding Panels and Buttons:

- Adds the panelTop to the north of the frame.

- Adds the textAreaStaffMembers to the center of the frame.

- Adds the buttonReviewPerformance to the south of the frame.

Helper Methods

1. addStaffMember():

- Reads the staff member details from the text fields.

- Creates a new StaffMember object and adds it to the staffMembers list.

- Updates the display with the current list of staff members.

- Clears the input fields.

2. updateStaffList():

- Builds a string representation of all staff members.

- Updates the text area with the staff member list.

3. clearFields():

- Clears the input fields.

4. reviewPerformance():

- Builds a performance report string from the staff members.

- Displays the performance report in a message dialog.

StaffMember Class

The StaffMember class encapsulates the details of a staff member:

- Attributes:

  - name: The name of the staff member.

  - trainingModule: The training module the staff member is enrolled in.

  - progress: The progress percentage of the training.

  - performanceScore: The performance score of the staff member.

- Constructor:

  - Initializes the attributes with the given values.

- toString():

  - Returns a string representation of the staff member.

Detailed Explanation

1. Initialization:

- When an instance of StaffTrainingAndPerformanceManagement is created, the constructor sets up the frame and UI components.

2. Adding Staff Members:

- When the "Personel Ekle" button is clicked, the addStaffMember() method:

- Reads the input from the text fields.

- Creates a new StaffMember object with the input data.

- Adds the staff member to the staffMembers list.

- Updates the text area to display the current list of staff members.

- Clears the input fields.

3. Reviewing Performance:

- When the "Performans İncele" button is clicked, the reviewPerformance() method:

- Builds a performance report string from the staffMembers list.

- Displays the performance report in a message dialog.

This setup provides a functional GUI for managing staff training and performance, allowing the user to add staff members, view their details, and review performance reports.

Overview

The MainApplication class represents the main window of a restaurant management system. It includes user authentication based on roles (Waiter, Chief Waiter, Patron) and provides access to different management features based on the authenticated user's role. The UI uses JLayeredPane for layering components and includes background images, a main panel for features, and a dashboard panel for active tables and accounts.

Components and Structure

1. MainApplication Class: Extends JFrame and serves as the main application window. Handles user authentication, role-based UI setup, and manages active tables and accounts.

2. OrderAndKitchenCommunication Class: Represents a subsystem for order and kitchen communication (not fully implemented here, but referenced).

3. BackgroundPanel Class: A custom JPanel to display a background image.

4. InventoryOptimizationAndSupplierManagement Class: Represents one of the feature classes available in the system (already explained in your previous code).

Detailed Explanation

MainApplication Class

Fields and Constructor:

- OrderAndKitchenCommunication orderSystem: An instance for handling order and kitchen communication.

- BackgroundPanel backgroundPanel: A custom panel to display a background image.

- JPanel dashboardPanel, JPanel mainPanel: Panels for displaying the dashboard and main features.

- Role Passwords: Constants for storing predefined passwords for different roles (GARSON_PASSWORD, SEF_GARSON_PASSWORD, PATRON_PASSWORD).

- Map<Integer, Double> activeTables: A map to store active table numbers and their corresponding amounts.

- Buttons: Buttons for navigating the dashboard and features.

The constructor sets up the main application window, initializes UI components, and handles user authentication.

UI Setup:

- Title and Size: Sets the window title and size.

- Close Operation: Specifies the default close operation to exit the application.

- Layered Pane: Uses JLayeredPane with OverlayLayout to manage different layers (background, main panel, dashboard panel).

- Background Panel: Adds a custom background panel with an image.

- Main Panel: Initializes the main panel with a GridLayout for feature buttons.

- Dashboard Panel: Initializes the dashboard panel with GridBagLayout.

User Role Selection and Authentication:

- Prompts the user to select a role (Garson, Şef Garson, Patron) using a JOptionPane.

- Prompts for a password and verifies it using the isPasswordCorrect method.

- If the password is correct, calls setupUI to initialize the UI for the selected role and sets the layered pane as the content pane.

Password Verification (isPasswordCorrect Method):

- Checks if the entered password matches the predefined password for the selected role.

UI Setup Based on Role (setupUI Method):

- Initializes a list of buttons based on the user's role.

- For all roles, adds buttons for table management, order and kitchen communication, and payment and feedback.

- For Şef Garson and Patron, adds additional buttons for menu engineering, staff training, customer loyalty, inventory optimization, and marketing integration.

- Adds a "Back" button to return to the dashboard.

- Stores the buttons in an array (featureButtons) for later use.

Button Creation (createButton Method):

- Creates a button with specified text and action listener.

- Customizes the button appearance (foreground color, background color, border, font).

Active Tables Management:

- updateActiveTables: Updates the amount for a specified table.

