# Golden Dragon Hot Pot House Restaurant System

**COMP 246: Object Oriented Software Engineering Project Part A, B, C**

Jiawei (Gerry) Wu, Chao (Fiona) Feng, Gordon Stevens

Instructor: Mohamed Khan

**Table of Contents**

Problem, need and solution statements

**Problem**: Restaurant patrons want to visualize their meal before they order, and want to make the ordering process as fast and accurate as possible.

**Need**: A restaurant information system to process orders and generate sales reporting.

**Solution**: An information system is proposed

- Membership registration subsystem
- Ordering subsystem
- Payment subsystem
- Reporting subsystem
- Menu management subsystem

Vision statement: Problem description, system capabilities and business benefits

**Problem Description**

With hundreds of years of tradition, Chinese Restaurants have been feeding and entertaining patrons. Many people have experience and comfort with modern technology and most importantly, to save time, they order food using technology. To maintain a competitive advantage, it is important to quickly and accurately collect ordering information for the restaurant. Management will be able to track business performance and transactions.

Additionally the customer can have the menu in several languages, have pictures with detailed descriptions, see current promotions and estimated wait times.

It is recommended that the restaurant ordering system uses a web application. The web application will run on portable tablet computers in the restaurant.

**System Capabilities**

The web application system should be capable of:

- Displaying the menu and current specials, with details and pictures
- Assisting customer with membership account signup
- Allowing the customer to order food and confirm it
- Sending confirmed order notification to the kitchen
- Sending prepared food notification to wait staff
- Having customer request bill and choose payment method
- Requesting customer feedback
- Displaying table information: number of guests and order information
- Assisting management with daily, weekly, monthly and yearly sales
- Having management create, update or delete menu items

**Business Benefits**

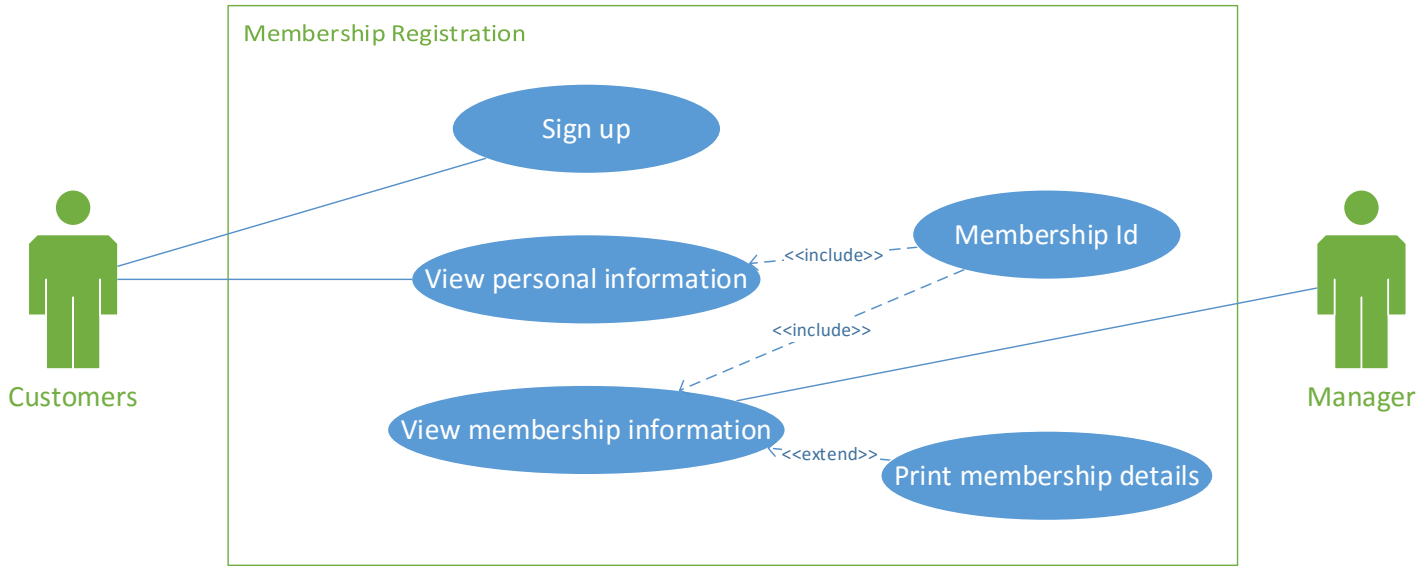
The new system will bring benefits to the restaurant:

- Decrease and track waiting times
- Increase accuracy of orders with built-in upselling
- Track financial progress of the business
- Tracking food distributor orders

User goals chart

| User | User goal and resulting use case |
|---|---|
| Customer | Sign up<br>View personal information<br>View menu and items<br>Order items<br>Cancel ordered items<br>Confirm order<br>View ordered items<br>Request bills<br>View bills<br>Provide feedbacks<br>View feedbacks |
| Waiter | View ordered items<br>View notification of prepared items<br>View bills<br>Confirm payment<br>View feedbacks |
| Kitchen | View ordered items<br>Confirm prepared items |
| Manager | View membership information<br>View bills<br>View feedbacks<br>Require reports<br>View reports<br>Download reports<br>Add menu items<br>Modify menu items<br>Delete menu items<br>View Menu |

Use-case diagram: Membership registration subsystem



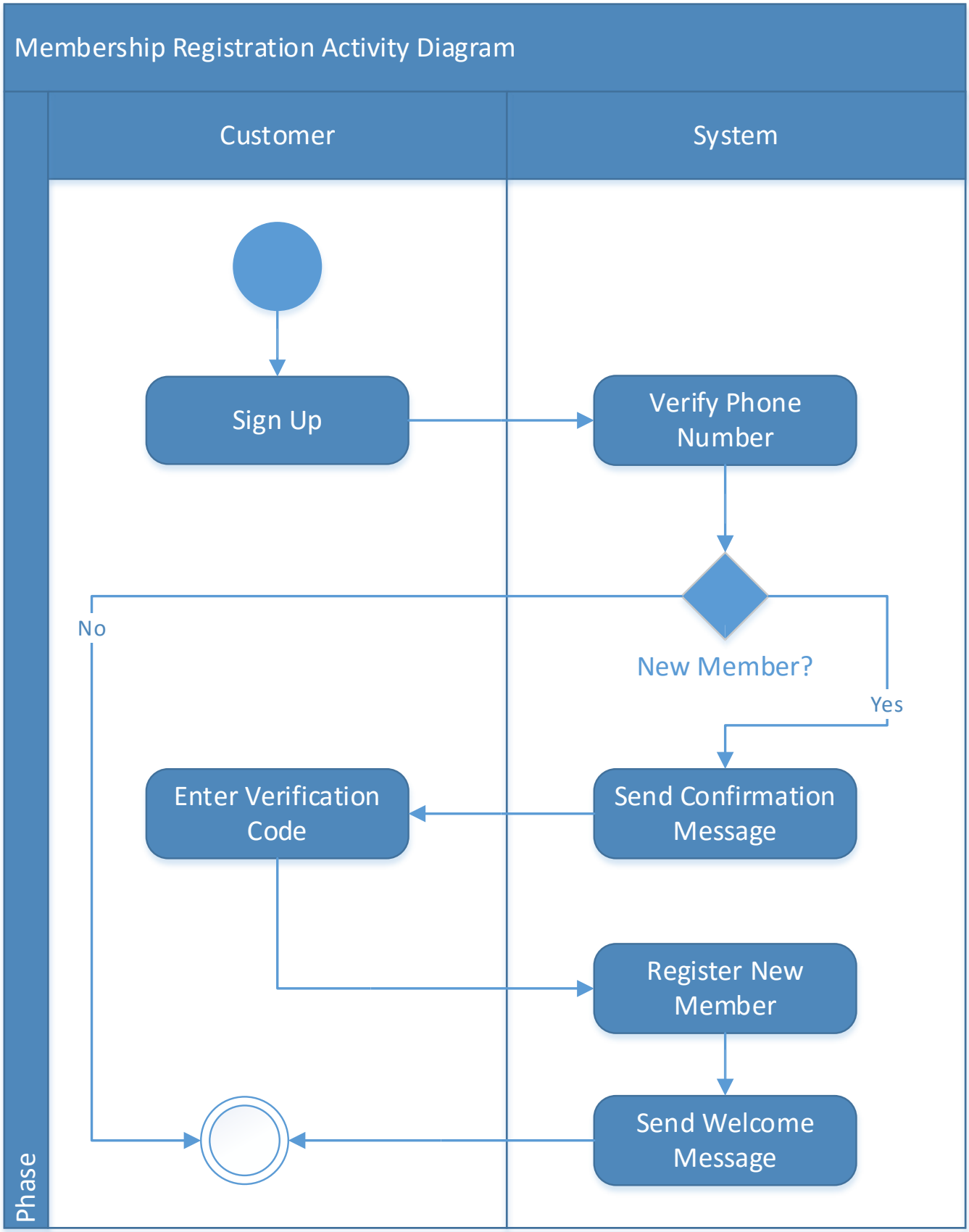

Use-case description: Membership registration subsystem

| Use case | Brief use case description |
|---|---|
| Sign up | Customer enters personal information in the system, and the system verifies the information and then creates a new membership account |
| View personal information | Customer requests to the personal membership information, and the system display the information |
| View membership information | Manager requests to view a collection of membership information, and the system displays the information |

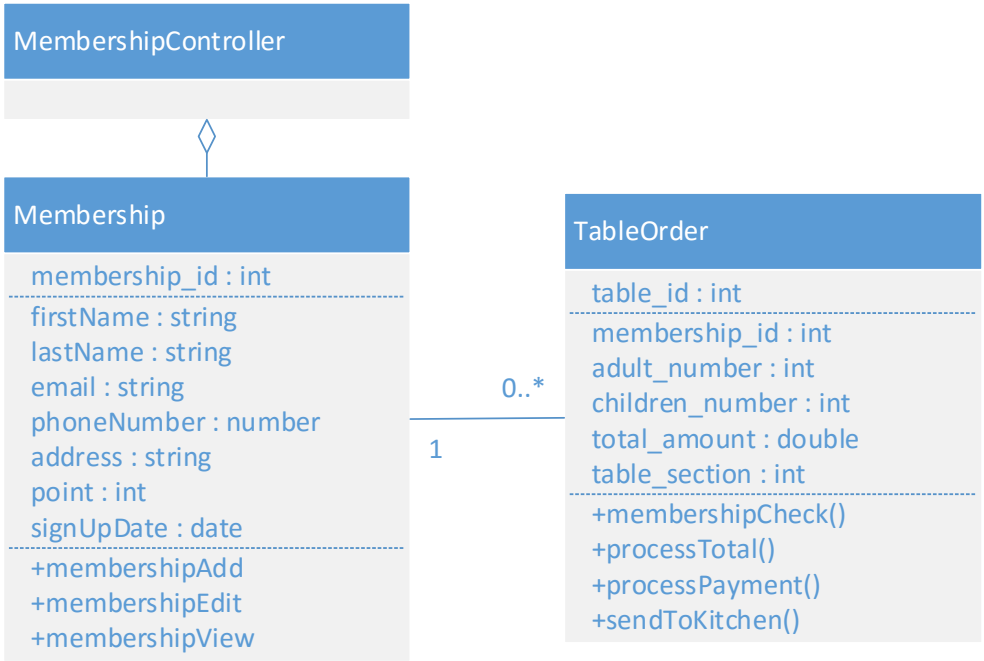
Workflow description: Membership registration workflow

1. Membership Registration Workflow
   - 1.1. Customer sign up.
   - 1.2. System verifies phone number.
   - 1.3. If it is an existing membership, no registration is needed; if it is new member, system sends confirmation message to customer.
   - 1.4. Customer enters verification code from the message.
   - 1.5. System registers new member.
   - 1.6. System sends welcome message to customer.

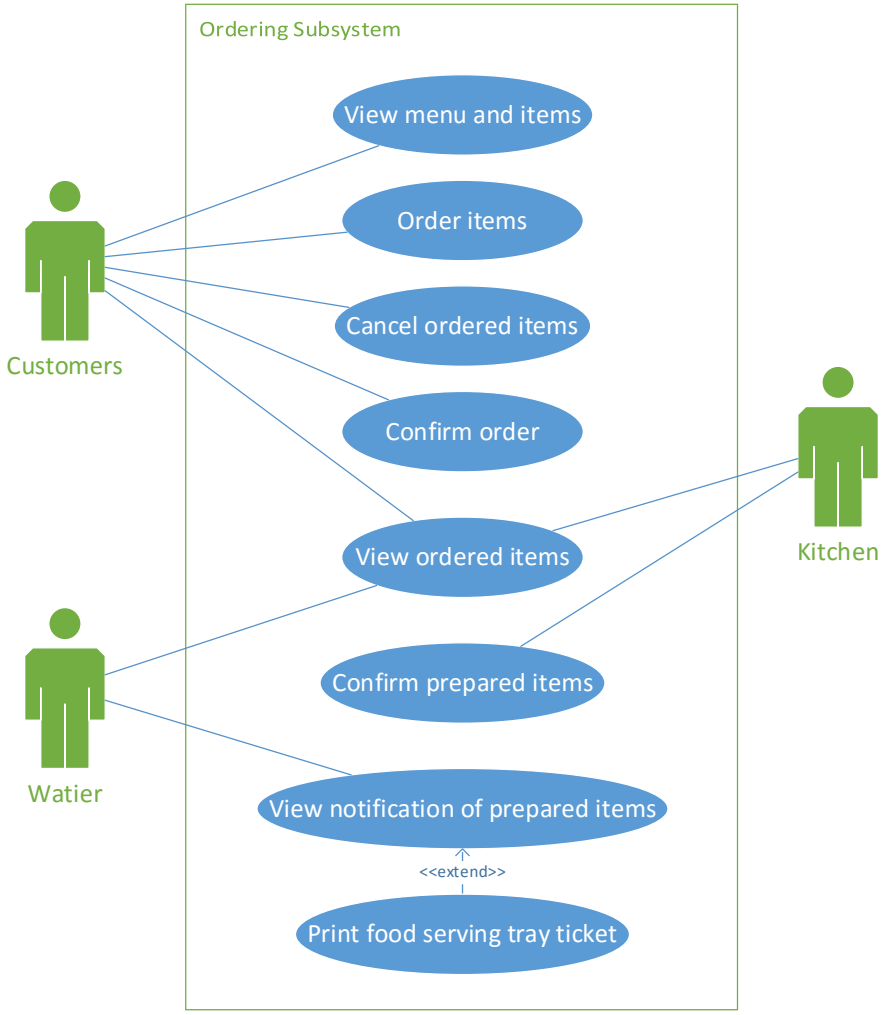
Membership registration activity diagram



Design class diagram: Membership




Use-case diagram: Ordering subsystem



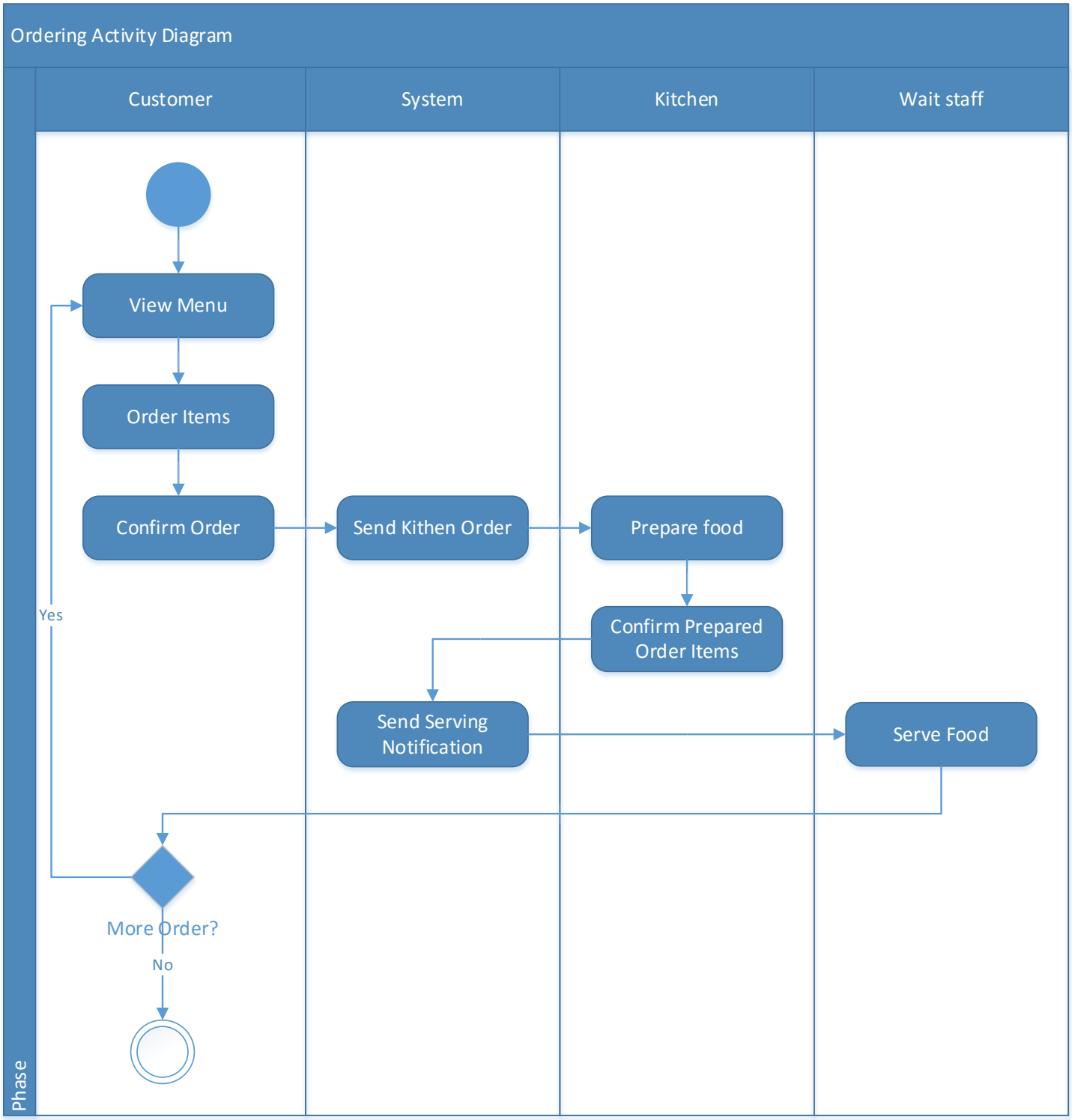
Use-case description: Ordering subsystem

| Use case | Brief use case description |
|---|---|
| View menu and items | Customer requests to view the menu, and the system display the pictures and price of dishes |
| Order items | Customer chooses items from the menu |
| Cancel ordered items | Customer remove the items they does not want to order |
| Confirm order | Customer views the ordered items and confirm order, the system create the order list and send the orders information |
| View ordered items | Customer requests to view the confirmed ordered list, the system display the list |
| Confirm prepared items | Kitchen confirm the items they prepared ,and the system send the notification |
| View notification of prepared items | System display the notification, and waiter views the notification of prepared items |

Workflow description: Ordering subsystem workflow

2. Ordering Workflow
    - 2.1. Customer views menu and items.
    - 2.2. Customer orders different items.
    - 2.3. Customer confirms order.
    - 2.4. System sends order to Kitchen.
    - 2.5. Kitchen prepares food according to order.
    - 2.6. Kitchen confirms prepared order items.
    - 2.7. System sends serving notification to wait staff.
    - 2.8. Wait staff serves food to Customer.
    - 2.9. If customer wants to place another order, go to 2.1; if customer finishes ordering, exits process.

