Cafe Management System

PROJECT BY:

Iqra Agha | 01-134251-026
Meerab Hanif | 01-134251-041

Catalog

Café Management System

1. Case Study Introduction

The Café Management System is designed to digitalize and streamline daily café operations for efficient service delivery. The system provides separate access levels for Admin and Staff to ensure secure and organized workflow. Its primary goal is to minimize manual effort, eliminate calculation errors, improve service quality, and maintain accurate digital records of all café activities. By implementing object-oriented design, the system ensures flexibility, scalability, and strong separation of responsibilities across different user roles.

The system includes a Graphical User Interface (GUI) developed using Qt Widgets in C++. The GUI provides interactive windows, buttons, and input fields, allowing Admin, Staff, and Customer to interact with the system efficiently.
The GUI is integrated with the object-oriented backend, ensuring separation between user interface and business logic.

Intended Users

- **Admin:** Manages menu items, staff accounts, and business reports.
- **Staff:** Handles customer orders, updates order statuses, and generates bills.
- **Customer:** Interacts with the system through a GUI, can place orders, generate receipts, and submit feedback.

The system follows a modular OOP design to ensure scalability and maintainability, making it suitable for small cafés or medium-scale food businesses.

2. Examining the Requirements

Functional Requirements

- **Admin Functions**
  - Add and remove staff members
  - Update menu item prices
  - Change menu item availability
  - Restock ingredient inventory

- View all customer feedback through the UI

- **Staff Functions**
    - Create customer orders
    - Add menu items to orders after checking inventory availability
    - Automatically deduct inventory when items are added to an order
    - Update order status

- **Customer Functions**
    - View available menu items
    - Place orders independently
    - Generate a receipt file for completed orders
    - Submit feedback with ratings and comments

- **System Functions**
    - Generate receipt files in text format
    - Store feedback persistently in a text file
    - Generate daily sales reports
    - Identify best-selling menu items
    - Display low-stock alerts for ingredients
    - Provide a role-based console UI for all interactions

Graphical User Interface Requirements

- The system shall provide a window-based graphical interface.
- Buttons shall be used to trigger system actions.
- Text fields and input controls shall replace console input.
- Dialog boxes shall display success or error messages.
- The GUI shall support role-based navigation (Admin, Staff, Customer).

- GUI events shall invoke backend OOP functions.

Non-functional Requirements

- System must be modular and object-oriented
- Data stored in files for persistence
- Input validation and error handling
- User-friendly graphical interface with event-driven interaction using Qt Widgets.

3. Identifying the Classes

Based on requirements, the following classes are identified:

Class	Description
User	Base class for Admin and Staff
Admin	Adds staff, manages menu & inventory
Staff	Takes orders, updates order status
Customer	Represents the person placing an order
MenuItem	Stores food/drink item details
Ingredient	Tracks ingredient stock
Order	Represents customer orders
InventoryManager	Verifies & updates ingredient quantities
Billing	Calculates totals & generates receipt
Report	Generates daily/weekly sales analytics
FeedbackManager	Manages feedback submission, storage, filtering, and file handling

- MainWindow (GUI Class): Handles all graphical user interface components and connects user actions with backend system operations.

4. Identifying Class Attributes

User

- username
- password

Admin

- staffUsernames[]
- menuRef
- inventoryRef
- feedbackRef

Staff

- inventoryRef

Customer

- username (inherited)
- password (inherited)

MenuItem

- id
- name
- price
- available
- ingredient names[]
- ingredient quantities[]

Ingredient

- name
- quantity

Order

- orderID
- menuItem array
- quantity array

- checkedOut

InventoryManager

- Ingredient array
- IngredientCount

Billing

- paymentMethod
- discountApplied

Report

- orderHistory
- totalSales

Feedback Entry

- id
- customerName
- rating (1–5)
- comments

MainWindow (GUI)

- inventory (InventoryManager)
- feedback (FeedbackManager)
- admin (Admin)
- staff (Staff)
- customer (Customer)

5. Object States & Activities

Order Lifecycle

- Created

- Items Added
- Checked Out
- Stored for reporting

MenuItem Lifecycle

- Created → Available → Modified → Removed

Ingredient Lifecycle

- Stocked → Used → Warning State (low stock) → Restocked

Feedback Lifecycle

- Submitted by customer
- Stored in file
- Viewed by admin through UI

6. Identifying Class Operations

User

- login()
- logout()
- changePassword()

