

Application Requirement Specification  
For  
Food Ordering System

Prepared By:  
Jingzhi Zhang

# Purpose and Scope Statement

The purpose of this project is to develop a food ordering system. The system has 2 users – admin and student. The system allows Customers to browse menu items, add items to their order, view their current order, and remove items. Customers can also check out to view the total amount for their order. Admins can edit menu item prices, add new menu items, and remove existing items through the admin interface. The application includes a graphical user interface (GUI) implemented in Java Swing to provide an interactive and user-friendly experience.

**Out of Scope:** Sophisticated order tracking, real-time order status updates, and payment processing are not handled by this system. Admins are not responsible for handling payment transactions, user account management beyond basic functionalities, or generating advanced analytics. This program does not have database.

## Requirements Narrative

In this food court management system project, the primary objective is to provide users with a seamless and intuitive platform to interact with the food court services. Users will have distinct roles, including regular users and administrators. The system will focus on the following key functionalities:

### 1. User Authentication:

- Users can create an account with a unique username and password.
- Registered users can log in to the system using their credentials.

### 2. User Roles:

- regular users and administrators.

- Regular users can view the menu, place orders, view their order details, and perform account-related actions.
- Administrators have the ability to edit menu items, add new items, and remove existing items.

### 3. Order Management:

- Users can add items to their order from the available menu categories.
- The system will calculate the total price of the order based on selected items and quantities.
- Users can view and modify their current order before proceeding to checkout.

### 4. Checkout and Payment:

- Users can proceed to checkout, where the system will display the total amount due.
- Completed orders will be cleared from the user's current order.

### 5. Menu Display:

- The menu is categorized into different sections such as entrees, drinks, desserts, pizza slices, and whole pizzas.
- Each menu item displays its name, price, and a corresponding image.

### 6. User Notifications:

- Users will receive informative notifications, such as successful login, registration, order placement.
- Error messages will be displayed if login credentials are incorrect or if there are issues with order processing.

### 7. Admin Features:

- Administrators can log in using specific admin credentials.
- Admins have the ability to edit the price of existing menu items, add new items to the menu, and remove items from the menu.

## 8. Admin Interface:

- An admin interface, accessible upon successful login, will provide options for editing menu items, adding new items, and removing items.

## 9. User Profile Management:

- Users can update their profiles by changing their passwords.
- The system should validate and ensure the security of user passwords.

## 10. Graphical User Interface (GUI):

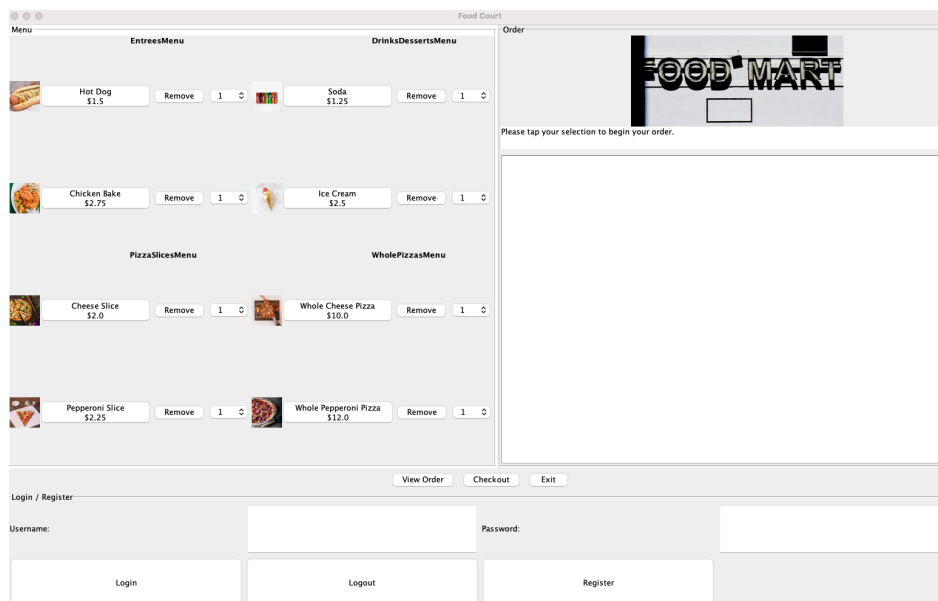
- The system will provide a graphical user interface for easy navigation and interaction.
- The GUI will include components like buttons, text fields, labels, and images to enhance the user experience.