Ordering activity diagram



User story: Order items

**Order Items User Story**

As a customer, I want to play an order as quickly as possible.

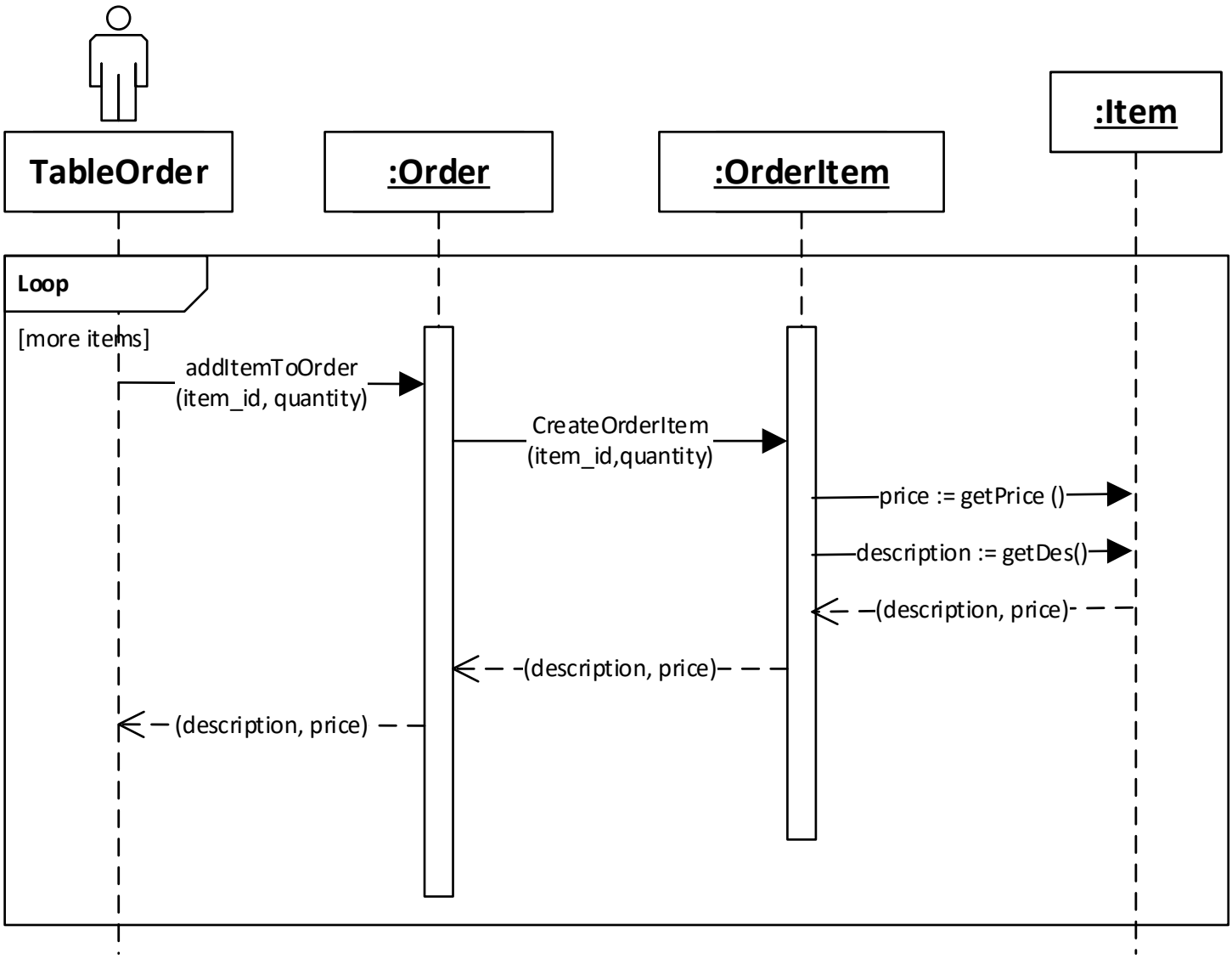
**Acceptance Criteria:**

1. Portable device (iPad) would cut down waiting time.
2. Sort available items by categories.
3. Recommend popular items.
4. Display item description and photos to help customer make decision.
5. Show ordered items and confirm order.

Detailed use-case: Order items

| | | |
|---|---|---|
| Use case name: | Order items | |
| Scenario: | Customer orders items from the menu. | |
| Triggering event: | Customer viewed menu items. | |
| Brief description: | Customer views the menu and adds the menu items to the order. | |
| Actors: | Customer. | |
| Related use cases: | None. | |
| Stakeholders: | Customer. | |
| Preconditions: | Customer views menu before ordering. | |
| Postconditions: | Customer can view ordered items.<br>Customer can cancel items. | |
| Flow of activities: | Actor: | System: |
| | 1. Customer requests order item<br><br>2. Customer choose items and add them to order list | 1.1 System receives the request of ordering item<br><br>2.1 System adds items to order list and show the result |
| Exception conditions: | Customer cannot order item(s) if sold out. | |

System sequence diagram: Order items use-case



User story: View ordered items

**View Ordered Items User Story**

As a customer/employee, I want to view an order status as quickly and accurately as possible.

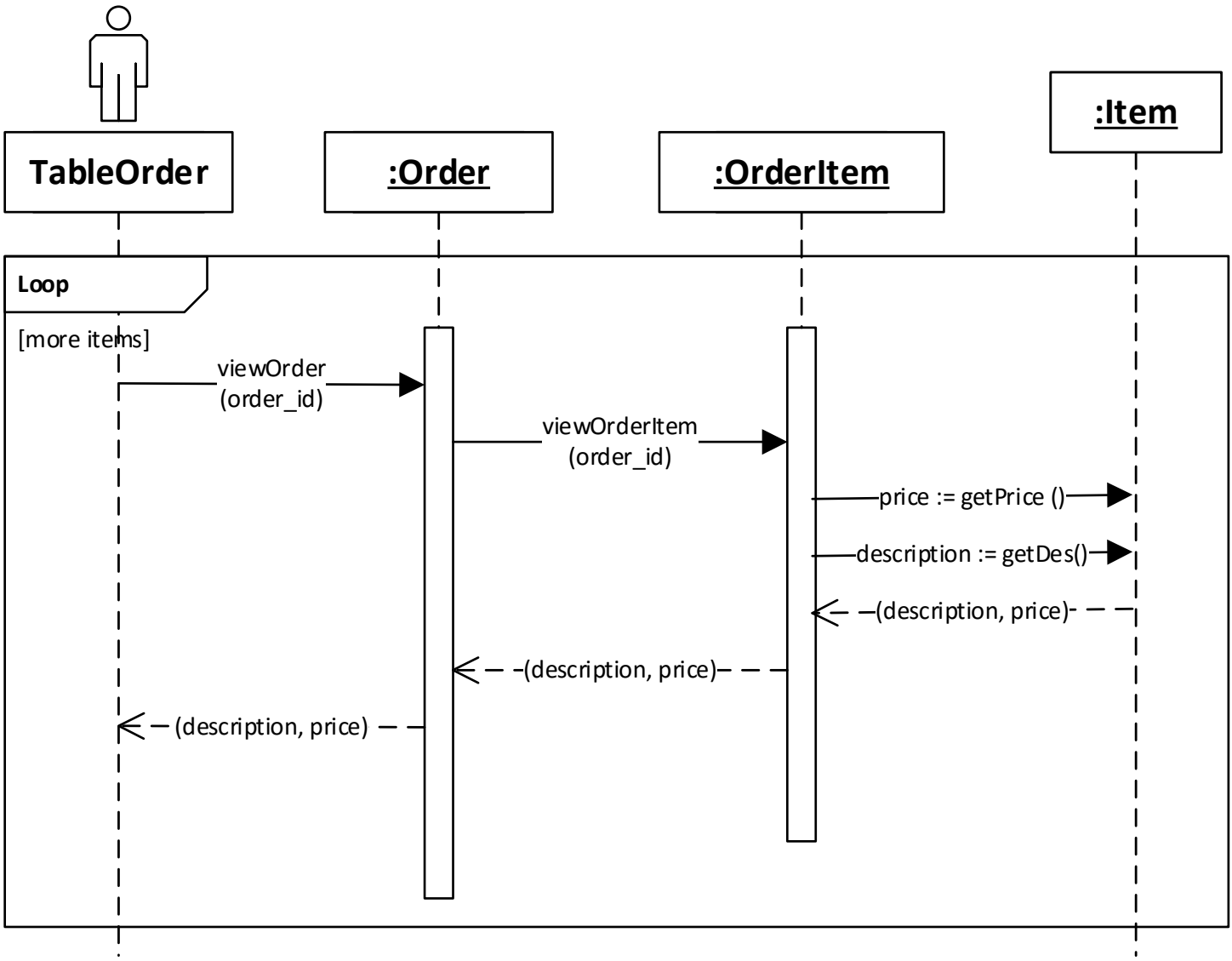
**Acceptance Criteria:**

1. Synchronically update new order in all devices (customer's iPad, waiter's station computer and kitchen's computer)
2. Highlight unserved items
3. Kitchen staff can print table order receipt
4. Ordered Items will be sorted according to kitchen station and displayed on different screen.

Detailed use-case: View ordered items

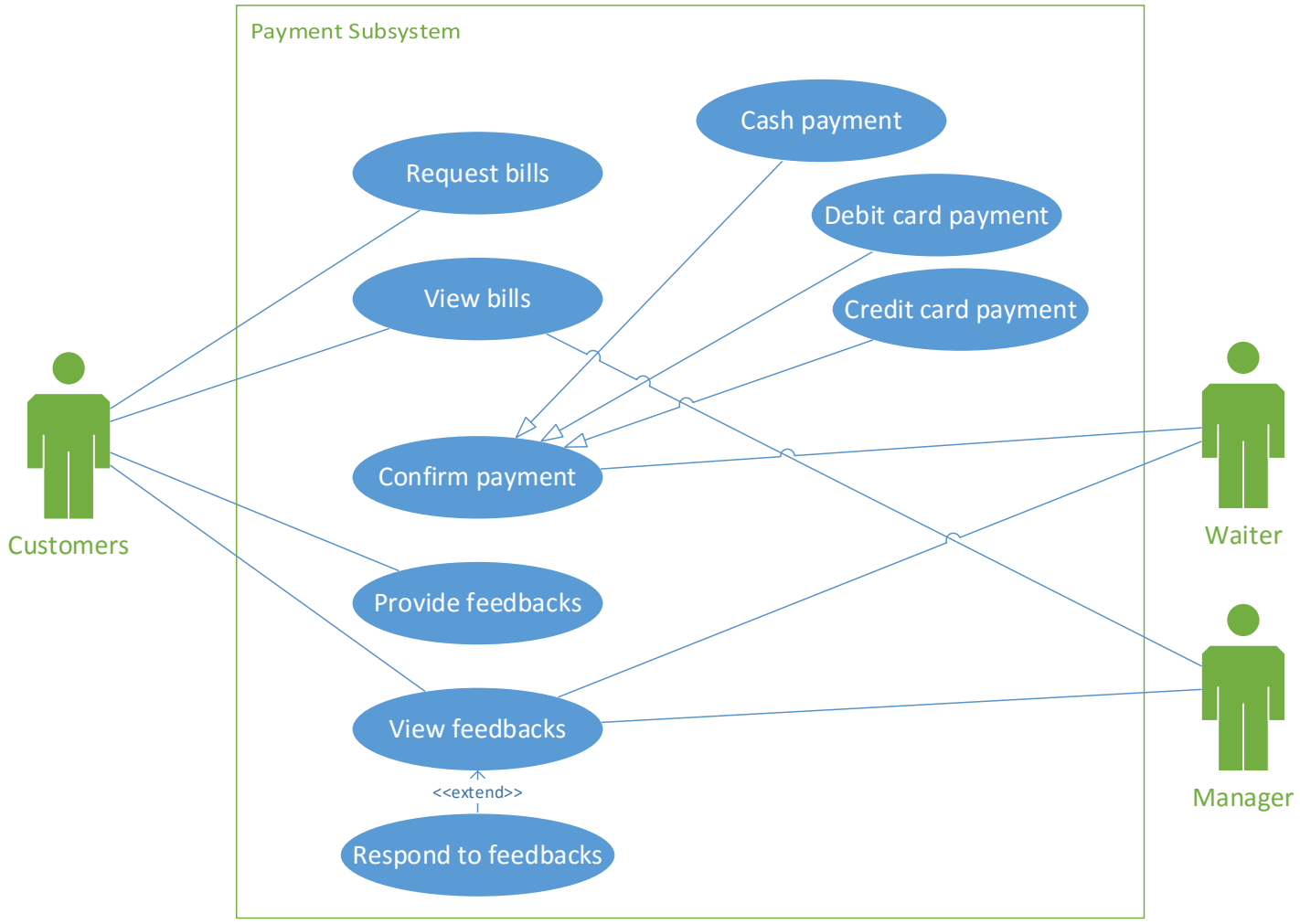
| | | |
|---|---|---|
| Use case name: | View ordered items | |
| Scenario: | Customer view the ordered items | |
| Triggering event: | Customer, waiter and kitchen staff want to view the order list | |
| Brief description: | Customer, waiter or kitchen request to view the confirmed ordered list and then the system display the list | |
| Actors: | Customer, Waiter, Kitchen | |
| Related use cases: | None | |
| Stakeholders: | Customer, Waiter, Kitchen | |
| Preconditions: | Customer must exist<br>Items must be ordered | |
| Postconditions: | Ordered items are confirmed by kitchen<br>Ordered items are prepared by kitchen<br>Ordered items are passed on table by waiter | |
| Flow of activities: | Actor: | System: |
| | 1. Customer, waiter and kitchen request view the ordered items.<br><br>2. Customer, waiter and kitchen views the ordered items. | 1.1 System receive the request from customer, waiter and kitchen of viewing the ordered items.<br><br>2.1 System shows the information of ordered items in the screen. |
| Exception conditions: | None | |

System sequence diagram: View ordered items use-case



Design class diagram: Ordering

Use-case diagram: Payment subsystem



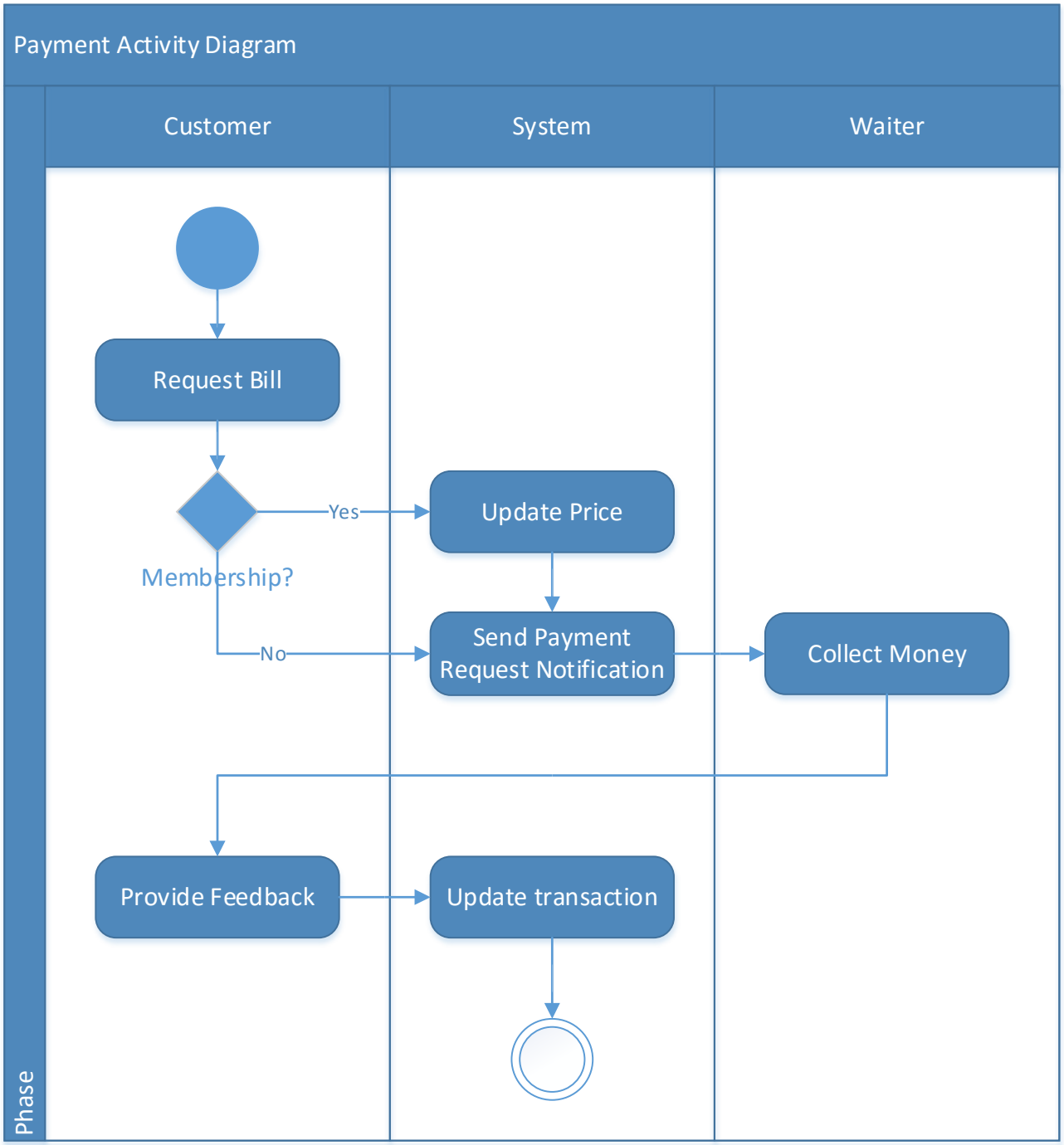

Use-case description: Payment subsystem

| Use case | Brief use case description |
|---|---|
| Request bills | Customer requests bill, the system create and send the bills |
| View bills | System display the bill, customer, waiter and manager view the bills |
| Confirm payment | Waiter confirms payment on system after customers pay for their bills |
| Provide feedbacks | Customer enter feedbacks into the system |
| View feedbacks | Customer, waiter and manager view the feedbacks from system |

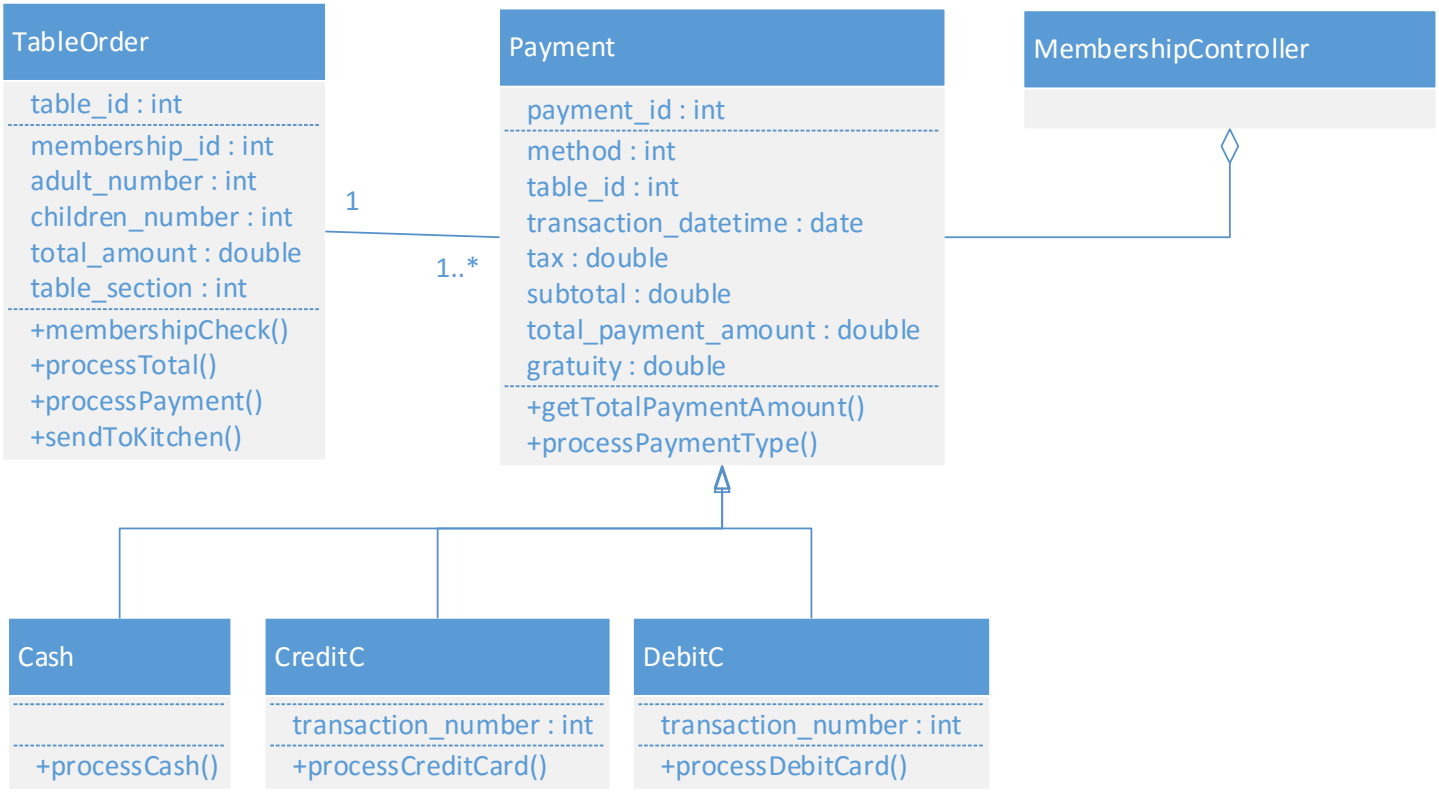
Workflow description: Payment subsystem workflow

3. Payment Workflow
3.1. Customer requests bill.
3.2. If customer has membership, system updates price and discount; if not, go to 3.3.
3.3. System sends payment request notification to wait staff.
3.4. Wait staff collects money.
3.5. Customer provides feedback.
3.6. System updates transaction.