- updateActiveTableAmount: Updates the amount after payment and removes the table if the amount is zero or less.

- removeActiveTable: Removes a table from the active tables.

- getActiveTableAmount: Retrieves the amount for a specified table.

Dashboard Display (showDashboard Method):

- Creates a dashboard panel showing active tables and their amounts.

- Adds a "Dashboard" button at the top and a panel displaying active tables.

- Updates the dashboardPanel and makes it visible.

Main Panel Display (showMainPanel Method):

- Displays the main panel with feature buttons.

- Hides the dashboardPanel.

Main Method:

- Runs the application using SwingUtilities.invokeLater.

BackgroundPanel Class

A custom JPanel to display a background image:

- Fields: Stores the background image.

- Constructor: Initializes the background image.

- paintComponent Method: Overrides the method to draw the background image.

Summary

The MainApplication class is the core of a restaurant management system, providing a role-based UI for managing various aspects of the restaurant. It includes user authentication, a layered UI with background images, and dynamic content based on the authenticated user's role. The BackgroundPanel class enhances the UI by displaying a custom background image.

Let's break down the provided code and explain each part in detail.

TablesidePaymentAndFeedback Class

The TablesidePaymentAndFeedback class extends JFrame and manages the GUI for tableside payment and feedback.

Attributes

- textFieldTableNumber: Text field for entering the table number.

- buttonLoadOrders: Button to load orders for the specified table number.

- buttonPaySelected: Button to pay for selected orders.

- listOrders: JList to display the orders for the specified table.

- listModel: DefaultListModel to manage the data for the listOrders.

- orderPrices: Map to store the prices of orders keyed by order ID.

- totalAmountLabel: Label to display the total amount to be paid.

- mainApp: Reference to the main application.

Constructor

The constructor initializes the frame and UI components.

1. Frame Setup:

   - Sets the frame title, size, default close operation, and layout.

2. North Panel:

   - Creates a northPanel with a GridLayout to contain the input field for the table number and the load orders button.

   - Adds a label, text field, and button to the northPanel.

- Adds the northPanel to the north of the frame.

3. Orders List:

- Creates a listModel and a listOrders to display the orders.

- Adds the listOrders to a scroll pane and adds it to the center of the frame.

4. South Panel:

- Creates a southPanel with a BorderLayout to contain the total amount label and the pay button.

- Adds the total amount label and the pay button to the southPanel.

- Adds the southPanel to the south of the frame.

5. Button Action Listeners:

- Adds an action listener to the buttonLoadOrders to load orders for the specified table number.

- Adds an action listener to the buttonPaySelected to pay for selected orders.

Helper Methods

1. loadOrders(ActionEvent e):

   - Clears the listModel and orderPrices.

   - Reads the orders.txt file and filters orders based on the specified table number.

   - Calculates the total amount for the orders.

   - Updates the listModel with the filtered orders.

   - Updates the totalAmountLabel with the total amount.

2. paySelectedItems(ActionEvent e):

- Gets the selected items from the listOrders.

- Iterates through the selected items and processes the payment:

  - Removes the paid item from the listModel.

  - Updates the orders.txt file by removing the paid item.

- Updates the active tables in the main application.

- Displays a confirmation dialog for the payment.

- Reloads the orders to update the list and total amount.

- Removes the table from active tables if the total amount is zero.

3. updateOrdersFile(int paidItemId):

- Reads the orders.txt file and writes all lines except the paid item to a temporary file.

- Replaces the original file with the temporary file.

Detailed Explanation

1. Initialization:

- When an instance of TablesidePaymentAndFeedback is created, the constructor sets up the frame and UI components.

2. Loading Orders:

- When the "Siparişleri Yükle" button is clicked, the loadOrders() method:

- Reads the input from the text field for the table number.

- Reads the orders.txt file and filters orders based on the specified table number.

- Calculates the total amount for the orders.

- Updates the listModel with the filtered orders.

- Updates the totalAmountLabel with the total amount.

3. Paying for Selected Items:

- When the "Seçilenleri Öde" button is clicked, the paySelectedItems() method:

- Gets the selected items from the listOrders.

- Iterates through the selected items and processes the payment:

- Removes the paid item from the listModel.

- Updates the orders.txt file by removing the paid item.

- Updates the active tables in the main application.

- Displays a confirmation dialog for the payment.

- Reloads the orders to update the list and total amount.

- Removes the table from active tables if the total amount is zero.

This setup provides a functional GUI for managing tableside payments and feedback, allowing the user to load orders, pay for selected items, and manage active tables.

Overview

The MenuEngineeringAndAnalytics class is a Java Swing application that provides functionality for analyzing menu items in a restaurant. It reads data from a file, calculates revenue and sales figures for each menu item, and generates a report. The GUI includes a button to trigger the analysis and a text area to display the results.