## 11. System Exit:

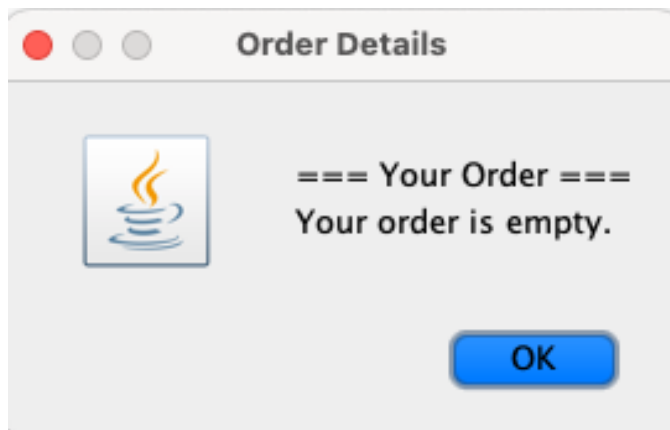
- Users can log out of the system, closing their session securely.
- Administrators can exit the admin interface, returning to the regular user interface.

# Objectives

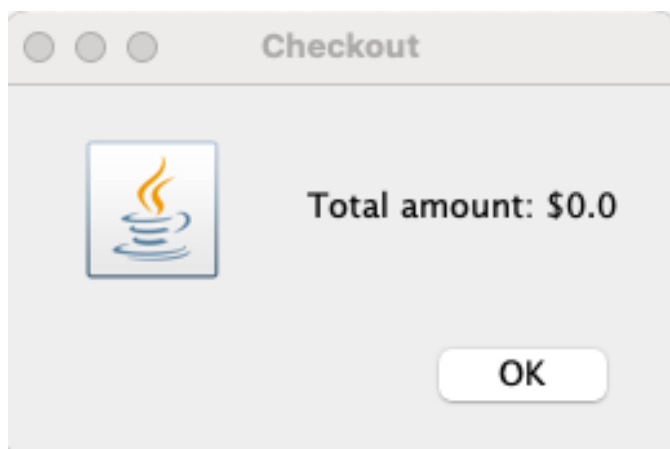
The basic order window and login/out for customer is as below



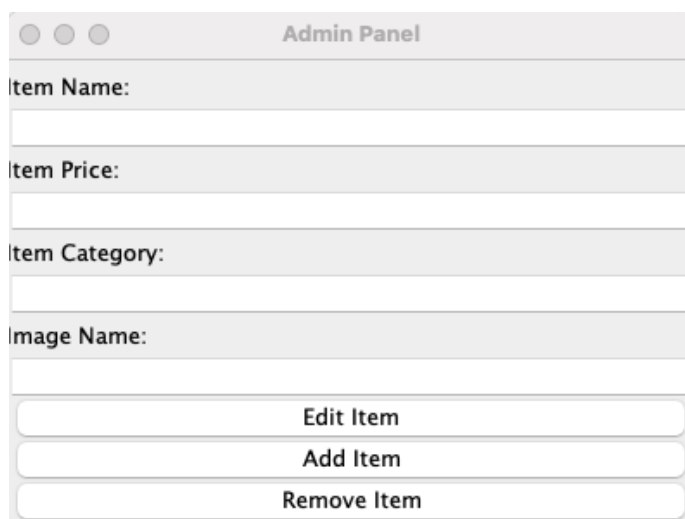
The order details window



The check window



The Admin view is as below

A screenshot of a macOS-style window titled "Admin Panel". The title bar includes three window control buttons. The form contains four labeled input fields: "Item Name:", "Item Price:", "Item Category:", and "Image Name:". Below these fields are three buttons: "Edit Item", "Add Item", and "Remove Item".

Item Name:
Item Price:
Item Category:
Image Name:
Edit Item
Add Item
Remove Item

# Functional Specification

## 1. User Class

The user class represents a registered user in the system.

### Attributes:

- username(String): The unique identifier for the user.
- password (String): The password associated with the user's account.

### Methods:

- getUsername(): String: Returns the username.
- getPassword(): String: Returns the password.

## 2. Order Class

The Order class manages the items selected by the user for purchase.

### Attributes:

- items (ArrayList<MenuItem>): A list of menu items in the order.

### Methods:

- addItem(item: MenuItem): Adds a menu item to the order.
- removeItem(item: MenuItem): Removes a menu item from the order.
- getItems(): ArrayList<MenuItem>: Returns the list of items in the order.
- calculateTotal(): double: Calculates the total cost of the order.
- clearOrder(): Clears all items from the order.

## 3. MenuItem Class

The MenuItem class represents an item on the food menu.

### Attributes:

- name (String): The name of the menu item.
- price (double): The price of the menu item.

- `imageName (String)`: The filename of the image associated with the menu item.