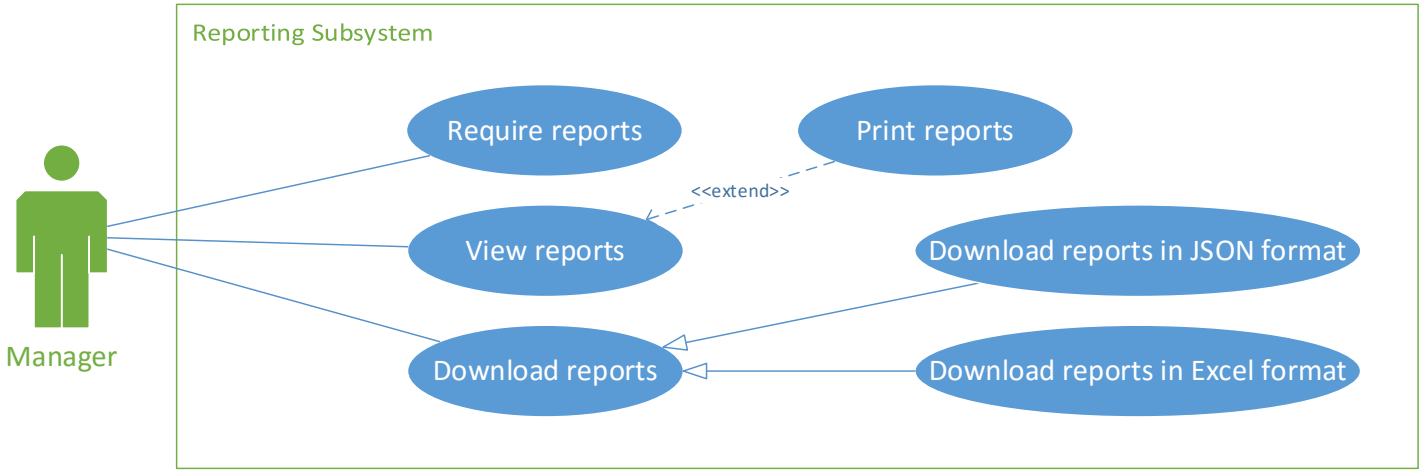
Payment activity diagram



Design class diagram: Payment




Use-case diagram: Reporting subsystem



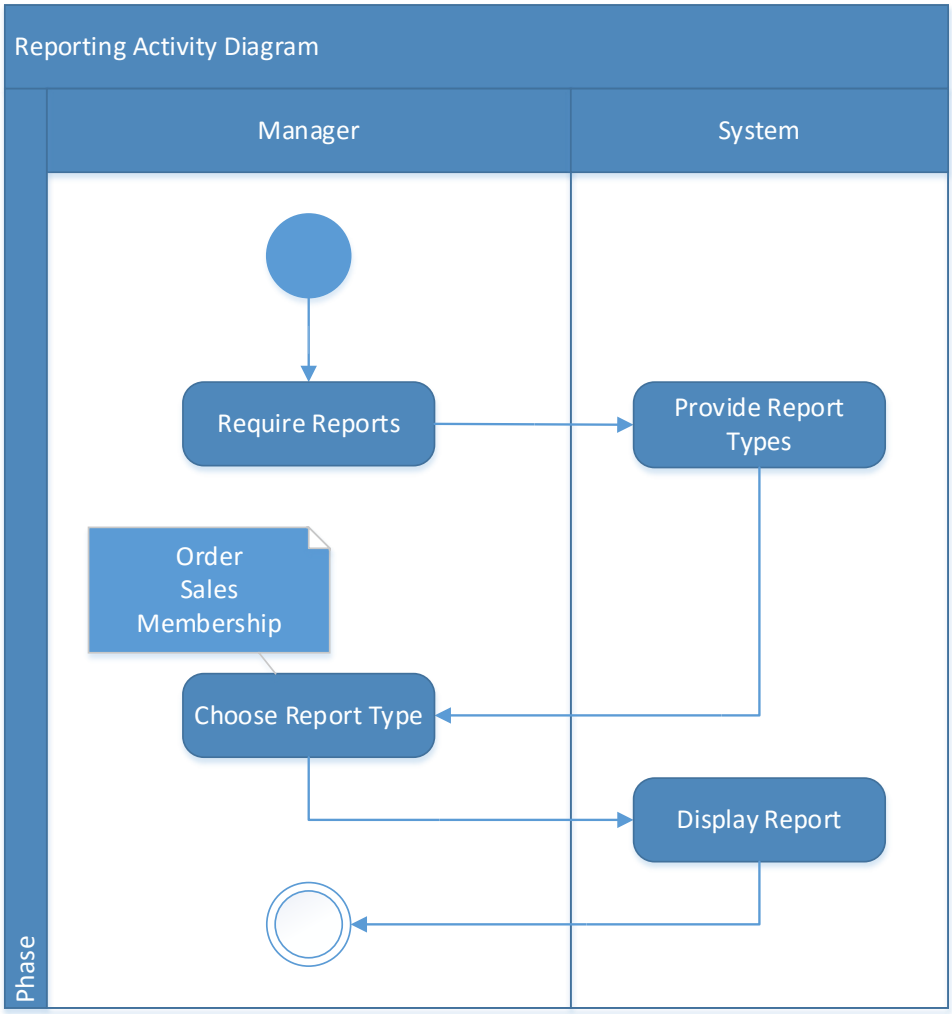
Use-case description: Reporting subsystem

| Use case | Brief use case description |
|---|---|
| Require reports | Manager requests the report and choose the reports type, the system creates and send the reports |
| View reports | System displays the reports and manager views the reports |
| Download reports | Manager downloads the reports from system |

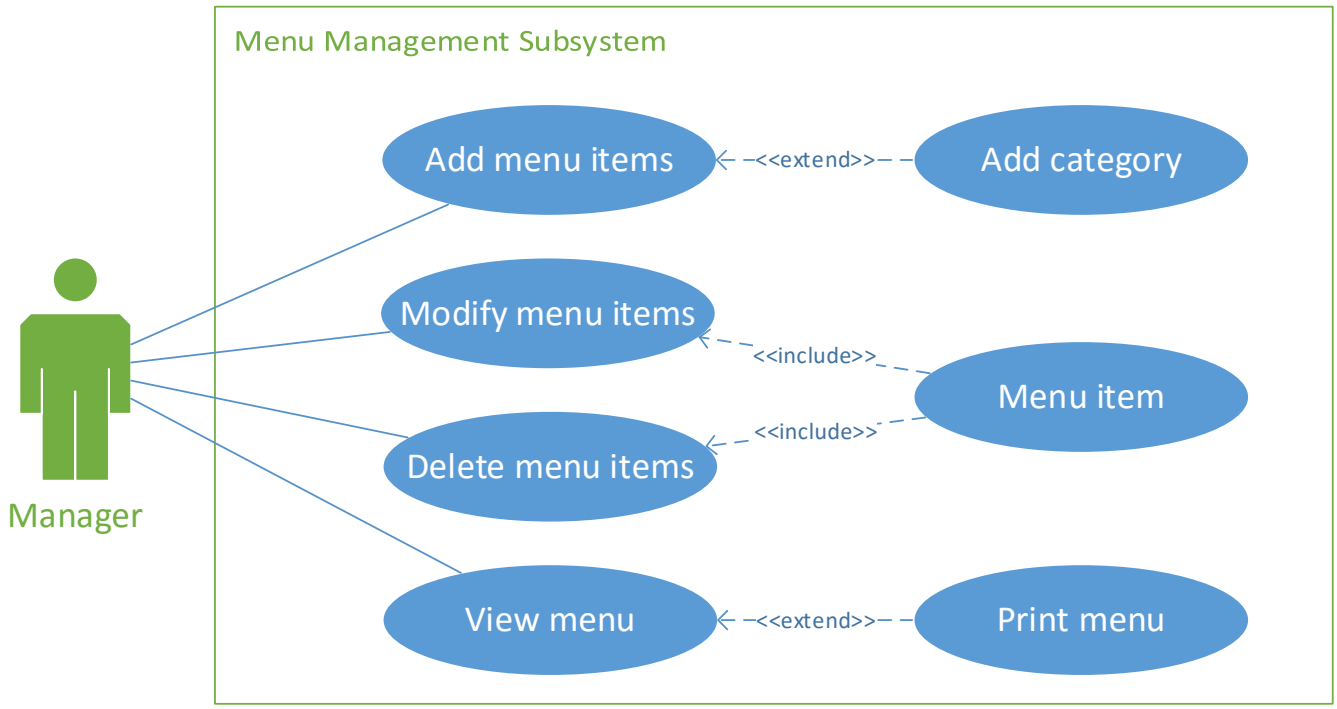
Workflow description: Reporting subsystem workflow

4. Reporting Workflow
4.1. Manager requires reports.
4.2. System provides different types of reports.
4.3. Manager chooses report type.
4.4. System displays report.

Reporting activity diagram



Use-case diagram: Menu management subsystem



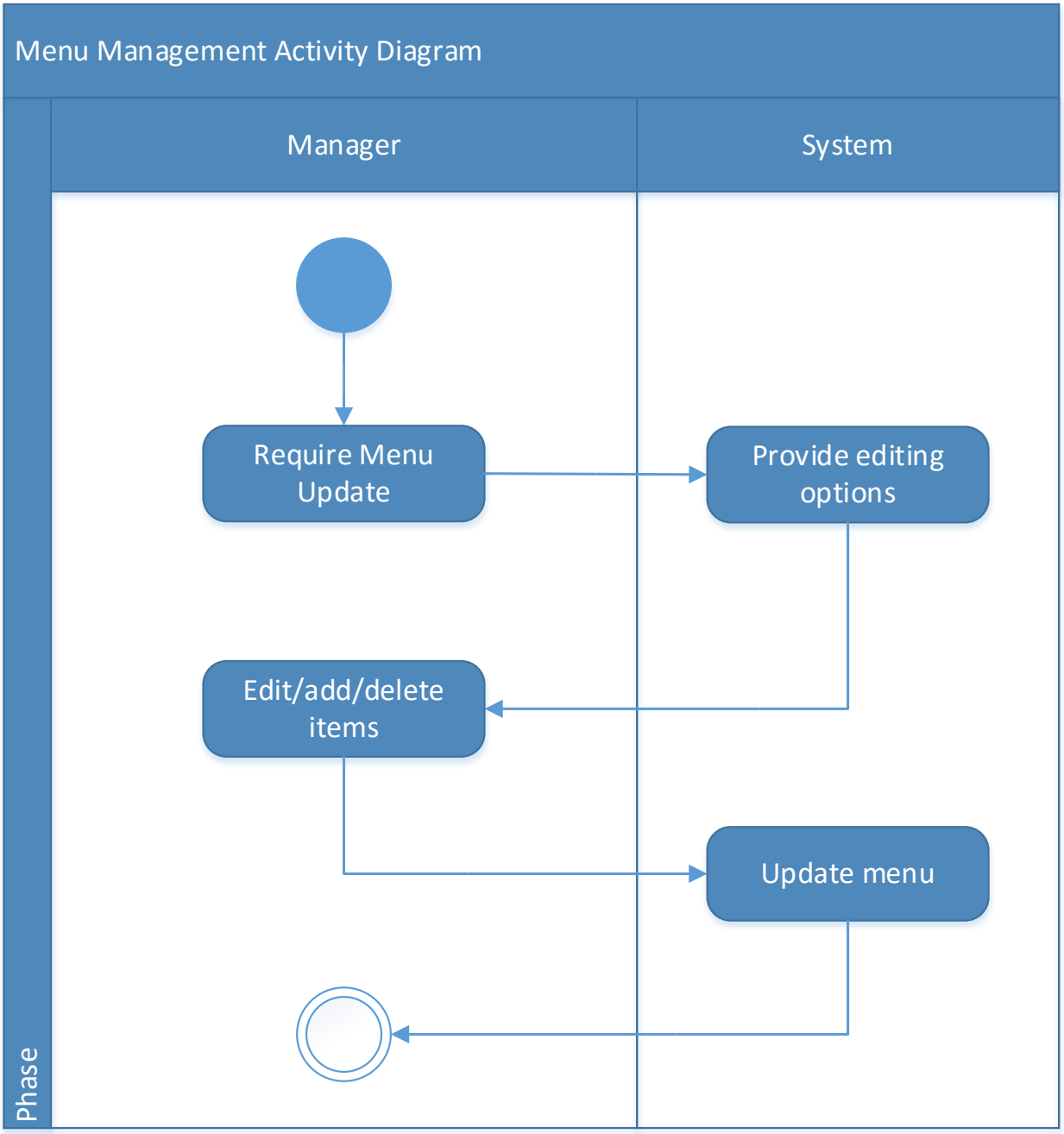

Use-case description: Menu management subsystem

| Use case | Brief use case description |
|---|---|
| Add menu items | Manager upload the new items' pictures to the system, and edit the description and price of the new items |
| Modify menu items | Manager modifies the exist items' price, pictures and description |
| Delete menu items | Manager deletes the exist items |
| View Menu | System display the new menu and manager views the updated menu |

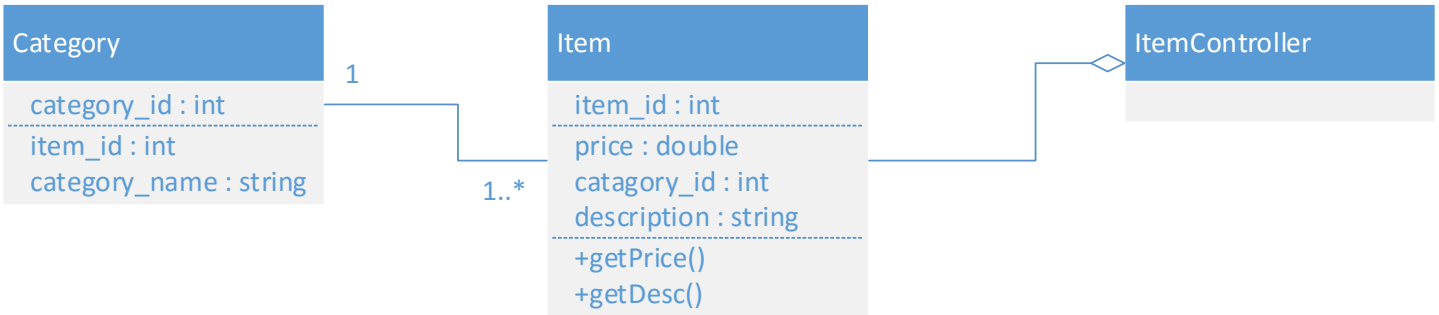
Workflow description: Menu management workflow

5. Menu Management Workflow
5.1. Manager requires menu update.
5.2. System provides different editing options (categories, items)
5.3. Manager edits/adds/deletes items.
5.4. System updates menu.

Menu management activity diagram




Design class diagram: Menu management



Technology tools for software development

At the restaurant, customers are able to order food from a menu which contains pictures, descriptions and price of the food from an Apple iPad Device, running iOS 10. Visual Studio 2015 Enterprise Edition with the Xamarin iOS Toolset will be used to develop the menu system container which will isolate the web application to view only the menu system. The software system will be created using Microsoft Office for documents, Microsoft Visio for charts and diagrams, Microsoft Project to track project progression using a Gantt chart, and Visual Studio Enterprise 2015 with Xamarin Toolset for development. The interfaces will be created with Adobe Experience Design and Adobe Photoshop CC. Microsoft PowerPoint will be used to present the work.