Components and Structure

1. MenuEngineeringAndAnalytics Class: Extends JFrame and represents the GUI for menu engineering and analytics. It includes fields for displaying menu items and buttons to trigger analysis.

2. MainApplication Class Reference: Holds a reference to the main application, allowing interaction with other parts of the system.

Detailed Explanation

MenuEngineeringAndAnalytics Class

Fields and Constructor:

- MainApplication mainApp: This field holds a reference to the main application.

- JTextArea textAreaMenuItems: A text area for displaying menu items and analysis results.

The constructor MenuEngineeringAndAnalytics(MainApplication app) initializes the main application reference and calls initializeUI() to set up the GUI components.

initializeUI Method:

- Title and Size: Sets the title of the window and its size.

- Close Operation: Specifies the default close operation to dispose of the window when closed.

- Layout: Sets the layout manager for the frame to BorderLayout.

Top Panel:

- Adds a label indicating "Menu Analysis Operations" using a GridLayout with a single cell.

- Adds this panel to the north region of the BorderLayout.

Center Area:

- Adds a JTextArea wrapped in a JScrollPane for displaying menu items and analysis results.

- The text area is set to be non-editable.

Bottom Area:

- Adds an "Analyze Menu" button to the south region of the BorderLayout.

- Registers an ActionListener to the button to trigger the analyzeMenu method when clicked.

analyzeMenu Method:

- Data Structures: Initializes two maps, itemRevenue and itemSales, to store revenue and sales quantities for each menu item.

- File Reading: Uses a BufferedReader to read menu items from a file named "menu_items.txt". Each line in the file represents a menu item with its name, price, and quantity sold, separated by commas.

- Processing Each Line: Splits each line into parts, parses the item name, price, and quantity, and updates the itemRevenue and itemSales maps.

- Error Handling: Displays an error message if there is an issue reading the file.

Generating the Analysis Report:

- StringBuilder: Constructs a detailed report of the analysis using a StringBuilder.

- Total Revenue Calculation: Sums up the total revenue from all items.

- Popular Item Identification: Finds the item with the highest revenue.

- Appending Results: Appends each item's sales and revenue, total revenue, and the most popular item to the report.

Displaying the Report:

- Shows the analysis report in a JOptionPane dialog.

Summary

The MenuEngineeringAndAnalytics class provides a straightforward interface for analyzing restaurant menu items. It reads data from a file, processes the information to calculate revenues and sales quantities, and generates a comprehensive analysis report. The GUI consists of a label, a text area for displaying results, and a button to trigger the analysis. The class handles file reading and data processing efficiently, and it provides clear feedback to the user through dialogs.

Overview

The MenuItem class represents a single menu item in a restaurant, encapsulating its name, price, and the number of sales. This class includes methods for accessing these properties and provides a string representation of the item.

Components and Structure

1. Fields:

- name: A string representing the name of the menu item.

- price: A double representing the price of the menu item.

- sales: An integer representing the number of sales of the menu item.

2. Constructor:

- Initializes the fields name, price, and sales with values provided as arguments.

3. Getter Methods:

- getName(): Returns the name of the menu item.

- getPrice(): Returns the price of the menu item.

- getSales(): Returns the number of sales of the menu item.

4. toString Method:

- Overrides the default toString method to provide a readable string representation of the menu item, including its name, price, and sales.

Detailed Explanation

Fields:

- The name field stores the name of the menu item, such as "Pasta" or "Pizza".

- The price field stores the price of the menu item in double format, for example, 12.99.

- The sales field stores the number of units sold for this menu item, such as 150.

Constructor:

- The constructor takes three parameters: name, price, and sales.

- It initializes the name field with the provided name.

- It initializes the price field with the provided price.

- It initializes the sales field with the provided sales count.

Getter Methods:

- getName(): This method returns the value of the name field.

- getPrice(): This method returns the value of the price field.

- getSales(): This method returns the value of the sales field.

toString Method:

- The toString method is overridden to return a formatted string.

- The string includes the name, price (formatted as "X TL"), and the number of sales.

- This provides a human-readable format useful for displaying menu item details.

Summary

The MenuItem class is a simple model class that encapsulates the properties of a menu item in a restaurant. It includes fields for the name, price, and sales count, and provides methods to access these properties. The class also includes a toString method to return a formatted string representation of the menu item, making it easy to display the item details in a readable format.

Overview

The CustomerLoyaltyAndEngagement class represents a GUI module for managing customer loyalty and engagement programs within a restaurant management system. It extends JFrame and includes the necessary setup for displaying a window with a specific title and layout.

Components and Structure

1. CustomerLoyaltyAndEngagement Class: Extends JFrame to create a window for customer loyalty and engagement functionalities. It includes fields and methods to set up the user interface.