Admin

- addStaff()
- removeStaff()
- updateMenuItemPrice()
- changeMenuAvailability()
- viewAllFeedback()

Staff

- createOrder()
- addItemToOrder()
- updateOrderStatus()

Customer

- placeOrder()
- submitFeedback()

MenuItem

- updatePrice()
- changeAvailability()

Order

- addItem()
- removeItem()
- computeTotal()
- checkout()

InventoryManager

- changeMenuAvailability()
- deductForItem()
- restock()

Billing

- generateReceipt()
- applyDiscounts()

Report

- dailySalesReport()

- bestSellingItems()
- inventoryAlerts()

Feedback

- submitFeedback()
- viewAll()

7. Collaboration Among Objects

Sequence: Placing an Order

- Staff clicks “Create Order” button in the GUI.
- GUI (MainWindow) creates an Order object.
- Staff selects a menu item and quantity using GUI controls.
- GUI triggers Staff::addItemToOrder().
- InventoryManager checks ingredient availability.
- Ingredients are deducted if sufficient stock is available.
- Staff clicks “Checkout” button.
- Order is finalized and stored for reporting.

Sequence: Restocking Inventory (Admin)

- Admin clicks “Restock Inventory” button in the GUI.
- GUI collects ingredient name and quantity.
- GUI calls Admin::restockInventory().
- InventoryManager updates ingredient stock levels.
- GUI displays confirmation message to the Admin.

Sequence: Submitting Feedback

- Customer enters feedback comment and rating using GUI form.
- Customer clicks “Submit Feedback” button.

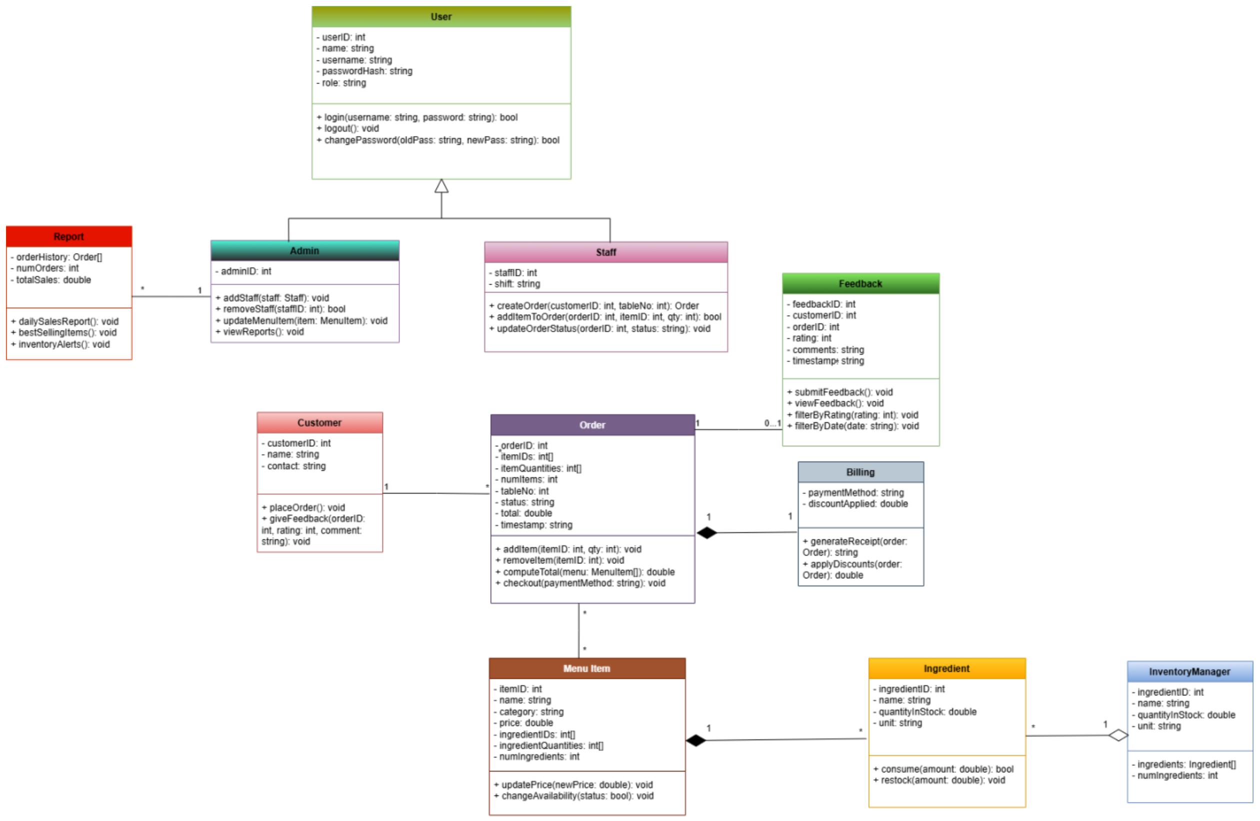
- GUI invokes Customer::giveFeedback().
- FeedbackManager stores the feedback in a file.
- Admin can later view all feedback through the GUI.