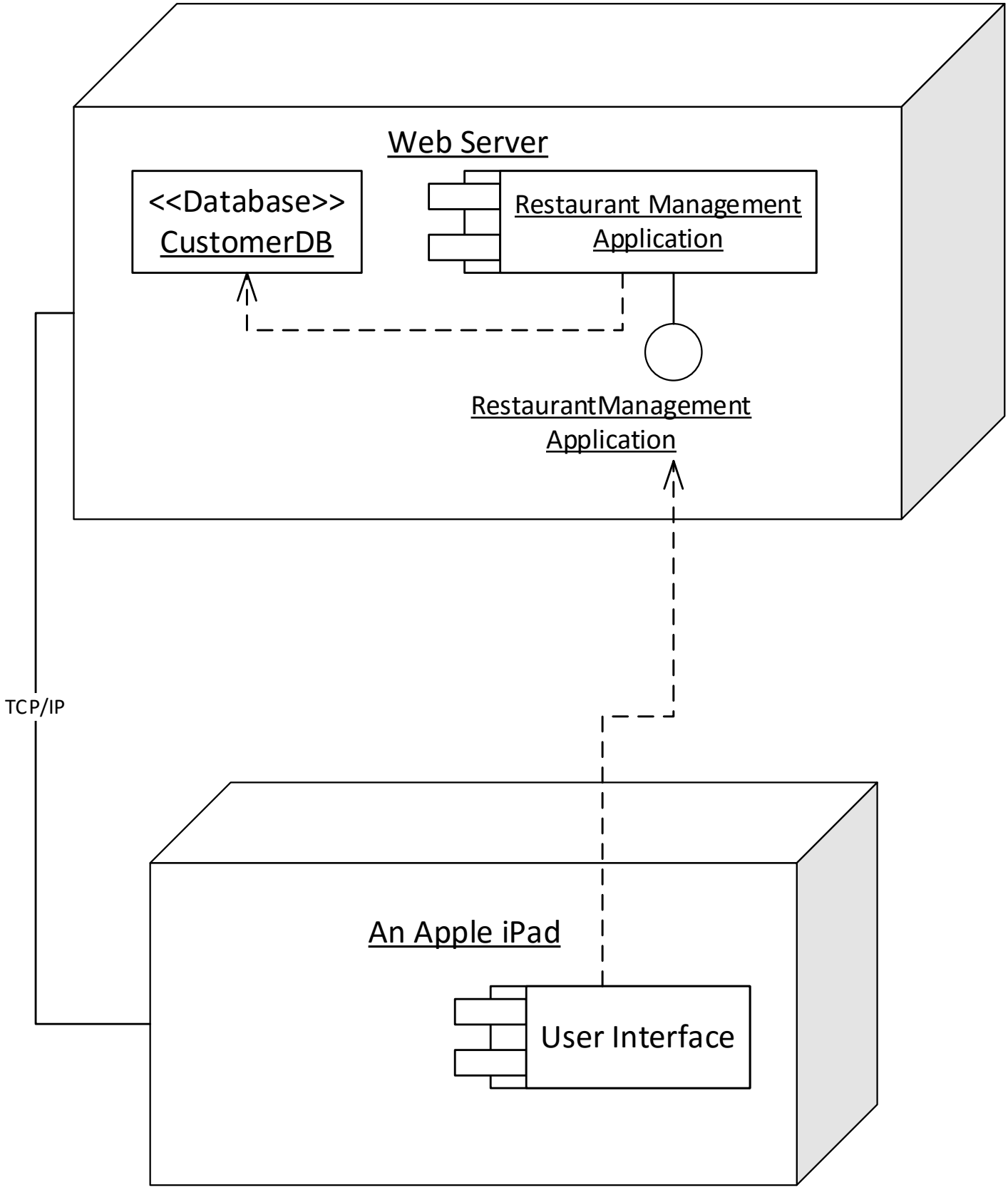
The web application (back-end system) will be programmed using the C# and ASP.NET programming languages, running on the .NET Framework CLR version 4.5.2, served on Microsoft IIS (Internet Information Services) Web Server Version 10 running on Microsoft Azure Cloud located in their Canadian data centre. The database system will use Microsoft SQL Server, with hourly backups. Backup files will be retained for 15 days. The Microsoft Azure Cloud account will be managed via Microsoft Azure Portal. All files will be published to Github and be automatically pushed to the Microsoft Azure Cloud upon file approval.

To keep data transfer fast and secure, the Apple iPads and desktop workstations will be connected to the restaurant's router, using CAT-6 networking cable using HTTP/2 and IPv6 protocols. Internet connectivity provided by Cogent Communications Canada via fibre optic cable. The Cisco internet router will be cable of electronically and programmatically isolating the activity from restaurant ordering and management functions and public wireless internet offered to restaurant customers, allowing such operations as checking E-mail, publishing images to their food blog, etc.

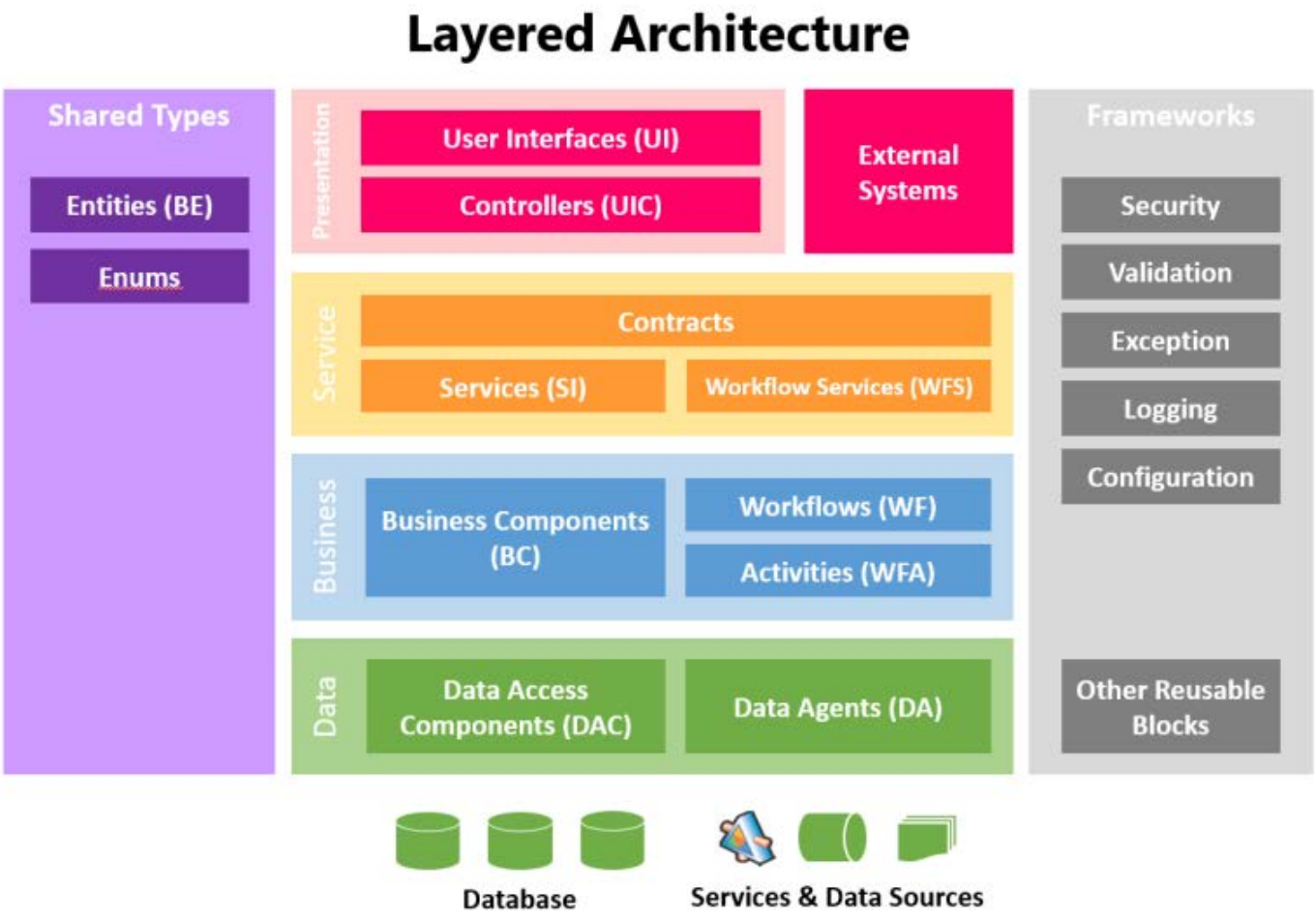
Management functions will be conducted using PC desktop workstations running Microsoft Windows 10, purchased from ASUS, running the unrestricted web application in Google Chrome Web-browser.

The kitchen and wait staff will use the same desktop hardware as the management setup, but instead have restricted menus in the web application which show current, upcoming orders and other relevant information at-a-glance, with the addition of an order advance button to allow for quick operation. In the kitchen, the desktop monitors will be mounted to the walls.

Component and deployment diagram



High-level, layered architecture diagram



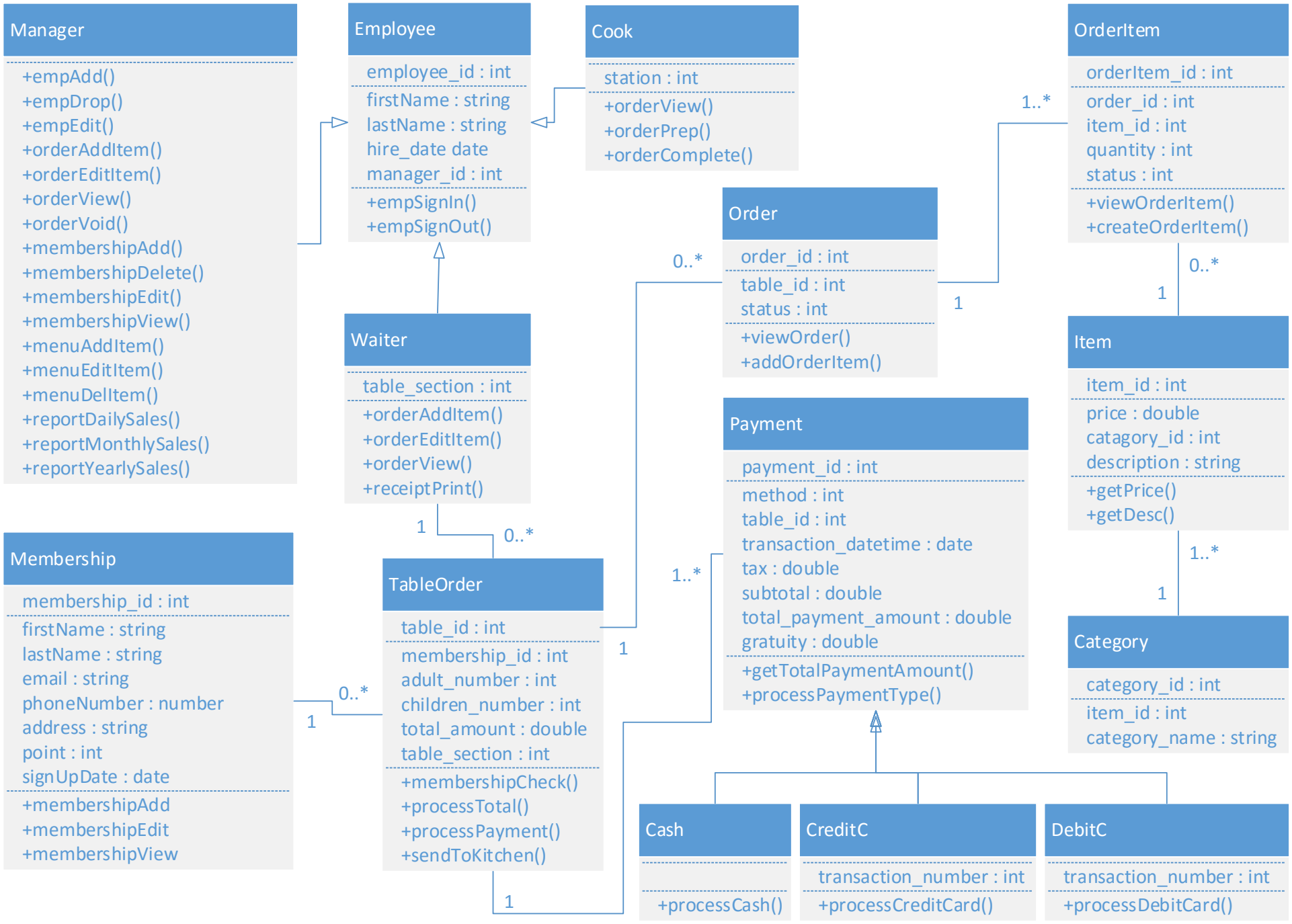
Domain class diagram

Class Responsibility Collaboration (CRC) cards, with function names

| Class | Responsibility | Function Name | Collaboration |
|---|---|---|---|
| Employee | Employee sign-in | +empSignIn | Manager |
| | Employee sign-out | +empSignOut | Waiter |
| | | | Cook |
| Manager | Add employee | +empAdd | Employee |
| | Drop employee | +empDrop | Waiter |
| | Edit employee | +empEdit | Cook |
| | Add items to order | +orderAddItem | TableOrder |
| | Edit items to order | +orderEditItem | Membership |
| | View order | +orderView | |
| | Void order | +orderVoid | |
| | Add membership | +membershipAdd | |
| | Delete membership | +membershipDelete | |
| | Edit membership details | +membershipEdit | |
| | View membership details | +membershipView | |
| | Add menu item | +menuAddItem | |
| | Edit menu item | +menuEditItem | |
| | Delete menu item | +menuDelItem | |
| | View daily sales report | +reportDailySales | |
| | View monthly sales report | +reportMonthlySales | |
| | View yearly sales report | +reportYearlySales | |
| Waiter | Add items to order | +orderAddItem | Employee |
| | Edit items to order | +orderEditItem | TableOrder |
| | View order | +orderView | |
| | Print customer receipt | +receiptPrint | |
| Cook | View order | +orderView | Employee |
| | Prepare order | +orderPrep | TableOrder |
| | Complete order | +orderComplete | |
| Membership | Add membership details | +membershipAdd | TableOrder |
| | Edit membership details | +membershipEdit | |
| | View membership details | +membershipView | |
| TableOrder | Check membership | +membershipCheck | Membership |
| | Calculate total amount | +processTotal | Waiter |
| | Process payment | +processPayment | Order |

| | | | |
|---|---|---|---|
| | Send order to kitchen | +sendToKitchen | Payment |
| Order | View order | +viewOrder | TableOrder |
| | Add item to order | +addOrderItem | OrderItem |
| OrderItem | View order item | +viewOrderItem | Item |
| | Create order item | +createOrderItem | Order |
| Item | Get item price | +getPrice | OrderItem |
| | Get item description | +getDesc | Category |
| Payment | Get total amount to pay | +getTotalPaymentAmount | TableOrder |
| | Select payment type | +processPaymentType | |
| Cash | Get cash paid amount | +processPaymentCash | Payment |
| CreditC | Get credit card paid amount | +processPaymentCreditCard | Payment |
| DebitC | Get debit card paid amount | +processPaymentDebitCard | Payment |

Design Class Diagram



C# Skeletal code file: Employee.cs

```
using System;
namespace GoldenDragonHotPotHouseRestaurantSystem
{
   class Employee
   {
      private int employee_id;
      private string firstName;
      private string lastName;
      private string password;
      private DateTime hire_date;
      private int manager_id;
      public Employee()
      {
         // Class constuctor logic here.
      }
      public void empSignIn(string firstName, string lastName, string password)
      {
         // Login logic here.
      }
      public void empSignOut()
      {
         // Logout logic here.
      }
      public bool empIsLoggedIn()
      {
         // Authentication check logic here.
         return true;
      }
      public int empType()
      {
         // Return employee role logic here.
         return employeeRoleType;
      }
   }
   class Waiter : Employee
   {
      private int table_section;
```

```
    private void orderAddItem(int order_id, int item_id)
    {
        // Add item to order logic here.
    }
    private void orderEditItem(int order_id, int item_id)
    {
        // Edit item in order logic here.
    }
    private void orderView(int order_id)
    {
        // View order logic here.
    }
    private void receiptPrint(int order_id, int receipt_type)
    {
        // Print receipt logic here.
    }
}
class Cook : Employee
{
    private int station;
    private void orderView(int order_id)
    {
        // View order on kitchen display logic here.
    }
    private void orderPrep(int order_id)
    {
        // Prep order on kitchen display logic here.
    }
    private void orderComplete(int order_id)
    {
        // Complete the order on kitchen display logic here.
    }
}
class Manager : Employee
{
    private void empAdd(string firstName, string lastName, string password)
    {
        // Add an employee logic here.
```

```
}
private void empDrop(int employee_id, int employee_drop_code, string drop_remarks)
{
   // Drop/delete an employee logic here.
}
private void empEdit(int employee_id)
{
   // Edit an employee logic here.
}
private void orderAddItem(int order_id)
{
   // Manager: Add an order item logic here.
}
private void orderEditItem(int order_id)
{
   // Manager: Edit an order item logic here.
}
private void orderView(int order_id)
{
   // Manager: View an order logic here.
}
private void orderVoid(int order_id, string order_void_reason)
{
   // Manager: Void an order logic here.
}
private void membershipAdd(string firstName, string lastName, string email, decimal phoneNumber, int point)
{
   // Manager: Manually add a membership logic here.
}
private void membershipDelete(int membership_id)
{
   // Manager: Manually delete a membership logic here.
}
private void membershipEdit(int membership_id)
{
   // Manager: Manually edit membership details logic here.
}
private void membershipView(int membership_id)
```

```
        {
            // Manager: View membership details logic here.
        }
        private void menuAddItem(string menu_item_name, string menu_item_desc)
        {
            // Add an item to the restaurant menu logic here.
        }
        private void menuEditItem(int menu_item_id)
        {
            // Edit an item on the restaurant menu logic here.
        }
        private void menuDelItem(int menu_item_id)
        {
            // Delete an item on the restaurant menu logic here.
        }
        private void reportDailySales(int report_detail_style)
        {
            // Display daily sales report logic here.
        }
        private void reportMonthlySales(int report_detail_style)
        {
            // Display monthly sales report logic here.
        }
        private void reportYearlySales(int report_detail_style)
        {
            // Display year-to-date sales report logic here.
        }
    }
}
```

C# Skeletal code file: Membership.cs

```
using System;

namespace GoldenDragonHotPotHouseRestaurantSystem
{
    class Membership
    {
        private int membership_id;
```

```
        private string firstName;
        private string lastName;
        private string email;
        private decimal phoneNumber;
        private string address;
        private int point;
        private string signUpDate;
        // Get and set properties
        public int Membership_id { get; }
        public Membership(string firstName, string lastName, string email, decimal phoneNumber, string address, int point)
        {
            // Class constructor logic here.
        }
        private void membershipAdd(string firstName, string lastName, string email, decimal phoneNumber, string address, int point)
        {
            // Add a membership logic here.
        }
        private void membershipDel(int membership_id, string membership_deletion_reason)
        {
            // Add a membership logic here.
        }
        private void membershipEdit(int membership_id)
        {
            // Edit a membership logic here.
        }
        private void membershipView(int membership_id)
        {
            // View a membership logic here.
        }
    }
}
```

C# Skeletal code file: Payment.cs

```
using System;

namespace GoldenDragonHotPotHouseRestaurantSystem
{
    class Payment
```

```
{
   private int method;
   private int table_id;
   private string transaction_datetime;
   private decimal tax;
   private decimal subtotal;
   private decimal gratuity;
   private decimal total_payment_amount;
   private int transactionNumber;
   // Member variable Get and set properties logic here.
   protected int TransactionNumber { get; set; }
   protected decimal Total_payment_amount { get; }
   private void getTotalPaymentAmount(int order_id)
   {
      // Calculate the total payment amount logic here.
   }
   private void processPaymentType(int method)
   {
      // Send total payment amount to inherited class logic here.
   }
}
class Cash : Payment
{
   private int processCash(decimal total_payment_amount, int transactionNumber)
   {
      // Process cash payment here.
      return transactionNumber;
   }
}
class CreditC : Payment
{
   private int processCreditCard(decimal total_payment_amount, int transactionNumber)
   {
      // Process credit card payment here.
      return transactionNumber;
   }
}
class DebitC : Payment
```

```
    {
        private int processDebitCard(decimal total_payment_amount, int transactionNumber)
        {
            // Process debit card payment here.
            return transactionNumber;
        }
    }
}
```