2. MainApplication Class Reference: Holds a reference to the main application, allowing interaction with other parts of the system.

Detailed Explanation

CustomerLoyaltyAndEngagement Class

Fields and Constructor:

- MainApplication mainApp: This field holds a reference to the main application, allowing the module to interact with it.

- The constructor CustomerLoyaltyAndEngagement(MainApplication app) initializes the main application reference and calls initializeUI() to set up the GUI components.

initializeUI Method:

- Title and Size: Sets the title of the window to "Müşteri Bağlılığı ve Sadakat Programları" (Customer Loyalty and Engagement Programs) and its size to 400x300 pixels.

- Close Operation: Specifies the default close operation to dispose of the window when closed.

- Layout: Sets the layout manager for the frame to BorderLayout.

Center Area:

- Adds a JLabel with the text "Müşteri Bağlılığı ve Sadakat Programları Modülü" to indicate the purpose of the module.

- The label is centered horizontally within its container by setting its horizontal alignment to JLabel.CENTER.

Summary

The CustomerLoyaltyAndEngagement class provides a basic structure for a GUI module dedicated to customer loyalty and engagement programs. It extends JFrame, includes a reference to the main application, and sets up a simple user interface with a centered label indicating the module's purpose. The window is configured with a specific title and size, and it will close properly when the user decides to close it. This class serves as a placeholder for future development where additional GUI components and functionalities related to customer loyalty and engagement will be added.

CHAPTER FIVE

This project aimed at developing a complete restaurant management system with several modules to ensure smooth operations, increase the satisfaction of the customers, and optimize the resources used. The major modules involved in the system were:

1. Main Application Interface: This was the central point that allows different types of users—Garson, Şef Garson, and Patron—to access several functionalities depending on their roles. Interface provided functionality to navigate easily through different modules.

2. Order and Kitchen Communication: This module facilitated managing orders from the table side, meaning commands are to be sent from the kitchen to the serving staff. Real-time processing was made to place and update the orders, along with adding an intuitive user interface so that anyone could easily use it.

3. Tableside Payment and Feedback: This is a feature that allowed tableside payment so that the customers could pay for their orders right on the table. There would also be room for feedback collection, and that way, experiences of the customers could be recorded and corrective measures taken.

4. Menu Engineering and Analytics: An analytical tool which could give insights or indications about the performance of a menu and assist the management with decisions

about the direction to take the menu items, as it will closely follow the sales and popularity of various dishes and give recommendations for the optimization of the menu.

5. Inventory Optimization and Supplier Management: This module eased the process functions of inventory management by tracking the stock levels and maintaining good relationship with suppliers to ensure optimal levels of inventory and reduction of waste.

6. Restaurant Management System: Part of the software suite that focused on table management and reservations, thus helping employees to effectively and in real-time handle the reservation related data, update statuses of tables, and so on.

It emphasized the development of a user-friendly interface, ensuring data integrity and real-time updating of all information in the various modules. This was targeted to make the system more operational, appreciated by the customers, and able to give insight into the business.

The Future Work

While the existing system encompasses all basic functions, a few enhancements and features will help optimize the efficiency of the system and give a better user experience overall.

1. Inclusion of Mobile Application: A mobile version of the application can bring flexibility and remote access for both staff and customers to update information in real time.

2. Advanced Analytics: Advanced analytics and machine learning algorithms within applications could help with customer choice prediction, staffing optimization, and inventory need forecasting.

3. Advanced Customer Feedback System: A more granular feedback mechanism, potentially including sentiment analysis, would reveal deeper insights into the levels of satisfaction of the customers and the areas that might need to be fixed.

4. Loyalty Programs: Setting up common loyalty programs to reward customers on a regular basis and encourage them to visit again.

5. Third-Party Integrations: Popular third-party services are online reservation platforms, food delivery services, and payment gateways with which the system can be integrated.

6. Automated Ordering System: The presence of self-service kiosks or table-side tablets for customers to place their own orders will help reduce the hassle of order placement and avoid delays.

7. Comprehensive Reporting: The reporting abilities should be enhanced so that it can provide detailed and customizable reports related to variable aspects of the restaurants, which can provide valuable insights towards managing the place.

8. Cloud-Based System: The adoption of a cloud-based architecture would increase further scalability, data security, and accessibility, hence easily managed for multi-location and large dataset usage.

9. Real-time Notifications and Alerts: Real-time notifications that can indicate to the management low inventories, high wait times, or even customers' complaints to start action and get it sorted right away.

10. Strengthen Security Features: Strengthen the security features we have on board so as to safeguard any customer-sensitive data, financial transactions, and even internal communications from possible breaches and cyber threats. With those next improvements, the management system for the restaurant will be lifted up to a level in which the solution becomes much more solid, scalable, and flexible, delivering increased value to its owners and users, including customers.