Methods:

- `getName()`: `String`: Returns the name of the menu item.
- `getPrice()`: `double`: Returns the price of the menu item.
- `setPrice(price: double)`: Sets the price of the menu item.
- `getImageName()`: `String`: Returns the filename of the associated image.

#### **4. Menu Interface**

The Menu interface defines the structure for different menu categories.

Methods:

- `getMenuItems()`: `List<MenuItem>`: Returns a list of menu items in the category.

#### **5. EntreesMenu, DrinksDessertsMenu, PizzaSlicesMenu, WholePizzasMenu**

##### **Classes**

These classes implement the Menu interface and represent specific menu categories.

#### **6. LoginManager Class**

The LoginManager class handles user authentication and registration.

Attributes:

- `users (Map<String, User>)`: A map to store registered users.
- `currentUser (User)`: The currently logged-in user.
- `foodCourtGUI (FoodCourtGUI)`: Reference to the GUI for displaying notifications.

Methods:

- `setFoodCourtGUI(foodCourtGUI: FoodCourtGUI)`: Sets the reference to the GUI.
- `registerUser(username: String, password: String)`: Registers a new user.
- `login(username: String, password: String)`: Logs in a user.
- `logout()`: Logs out the current user.
- `getCurrentUser(): User`: Returns the currently logged-in user.

## **7. FoodCourt Class**

The FoodCourt class orchestrates the overall functionality of the food court.

### Attributes:

- `menus (List<Menu>)`: A list of menu categories.
- `currentOrder (Order)`: The current user's order.
- `itemCategoryMap (Map<MenuItem, String>)`: Maps each menu item to its category.
- `loginManager (LoginManager)`: Manages user authentication.
- `admin (Admin)`: Handles administrative functions.

### Methods:

- `setMenuItemPrice(categoryName: String, itemName: String, newPrice: double)`: Updates the price of a menu item.
- `getItemCategoryMap(): Map<MenuItem, String>`: Returns the item-to-category mapping.
- `getAdmin(): Admin`: Returns the admin instance.
- `getLoginManager(): LoginManager`: Returns the login manager instance.
- `registerUser(username: String, password: String)`: Registers a new user.
- `loginUser(username: String, password: String)`: Logs in a user.
- `logoutUser()`: Logs out the current user.
- `getMenus(): List<Menu>`: Returns the list of menu categories.



- `removeOrderItem(item: MenuItem)`: Removes an item from the current order.
- `getCurrentOrder()`: `Order`: Returns the current order.
- `getCategoryName(item: MenuItem)`: `String`: Returns the category name of a menu item.
- `findMenuByCategory(category: String)`: `Menu`: Finds a menu by its category.
- `removeMenuItem(categoryName: String, itemName: String)`: Removes a menu item from a category.
- `placeOrder(categoryName: String, itemName: String)`: Places an order for a menu item.
- `displayOrder()`: Displays the current user's order.
- `getOrderMessage()`: `String`: Generates a message detailing the current order.

## **8. FoodCourtGUI Class**

The FoodCourtGUI class manages the graphical user interface for the Food Court Management System.

### Attributes:

- `foodCourt (FoodCourt)`: Reference to the main FoodCourt instance.
- `frame (JFrame)`: The main frame of the GUI.
- `menuPanels (List<JPanel>)`: Panels to display different menu categories.
- `orderPanel (JPanel)`: Panel to display the current order.
- `loginPanel (JPanel)`: Panel for user authentication.
- `notificationLabel (JLabel)`: Label to display notifications.

### Methods:

- `initGUI()`: Initializes the main GUI components.
- `displayMenus()`: Displays the menu panels.

- `displayOrder()`: Displays the current order.
- `displayLogin()`: Displays the login panel.
- `showNotification(message: String)`: Displays a notification to the user.
- `clearNotification()`: Clears the notification label.

## 9. Admin Class

The Admin class handles administrative functions within the Food Court Management System.

### Attributes:

- `foodCourt (FoodCourt)`: Reference to the main FoodCourt instance.

### Methods:

- `setFoodCourt(foodCourt: FoodCourt)`: Sets the reference to the FoodCourt instance.
- `modifyMenuItemPrice(categoryName: String, itemName: String, newPrice: double)`: Modifies the price of a menu item.
- `removeMenuItem(categoryName: String, itemName: String)`: Removes a menu item from a category.
- `addMenuItem(categoryName: String, item: MenuItem)`: Adds a menu item to a category.
- `createMenuCategory(categoryName: String)`: Creates a new menu category.
- `removeMenuCategory(categoryName: String)`: Removes a menu category.

## 10. AdminFrame Class

The AdminFrame class provides a separate frame for administrative actions.

### Attributes:

- `admin (Admin)`: Reference to the Admin instance.
- `frame (JFrame)`: The frame for administrative actions.