C# Skeletal code file: TableOrder.cs

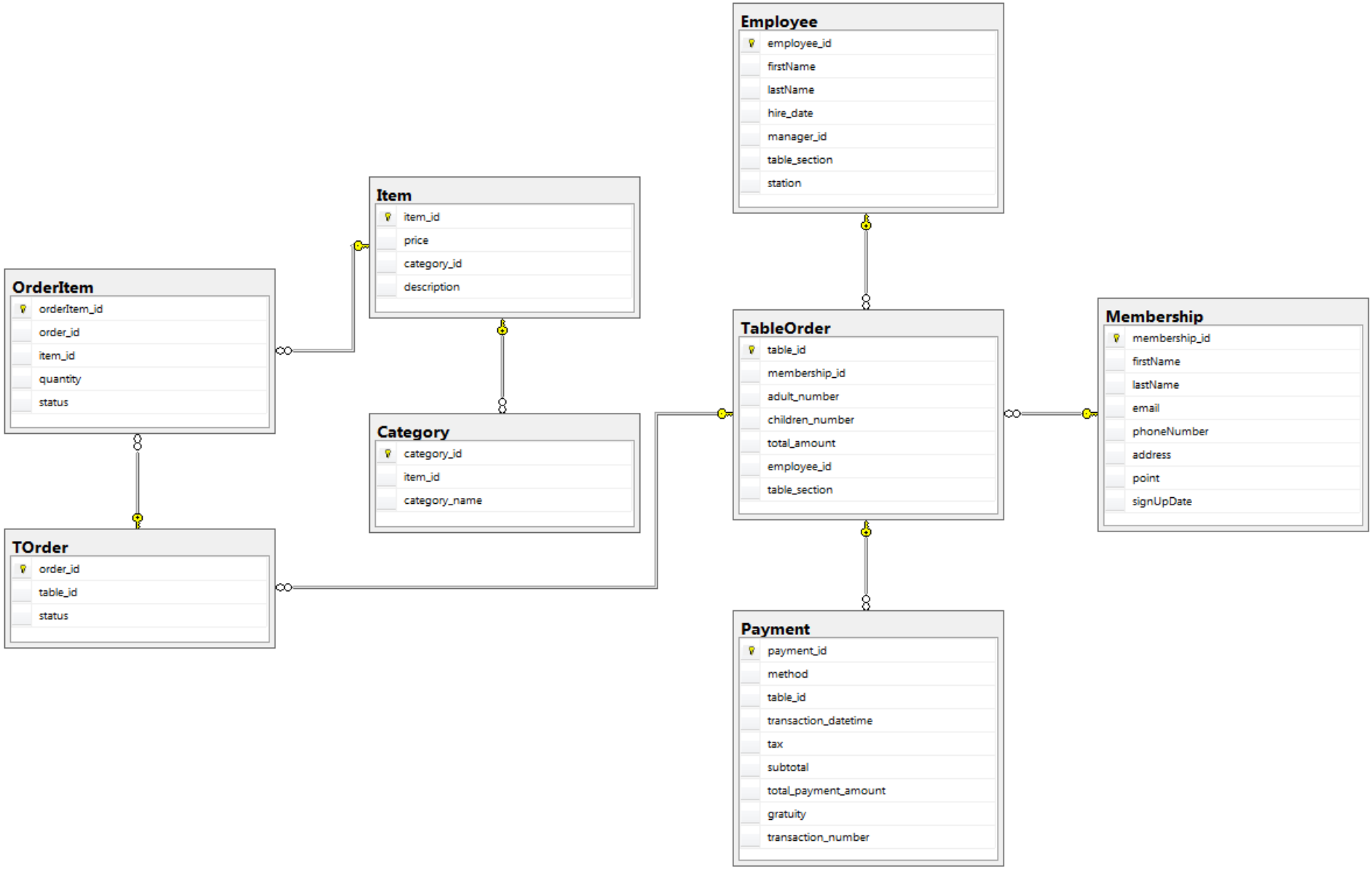
```
using System;
namespace GoldenDragonHotPotHouseRestaurantSystem
{
    class TableOrder
    {
        private int order_id;
        private int table_id;
        private int membership_id;
        private int adult_number;
        private int children_number;
        private decimal total_amount;
        private int table_section;
        // Get and Set properties logic here
        public int Order_id { get; }
        public int Table_id { get; }
        public int Membership_id { get; set; }
        public int Adult_number { get; }
        public int Children_number { get; }
        public decimal Total_amount { get; set; }
        public int Table_section { get; set; }
        // Methods
        private void membershipCheck(int membership_id)
        {
            // Check for membership information logic here.
        }
        private void processTotal(int order_id)
        {
            // Process total amount and calls to payment class logic here.
```

```
    }
    private void processPayment(int order_id)
    {
        // Process payment and calls to payment class logic here.
    }
    private void sentToKitchen(int order_id)
    {
        // Put order onto the kitchen displays logic here.
    }
    private void feedbackSubmit(int feedback_rating, string feedback_desc)
    {
        // Submit feedback logic here.
    }
    private void feedbackRetraction(int feedback_id)
    {
        // Retract feedback logic here.
    }
    private void feedbackEdit(int feedback_id)
    {
        // Edit feedback logic here.
    }
}
class Order : TableOrder
{
    private int status;
    // Methods
    private void viewOrder(int order_id)
    {
        // View order logic here.
    }
    private void addOrderItem(int order_id, int item_id)
    {
        // View order logic here.
    }
}
class OrderItem : Order
{
    private int orderitem_id;
```

```
    private int item_id;
    private int quantity;
    // Get and Set properties logic here.
    public int Orderitem_id { get; set; }
    public int Item_id { get; set; }
    public int Quantity { get; set; }
    // Methods
    private void viewOrderItem()
    {
        // View order item logic here.
    }
    private void createOrderItem()
    {
        // Create order item logic here.
    }
}
class Item : OrderItem
{
    private int item_id;
    private decimal price;
    private int category_id;
    private string description;
    // Get and Set properties logic here.
    public decimal Price { get; set; }
    public string Description { get; }
    // Methods
    private decimal getPrice(int item_id)
    {
        // Get item price logic here.
        return item_price;
    }
    private string getDesc(int item_id)
    {
        // Get item description logic here.
        return item_description;
    }
}
class Category : Item
```

```
  {
    private int category_id;
    private string category_name;
    // Get and Set properties
    public int Category_id { get; }
    public string Category_name { get; }
    // Method
    private void addCategory()
    {
      // Add a category logic here.
    }
    private void displayCategories()
    {
      // Display categories logic here.
    }
  }
}
```

Entity Relationship Diagram (ERD)

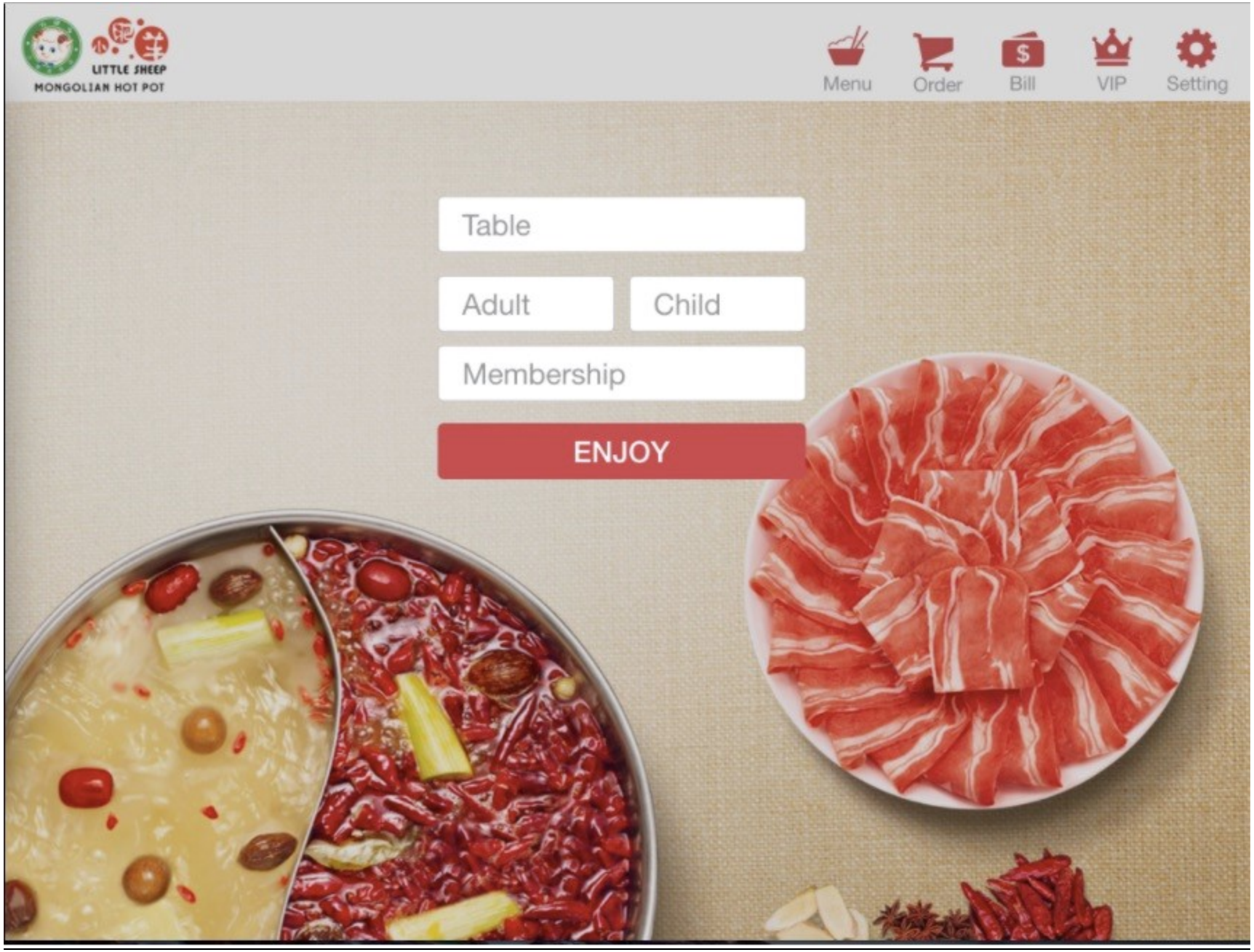


Transact-SQL (T-SQL) database code

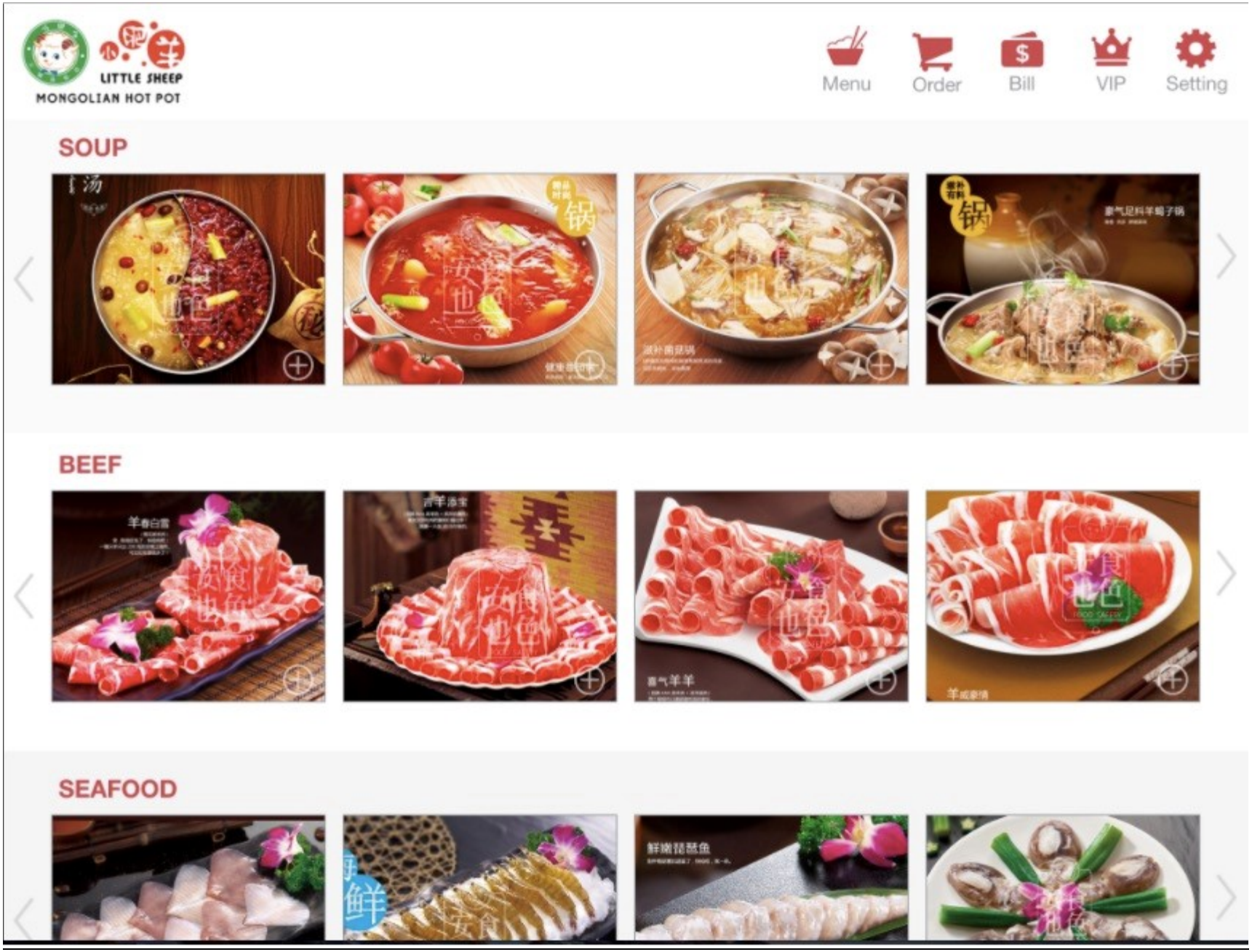
```
/* Written in Transact-SQL for MS SQL Server/Azure - Using default dbo schema */
CREATE TABLE [dbo].[Employee] (
	[employee_id] int NOT NULL IDENTITY(1,1) PRIMARY KEY,
	[firstName] nvarchar(50) NOT NULL,
	[lastName] nvarchar(50) NOT NULL,
	[hire_date] date NOT NULL,
	[manager_id] int NOT NULL,
	[table_section] int NOT NULL,
	[station] int NOT NULL
	);
CREATE TABLE [dbo].[TableOrder] (
	[table_id] int NOT NULL IDENTITY(1,1) PRIMARY KEY,
	[membership_id] int NOT NULL,
	[adult_number] int NOT NULL,
	[children_number] int NOT NULL,
	[total_amount] money NOT NULL,
	[employee_id] int NOT NULL,
	[table_section] int NOT NULL
	);
CREATE TABLE [dbo].[TOrder] (
	[order_id] int NOT NULL IDENTITY(1,1) PRIMARY KEY,
	[table_id] int NOT NULL,
	[status] int NOT NULL
	);
CREATE TABLE [dbo].[Membership] (
	[membership_id] int NOT NULL IDENTITY(1,1) PRIMARY KEY,
	[firstName] nvarchar(50) NOT NULL,
	[lastName] nvarchar(50) NOT NULL,
	[email] nvarchar(255) NOT NULL,
	[phoneNumber] nvarchar(35) NOT NULL,
	[address] nvarchar(255) NOT NULL,
	[point] int NOT NULL,
	[signUpDate] date NOT NULL
	);
CREATE TABLE [dbo].[OrderItem] (
```

```
	[orderItem_id] int NOT NULL IDENTITY(1,1) PRIMARY KEY,
	[order_id] int NOT NULL,
	[item_id] int NOT NULL,
	[quantity] int NOT NULL,
	[status] int NOT NULL
	);
CREATE TABLE [dbo].[Item] (
	[item_id] int NOT NULL IDENTITY(1,1) PRIMARY KEY,
	[price] money NOT NULL,
	[category_id] int NOT NULL,
	[description] nvarchar(255) NOT NULL
	);
CREATE TABLE [dbo].[Category] (
	[category_id] int NOT NULL IDENTITY(1,1) PRIMARY KEY,
	[item_id] int NOT NULL,
	[category_name] nvarchar(50) NOT NULL
	);
CREATE TABLE [dbo].[Payment] (
	[payment_id] int NOT NULL IDENTITY(1,1) PRIMARY KEY,
	[method] int NOT NULL,
	[table_id] int NOT NULL,
	[transaction_datetime] date NOT NULL,
	[tax] money NOT NULL,
	[subtotal] money NOT NULL,
	[total_payment_amount] money NOT NULL,
	[gratuity] money NOT NULL,
	[transaction_number] int NOT NULL
	);
/* Add Foreign Key Constraints */
ALTER TABLE [dbo].[TableOrder] ADD CONSTRAINT FK_membership_id FOREIGN KEY (membership_id) REFERENCES dbo.Membership (membership_id);
ALTER TABLE [dbo].[TableOrder] ADD CONSTRAINT FK_employee_id FOREIGN KEY (employee_id) REFERENCES dbo.Employee (employee_id);
ALTER TABLE [dbo].[TOrder] ADD CONSTRAINT FK_table_id FOREIGN KEY (table_id) REFERENCES dbo.TableOrder (table_id);
ALTER TABLE [dbo].[Payment] ADD CONSTRAINT FK_table_payment_id FOREIGN KEY (table_id) REFERENCES dbo.TableOrder (table_id);
ALTER TABLE [dbo].[Item] ADD CONSTRAINT FK_item_id FOREIGN KEY (item_id) REFERENCES dbo.OrderItem (item_id);
ALTER TABLE [dbo].[OrderItem] ADD CONSTRAINT FK_order_id FOREIGN KEY (order_id) REFERENCES dbo.TOrder (order_id);
ALTER TABLE [dbo].[OrderItem] ADD CONSTRAINT FK_orderitem_id FOREIGN KEY (item_id) REFERENCES dbo.Item (item_id);
ALTER TABLE [dbo].[Category] ADD CONSTRAINT FK_item_category_id FOREIGN KEY (item_id) REFERENCES dbo.Item (item_id);
```

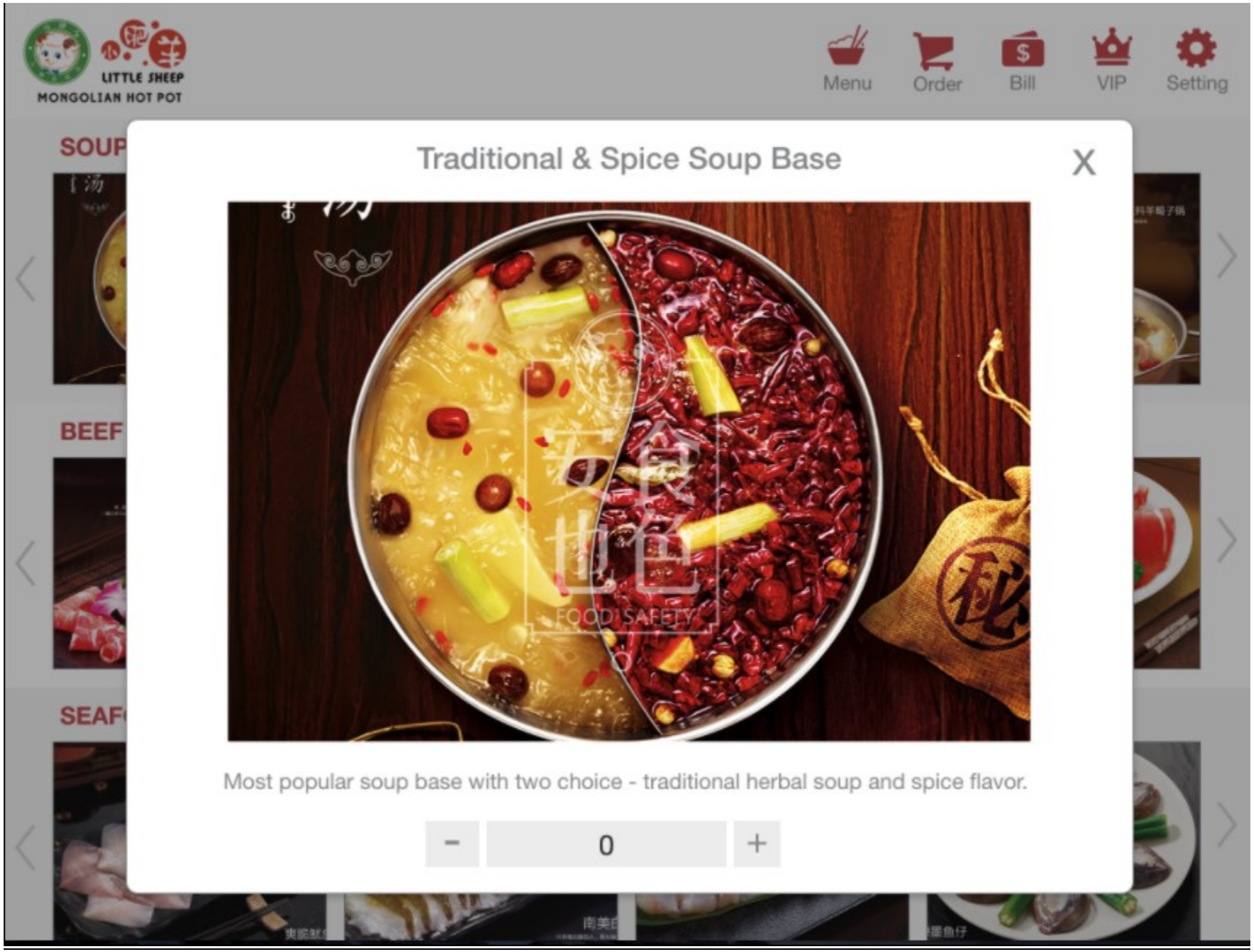
User Interface (UI) wireframe: Customer enters table information