All interactions are event-driven, where GUI actions trigger backend object collaboration, ensuring separation between user interface and business logic.

8. Wrap-Up

This Café Management System implements a complete OOP structure with inheritance (User → Admin/Staff), aggregation (Order → MenuItems), and management modules (InventoryManager, Billing, Report).
The system provides a realistic workflow for café operations and ensures modularity, encapsulation, and reusability. The Feedback module enhances user engagement and allows the admin to analyze service quality. Its integration with Orders and Reports further improves the system's completeness and real-world usefulness.

9. UML


User
- userID: int
- name: string
- username: string
- passwordHash: string
- role: string
+ login(username: string, password: string): bool
+ logout(): void
+ changePassword(oldPass: string, newPass: string): bool
Report
- orderHistory: Order[]
- numOrders: int
- totalSales: double
+ dailySalesReport(): void
+ bestSellingItems(): void
+ inventoryAlerts(): void
Admin
- adminID: int
+ addStaff(staff: Staff): void
+ removeStaff(staffID: int): bool
+ updateMenuItem(item: MenuItem): void
+ viewReports(): void
Staff
- staffID: int
- shift: string
+ createOrder(customerID: int, tableNo: int): Order
+ addItemToOrder(orderID: int, itemID: int, qty: int): bool
+ updateOrderStatus(orderID: int, status: string): void
Feedback
- feedbackID: int
- customerID: int
- orderID: int
- rating: int
- comments: string
- timestamp: string
+ submitFeedback(): void
+ viewFeedback(): void
+ filterByRating(rating: int): void
+ filterByDate(date: string): void
Customer
- customerID: int
- name: string
- contact: string
+ placeOrder(): void
+ giveFeedback(orderID: int, rating: int, comment: string): void
Order
- orderID: int
- itemIDs: int[]
- itemQuantities: int[]
- numItems: int
- tableNo: int
- status: string
- total: double
- timestamp: string
+ addItem(itemID: int, qty: int): void
+ removeItem(itemID: int): void
+ computeTotal(menu: MenuItem[]): double
+ checkout(paymentMethod: string): void
Billing
- paymentMethod: string
- discountApplied: double
+ generateReceipt(order: Order): string
+ applyDiscounts(order: Order): double
Menu Item
- itemID: int
- name: string
- category: string
- price: double
- ingredientIDs: int[]
- ingredientQuantities: int[]
- numIngredients: int
+ updatePrice(newPrice: double): void
+ changeAvailability(status: bool): void
Ingredient
- ingredientID: int
- name: string
- quantityInStock: double
- unit: string
+ consume(amount: double): bool
+ restock(amount: double): void
InventoryManager
- ingredientID: int
- name: string
- quantityInStock: double
- unit: string
- ingredients: Ingredient[]
- numIngredients: int
*
1
1
0...1
1
*
1
1
*
*
1
*
*
1

10. GitHub

https://github.com/Iqra039/Cafe-Management-System

https://github.com/scriptveil/Cafe-Management-System.git

11. Contributions

Meerab Hanif

I contributed to shaping the basic idea of the project and took part in the early system analysis and planning. My primary responsibility was preparing the complete design report, including documenting requirements, describing system features, and explaining the UML diagrams. I ensured the report was clear, organized, and aligned with the project design.

Iqra Agha

I contributed to developing the basic idea of the project and participated in the initial planning discussions. My main responsibility was creating all UML diagrams, including the use case, class, activity, and sequence diagrams. I ensured that the system structure, workflows, and object interactions were clearly represented and aligned with the project requirements.