- categoryNameField (JTextField): Field for entering the category name.
- itemNameField (JTextField): Field for entering the item name.
- itemPriceField (JTextField): Field for entering the item price.
- actionComboBox (JComboBox<String>): Dropdown for selecting the

administrative action.

- executeButton (JButton): Button to execute the selected action.

#### Methods:

- initAdminFrame(): Initializes the components of the admin frame.
- addActionListener(listener: ActionListener): Adds an action listener to the

execute button.

- getSelectedAction(): String: Returns the selected administrative action.
- getCategoryName(): String: Returns the entered category name.
- getItemName(): String: Returns the entered item name.
- getItemPrice(): double: Returns the entered item price.
- clearFields(): Clears the input fields.

### **Libraries Used**

javax.swing: For building the graphical user interface.

java.util: For collections and utility classes.

java.awt: For basic windowing and layout management.

java.awt.event: For event handling.

java.util.stream: For stream processing of collections.

### **Technologies Needed**

Java Programming Language

Swing Library

Java Collections Framework

JOptionPane (javax.swing.JOptionPane)

Java I/O for File Handling

Java AWT and Java Swing for GUI Components

Java Event Handling (ActionListeners)

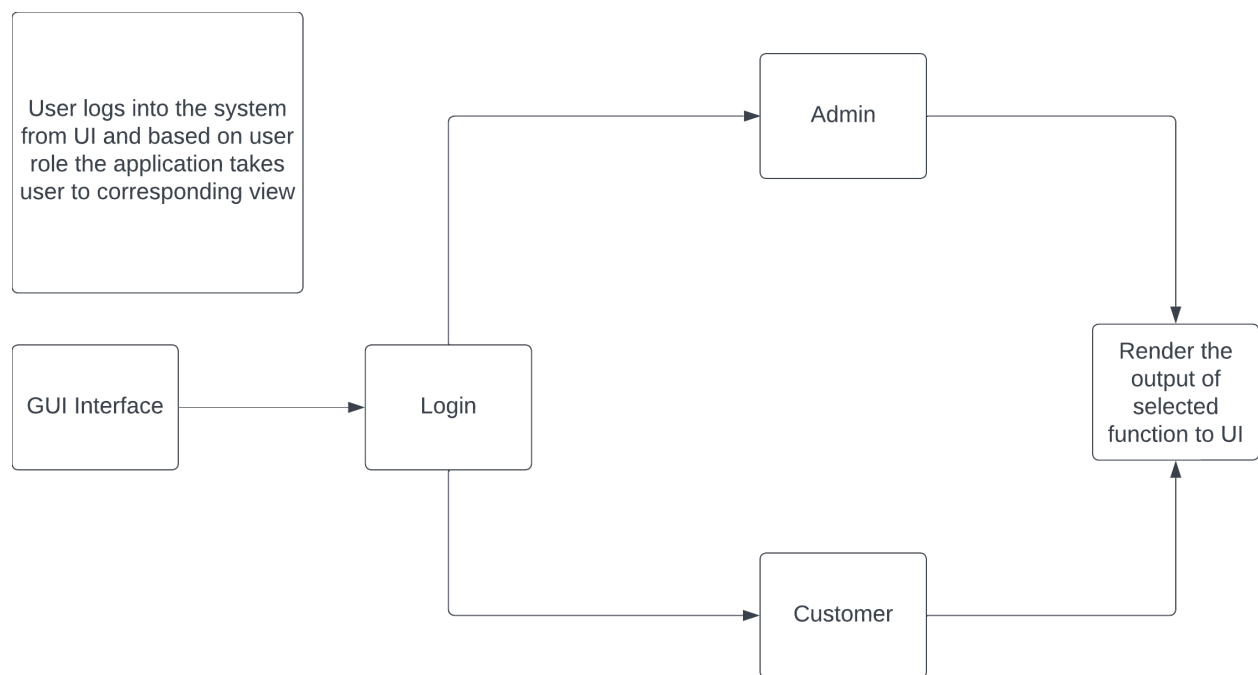
Java Collections Framework (java.util)

Java Image Handling (javax.imageio, java.awt.Image, javax.swing.ImageIcon)

MVC (Model-View-Controller) Architecture

Visual Studio Code IDE

## Logic Specification



## Next Step

After the Admin edits, adds, or removes menu items, the UI currently does not immediately show the corresponding changes; instead, it only displays a message in the terminal. As the next step:

1. incorporate more functions in the future to show the corresponding changes.
2. Enhance the user interface by adding more features, such as user feedback, error messages, or visual improvements.
3. Integrating a database into this project to enhance the ability to manage user information and orders more efficiently.