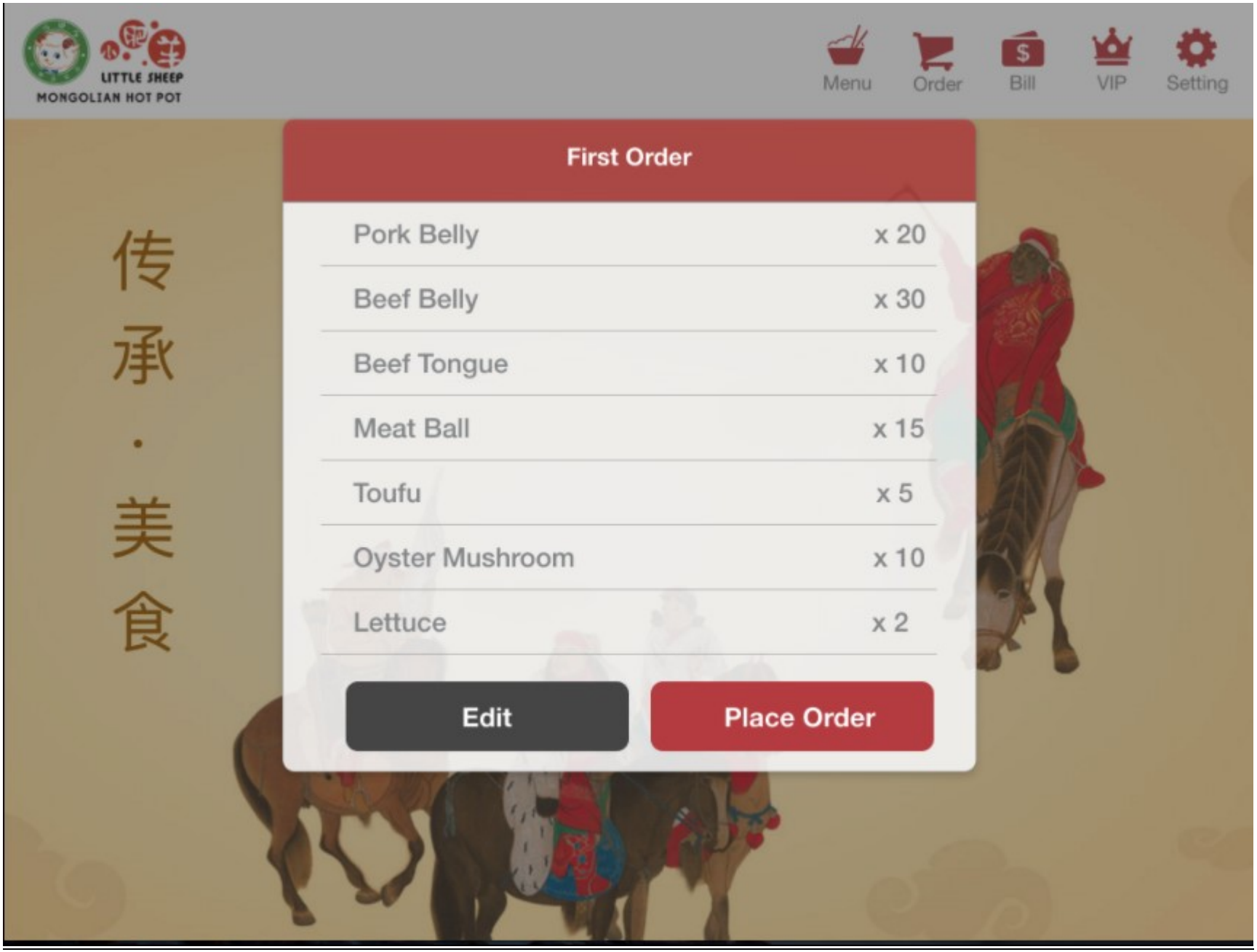
User Interface (UI) wireframe: Customer selects items



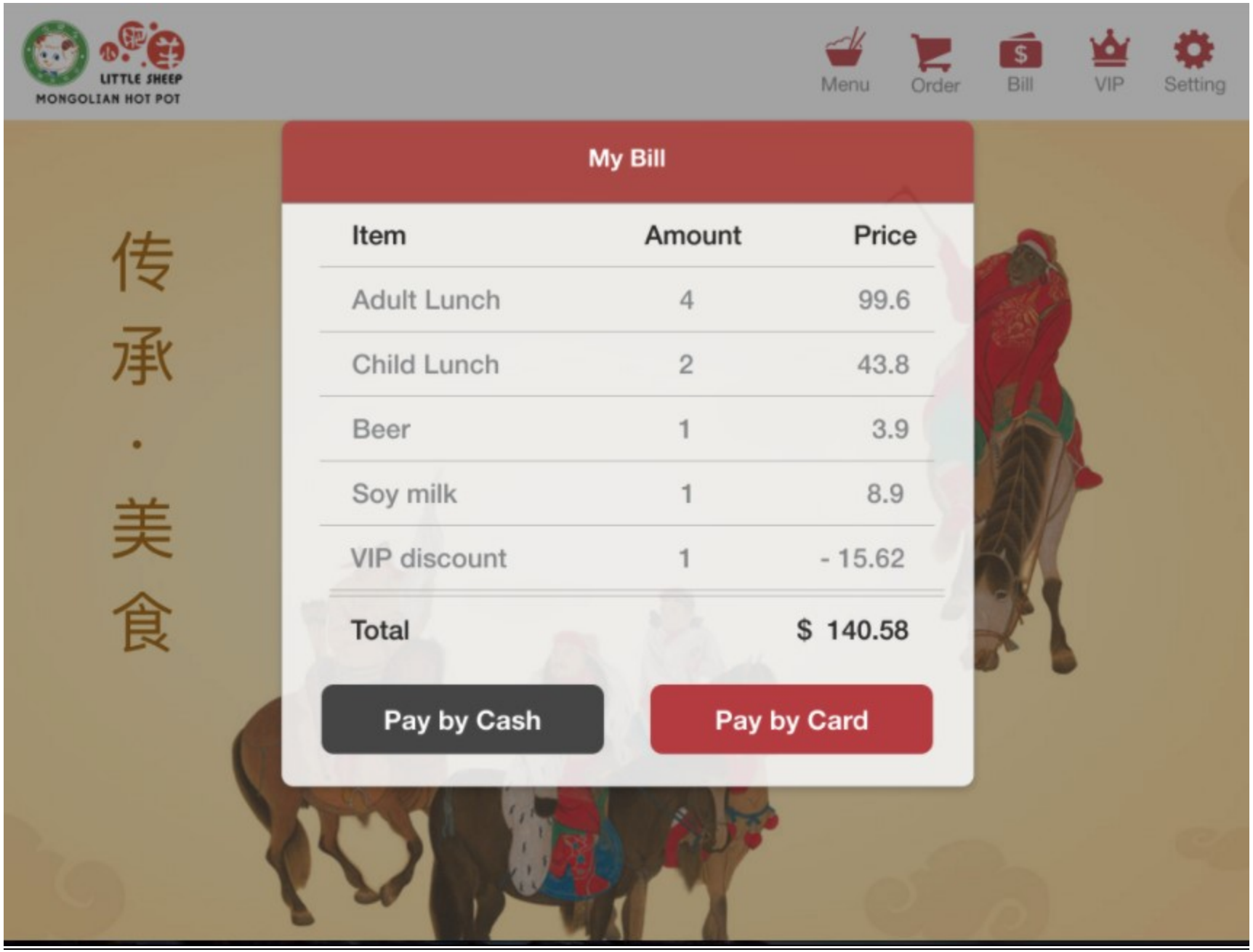
User Interface (UI) wireframe: Customer views item details



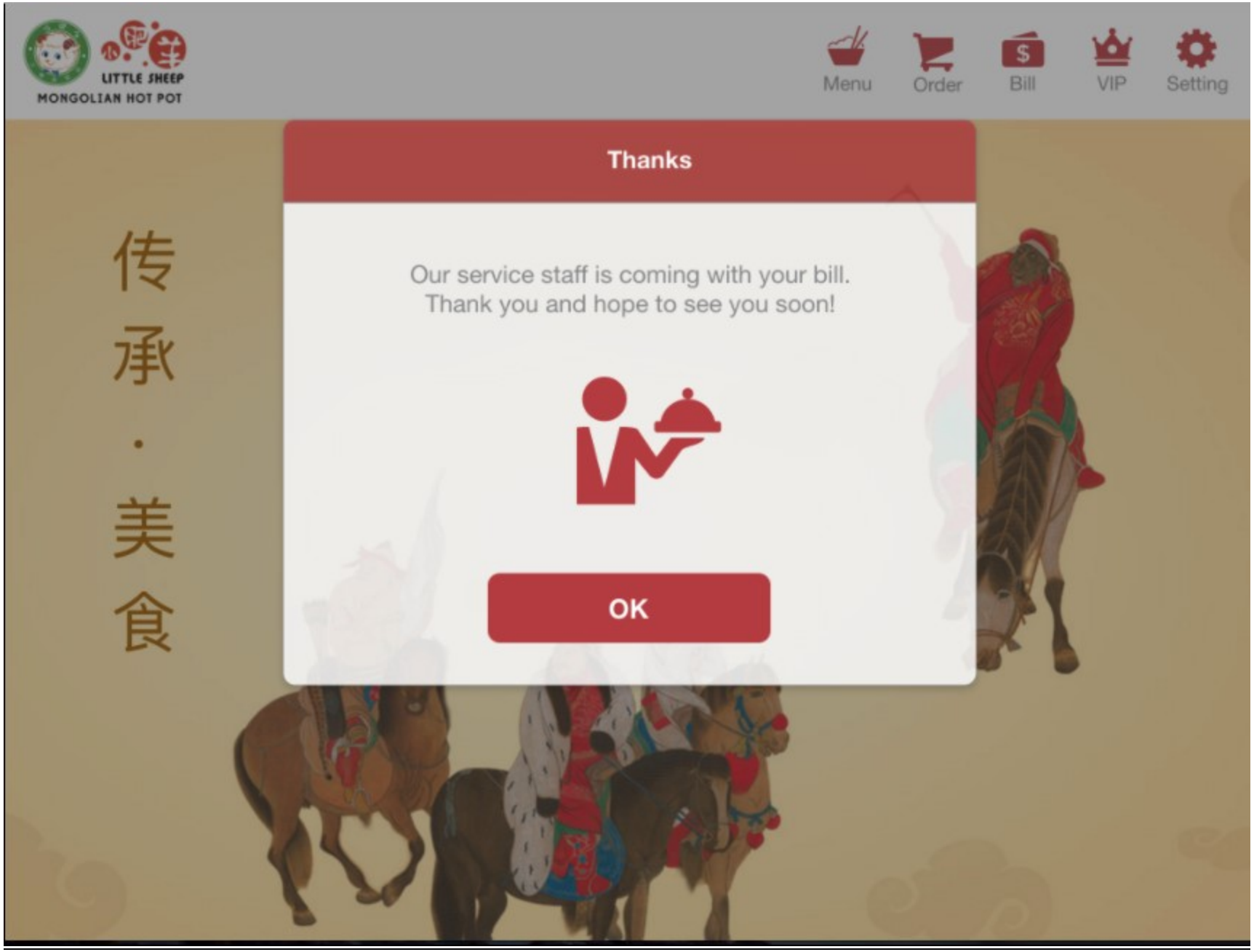
User Interface (UI) wireframe: Customer decides to edit or place order



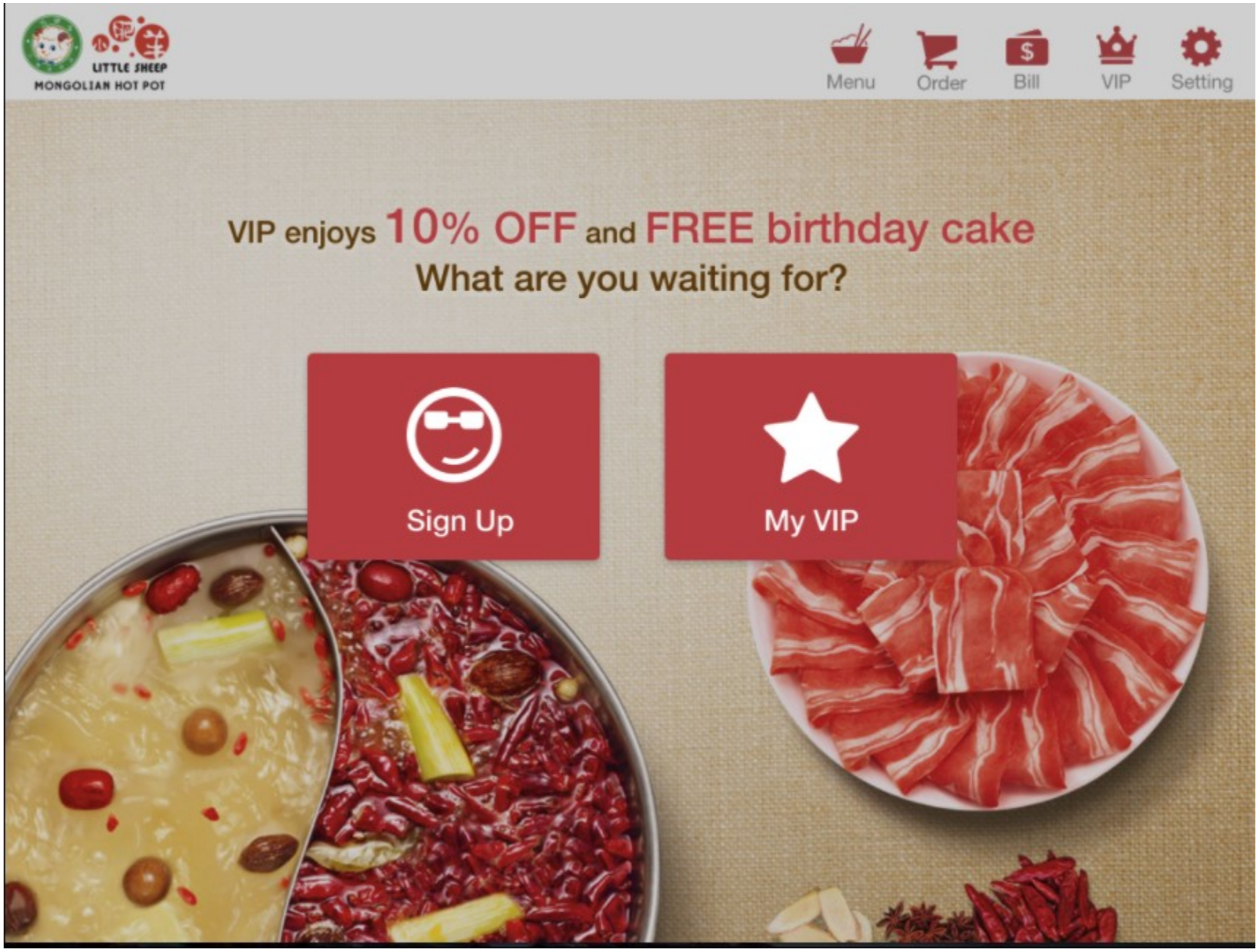
User Interface (UI) wireframe: Customer views order summary



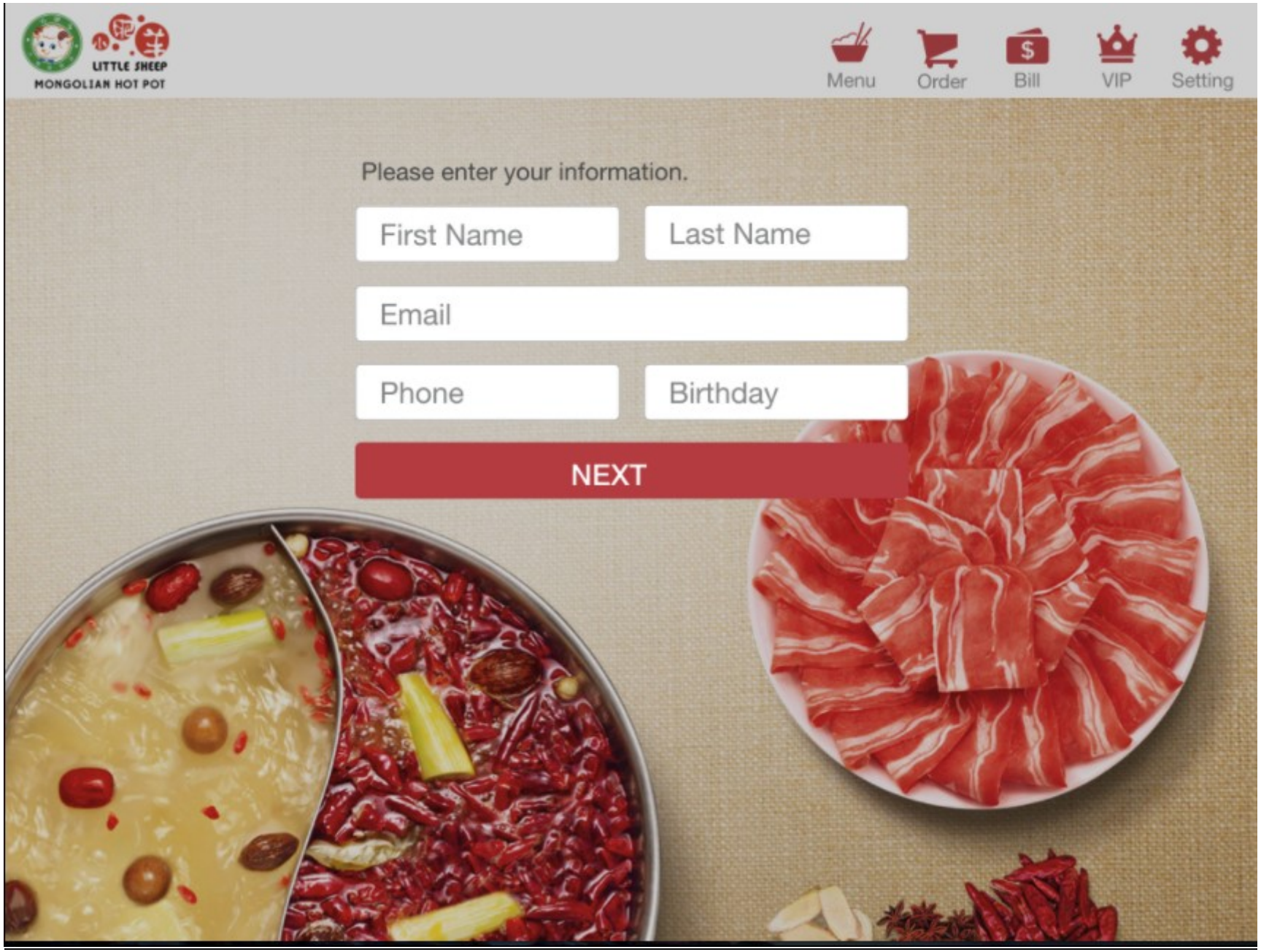
User Interface (UI) wireframe: System thanks and informs customer



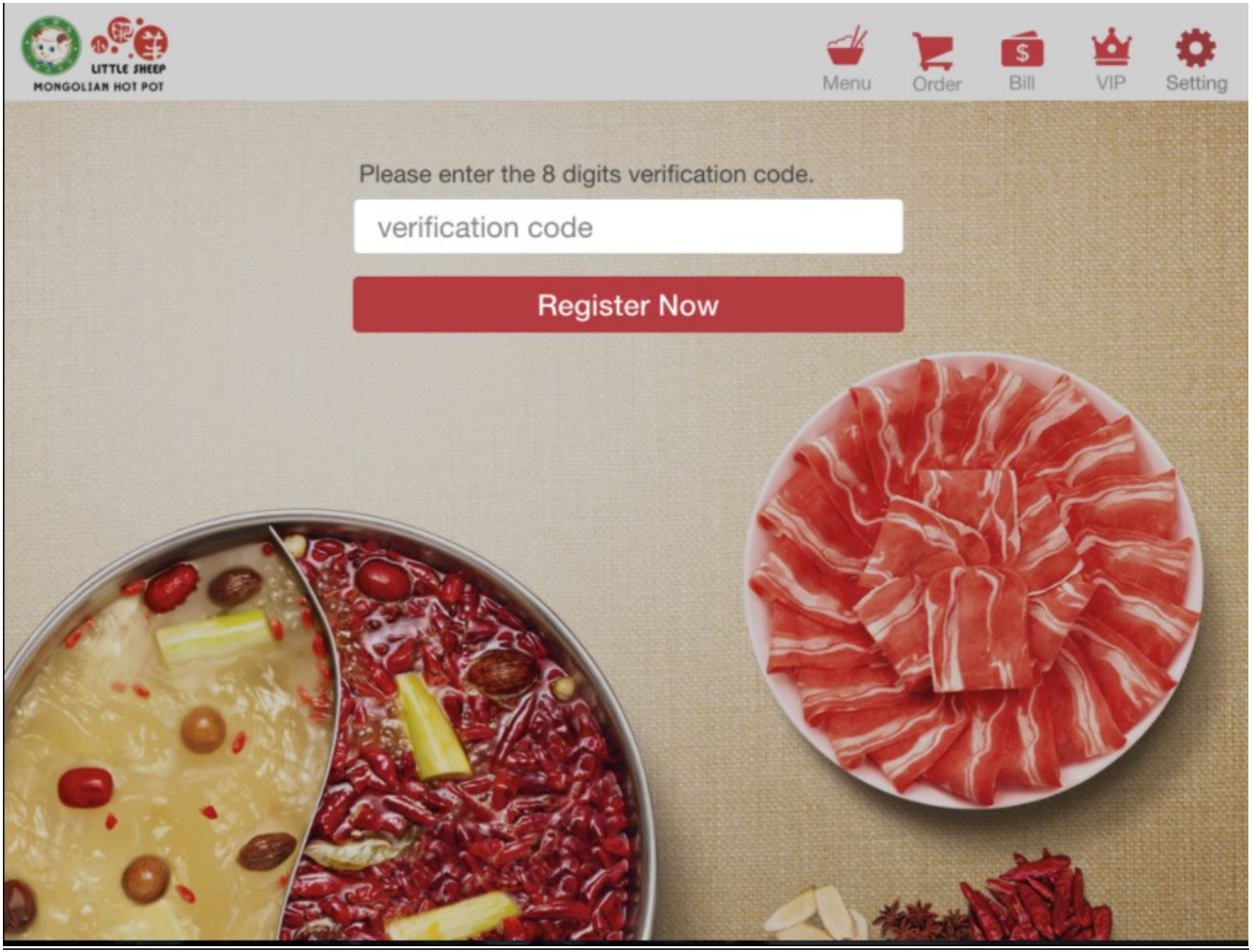
User Interface (UI) wireframe: System offers membership to customer



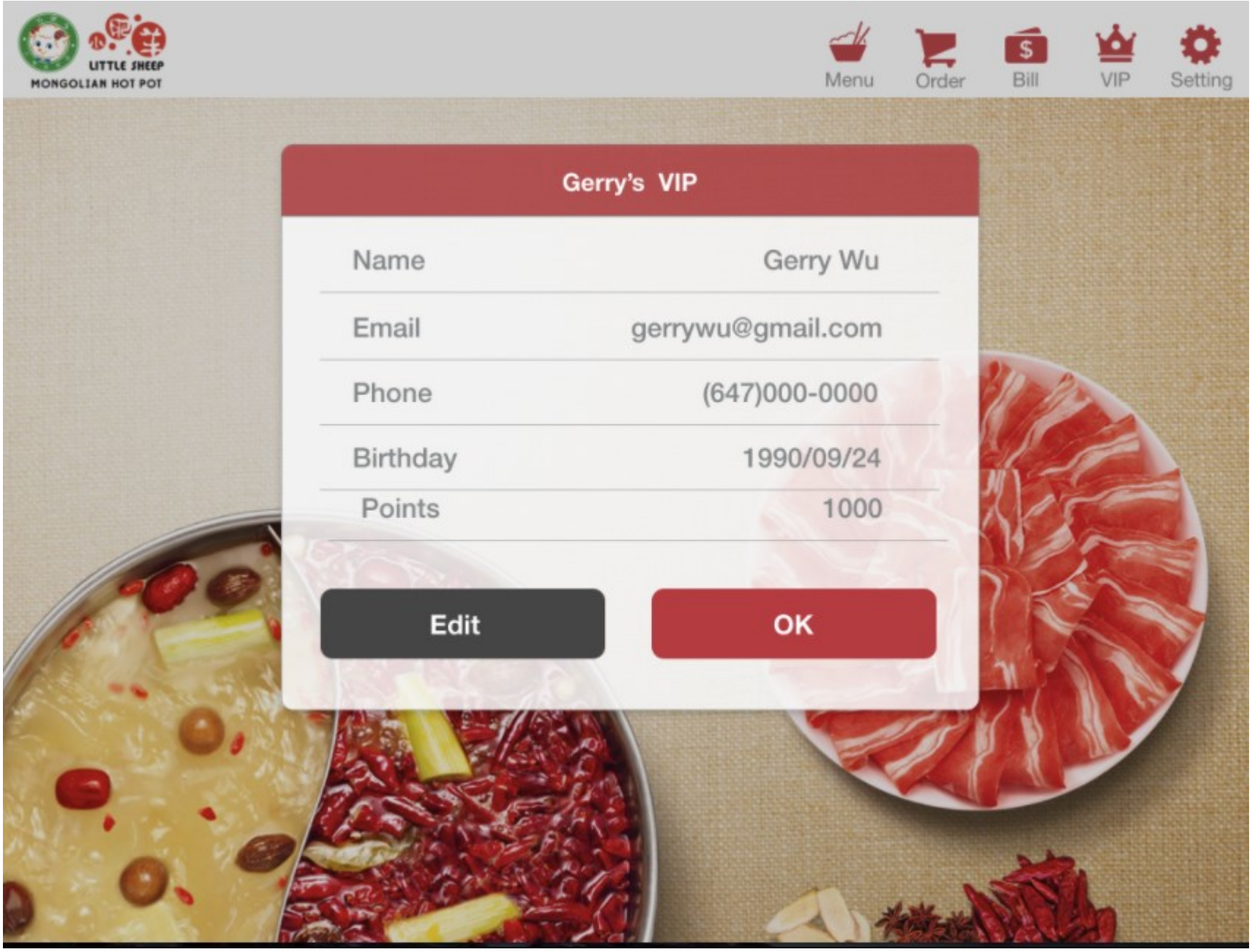
User Interface (UI) wireframe: Customer enters information for membership



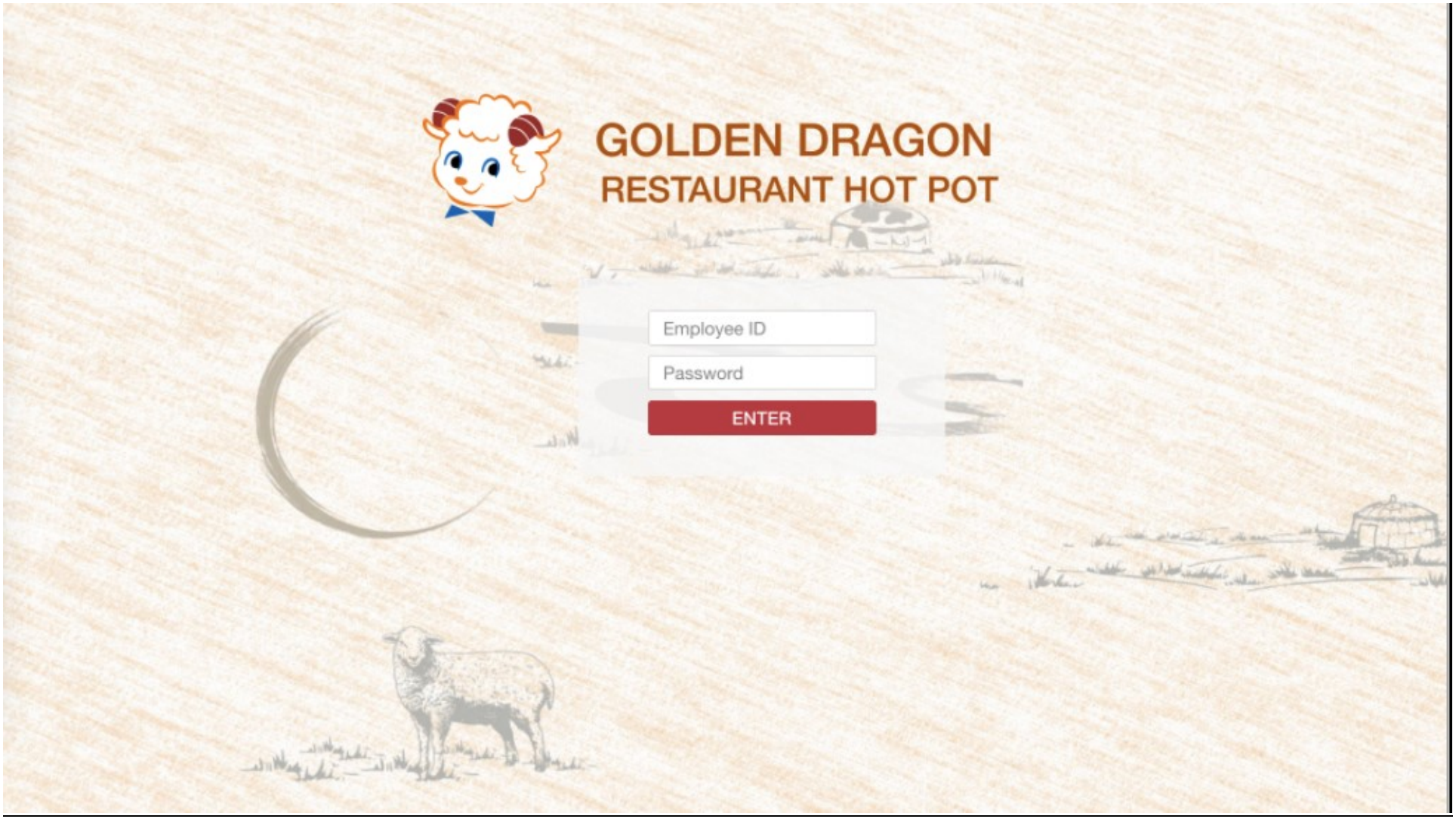
User Interface (UI) wireframe: Customer enters phone verification code



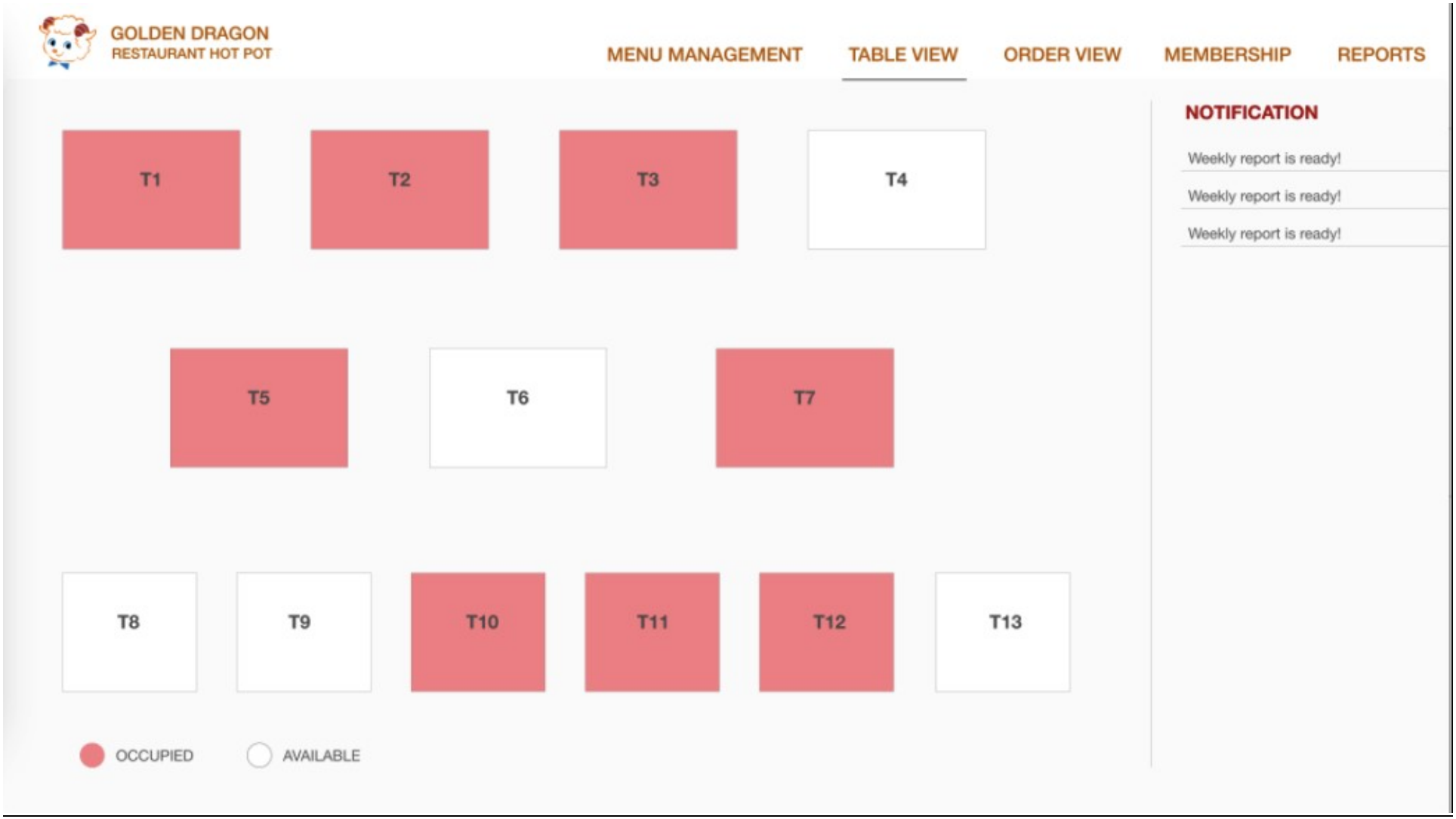
User Interface (UI) wireframe: Customer decides to edit or confirm membership



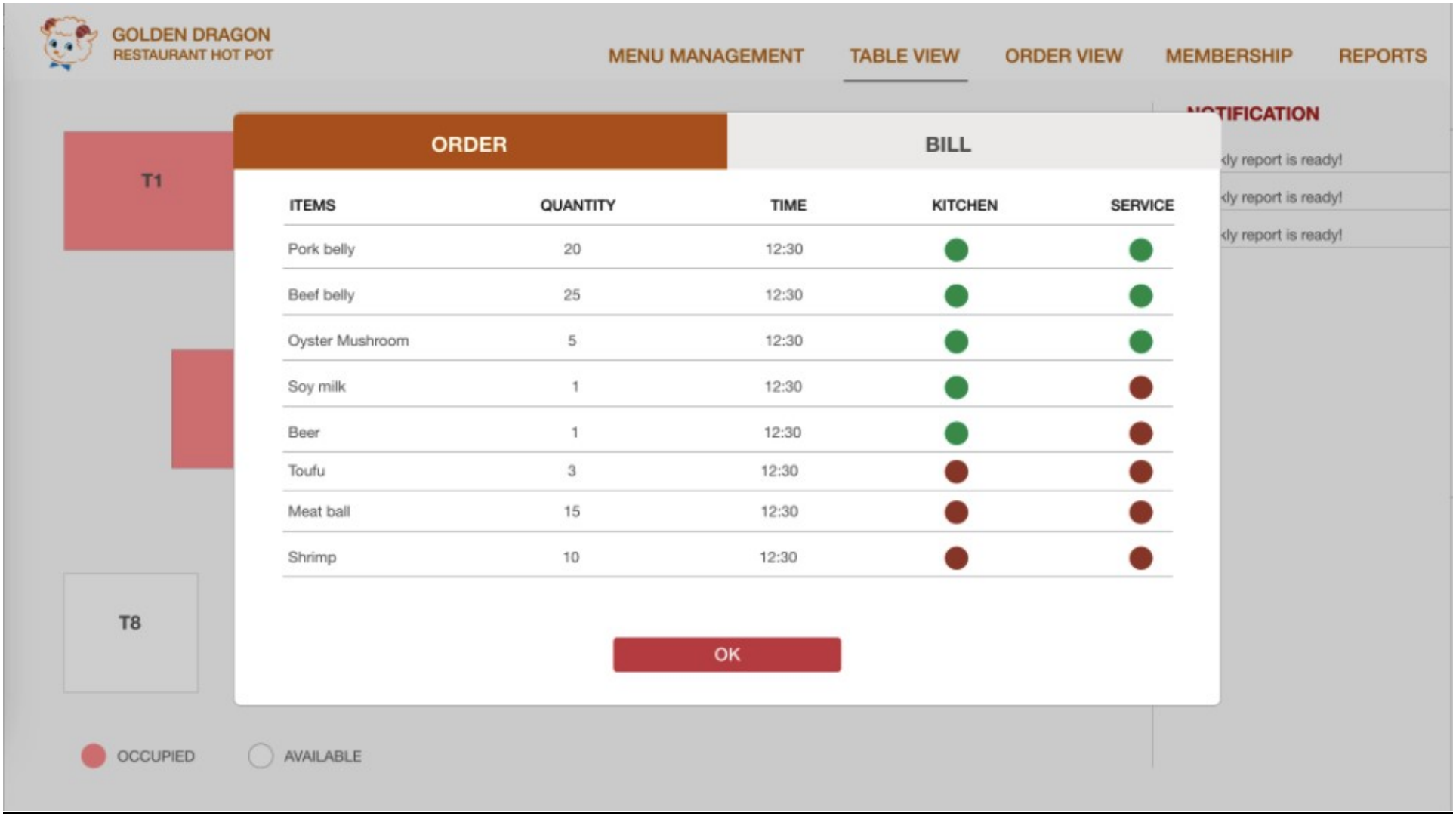
User Interface (UI) wireframe: Employee login interface



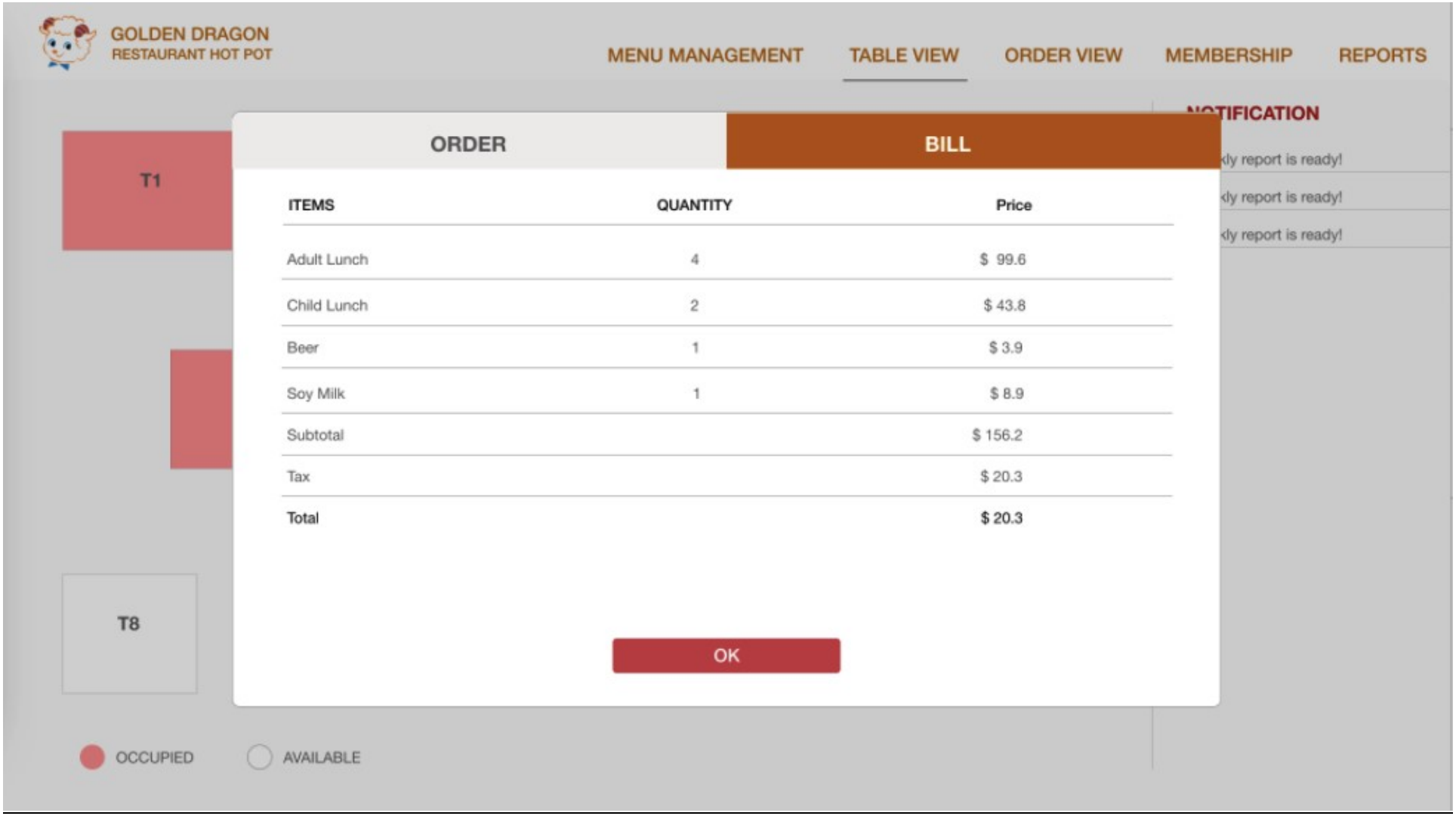
User Interface (UI) wireframe: Management screen, table view



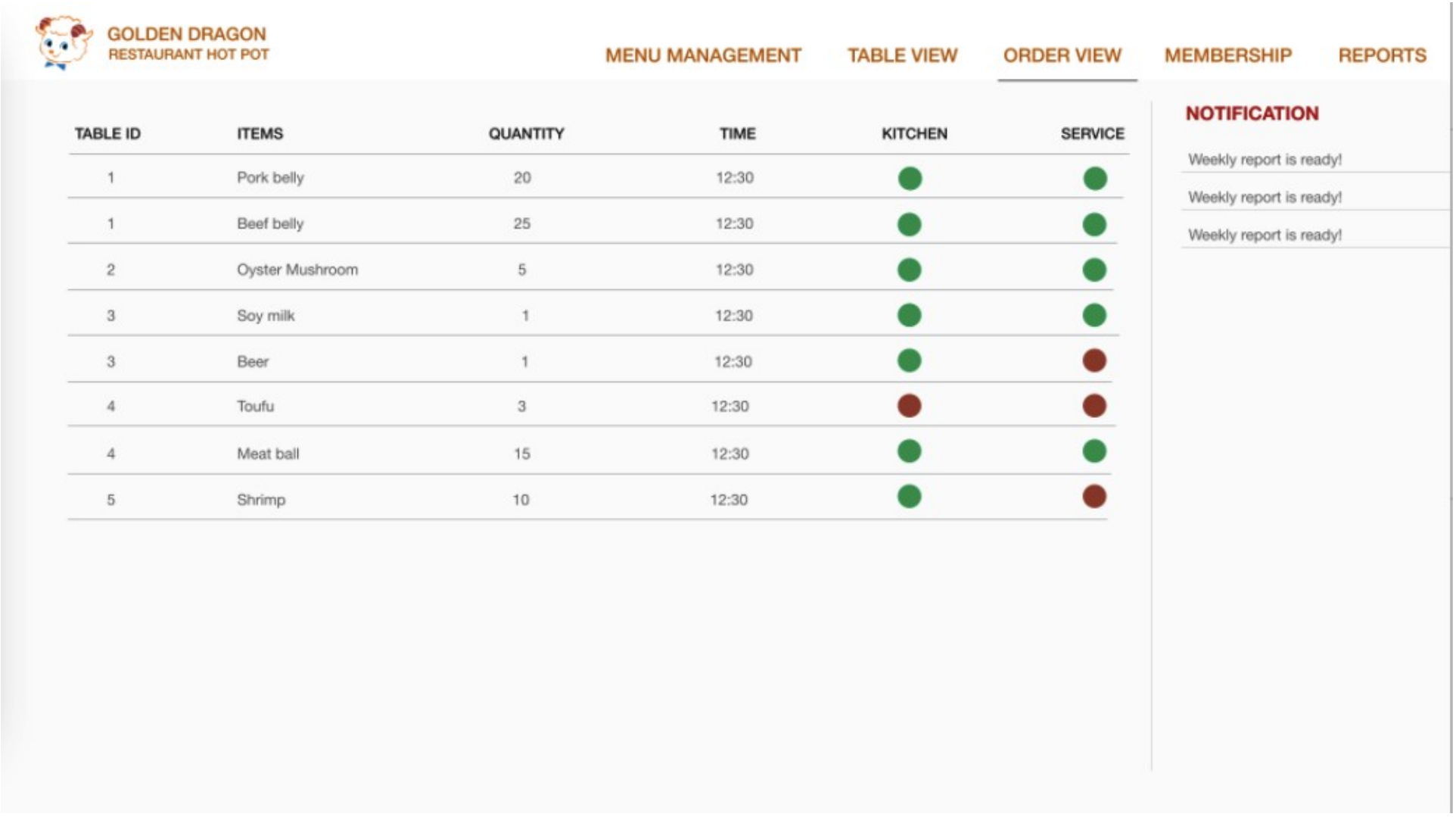
User Interface (UI) wireframe: Table status view



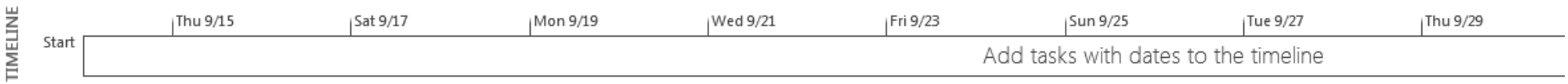
User Interface (UI) wireframe: Bill status view



User Interface (UI) wireframe: Order status view

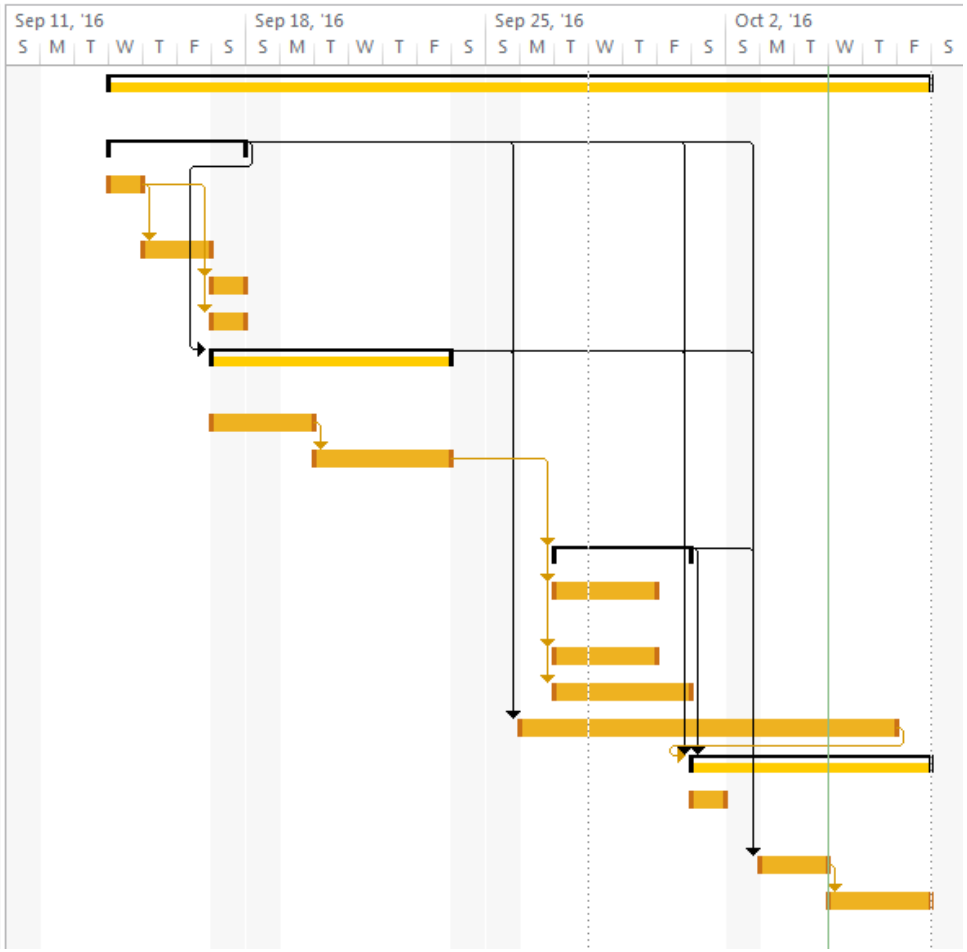


Project Plan Part A: Gantt chart




GANTT CHART

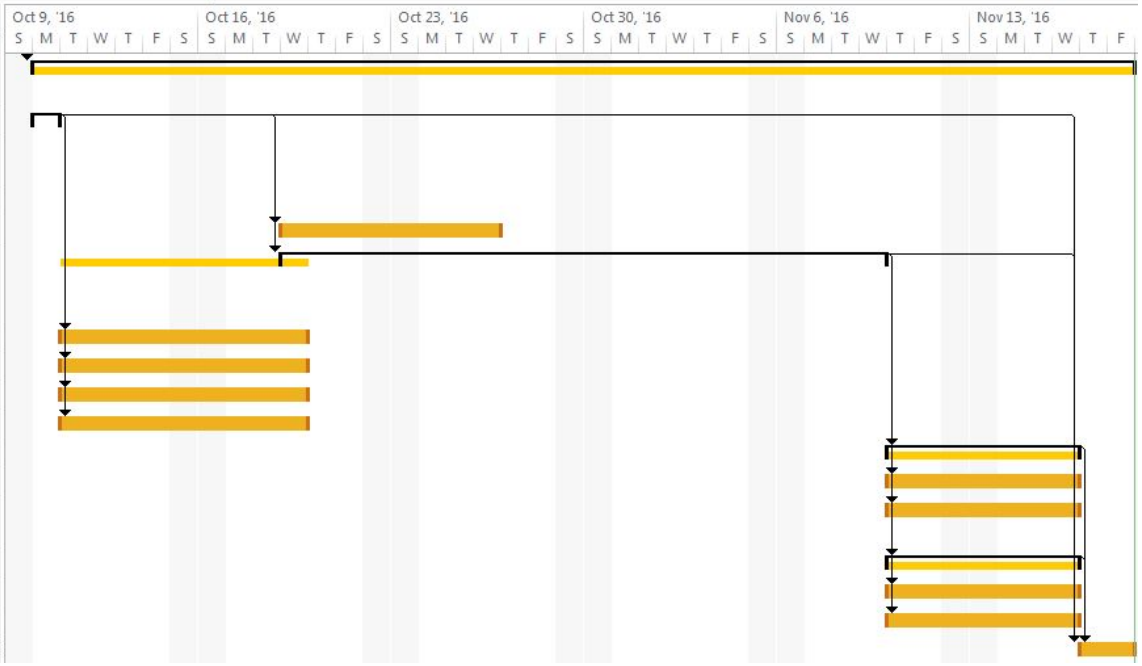
| | Task Name | Duration | Start | Finish | Predecess |
|---|---|---|---|---|---|
| 1 | **Restaurant Info System PART A** | **19 days** | **Wed 9/14/16** | **Fri 10/7/16** | |
| 2 | **Problem statement** | **3 days** | **Wed 9/14/1(** | **Sat 9/17/16** | |
| 3 | Problem needs and Subsystems | 1 day | Wed 9/14/16 | Wed 9/14/16 | |
| 4 | Vision statement | 2 days | Thu 9/15/16 | Fri 9/16/16 | 3 |
| 5 | System capabilities | 1 day | Sat 9/17/16 | Sat 9/17/16 | 3 |
| 6 | Business benefits | 1 day | Sat 9/17/16 | Sat 9/17/16 | 3 |
| 7 | **Workflow systems planning** | **6 days** | **Sat 9/17/16** | **Fri 9/23/16** | **2** |
| 8 | Workflow text description | 2 days | Sat 9/17/16 | Mon 9/19/16 | |
| 9 | Workflow activity diagrams (for each subsystem) | 4 days | Tue 9/20/16 | Fri 9/23/16 | 8 |
| 10 | **Use Cases** | **4 days** | **Tue 9/27/16** | **Fri 9/30/16** | **9** |
| 11 | Use case descriptions and chart | 3 days | Tue 9/27/16 | Thu 9/29/16 | 9 |
| 12 | User stories chart | 3 days | Tue 9/27/16 | Thu 9/29/16 | 9 |
| 13 | Subsystem charts | 4 days | Tue 9/27/16 | Fri 9/30/16 | 9 |
| 14 | Domain class diagram | 10 days | Mon 9/26/16 | Thu 10/6/16 | 2,7 |
| 15 | **Project plan and summary** | **6 days** | **Sat 10/1/16** | **Fri 10/7/16** | **2,7,10,14** |
| 16 | Technology tools for development | 1 day | Sat 10/1/16 | Sat 10/1/16 | |
| 17 | Gantt chart | 2 days | Mon 10/3/16 | Tue 10/4/16 | 2,7,10 |
| 18 | Project summary document | 3 days | Wed 10/5/16 | Fri 10/7/16 | 17 |



Project Plan Part B: Gantt chart


GANTT CHART

| | Task Name | Duration | Start | Finish | Predecessor |
|---|---|---|---|---|---|
| 19 | **Restaurant Info System PART B** | **30 days** | **Mon 10/10/16** | **Fri 11/18/16** | **1** |
| 20 | **Refinement to Part A** | **1 day?** | **Mon 10/10/1** | **Mon 10/10/1** | |
| 21 | Detailed Use Case descriptions | | | | |
| 22 | Detailed User Stories | | | | |
| 23 | CRC Cards | 6 days | Wed 10/19/1 | Wed 10/26/1 | 20 |
| 24 | **Detail Design Class Diagram (Attributes and Methods)** | **16 days** | **Wed 10/19/16** | **Wed 11/9/16** | **20** |
| 25 | Membership sub-system | 7 days | Tue 10/11/1 | Wed 10/19/1 | 20 |
| 26 | Ordering sub-system | 7 days | Tue 10/11/1 | Wed 10/19/1 | 20 |
| 27 | Payment sub-system | 7 days | Tue 10/11/1 | Wed 10/19/1 | 20 |
| 28 | Menu sub-system | 7 days | Tue 10/11/1 | Wed 10/19/1 | 20 |
| 29 | **Sequence Diagrams** | **5 days** | **Wed 11/9/1** | **Wed 11/16/:** | **24** |
| 30 | Order Items Use Case | 5 days | Thu 11/10/1 | Wed 11/16/1 | 24 |
| 31 | View Ordered Items Use Case | 5 days | Thu 11/10/16 | Wed 11/16/16 | 24 |
| 32 | **State Machine Diagrams** | **5 days** | **Wed 11/9/1** | **Wed 11/16/:** | **24** |
| 33 | SMD: Bill | 5 days | Thu 11/10/1 | Wed 11/16/1 | 24 |
| 34 | SMD: Item | 5 days | Thu 11/10/1 | Wed 11/16/1 | 24 |
| 35 | Project Plan Part B Gantt | 2 days | Thu 11/17/1 | Fri 11/18/16 | 20,24,29,32 |



Project Plan Part C: Gantt chart

GANTT CHART

| | Task Name | Duration | Start | Finish | Predecessors |
|---|---|---|---|---|---|
| 36 | **Restaurant Info System PART C** | **14 days** | **Wed 11/23/1** | **Mon 12/12/1** | **19** |
| 37 | Refinement to Part B | 1 day | Mon 11/21/16 | Mon 11/21/16 | 19 |
| 38 | High-level software architecture diagram | 2 days | Mon 11/21/16 | Tue 11/22/16 | 19 |
| 39 | Mock-up User Interfaces | 9 days | Mon 11/21/16 | Thu 12/1/16 | 19 |
| 40 | ERD Model and Database Schema | 5 days | Mon 11/21/16 | Fri 11/25/16 | 19 |
| 41 | Component and Deployment Diagrams | 4 days | Mon 11/21/16 | Thu 11/24/16 | 19 |
| 42 | Skeletal Code written in C# | 9 days | Mon 11/21/16 | Thu 12/1/16 | 19 |
| 43 | Technology tools for development | 1 day | Mon 11/21/16 | Mon 11/21/16 | 19 |
| 44 | Project Plan Part C Gantt | 7 days | Fri 12/2/16 | Mon 12/12/16 | 37,39,40,41,42,43,38 |