OOP FINAL PROJECT

Cloud Kitchen Management System

Submitted to:

Sir Uzair

Group Members:

Aamna Batool

Fasiha Qasim

Fatima Tehreem

Course: Object-Oriented Programming

Date: June 20, 2025

Cloud Kitchen Project Report

Designed using C++ and OOP Principles

Cloud Kitchen Management System – OOP Project

1. Abstract

The Cloud Kitchen Management System is an object-oriented C++ project that simulates the core functionalities of a virtual food delivery platform. This system enables customer interaction with a digital menu, facilitates order placement, and manages customer data. It supports both Regular and Premium customers with differentiated billing systems and includes administrative features such as reviewing all past orders and searching customer-specific records. The project showcases efficient use of inheritance, polymorphism, abstraction, and encapsulation in solving real-world problems. This report presents an in-depth analysis of the project architecture, code implementation, advantages, limitations, and future enhancements.

2. Introduction

The restaurant and food delivery industries have witnessed exponential growth with the rise of cloud kitchens. A cloud kitchen, or ghost kitchen, operates without a dine-in facility and focuses solely on online orders. This OOP-based simulation provides a simplified version of such a system.

Developed in C++, the project employs fundamental and advanced Object-Oriented Programming concepts to design a modular and extensible application. This project aims to demonstrate how OOP principles can be effectively utilized to represent real-world business logic, including users, menu systems, and delivery workflows.

3. Objective

- Implement a console-based food ordering system using OOP.
- Differentiate customer types with unique billing systems.
- Maintain user order history using file handling.
- Enable admin to access and manage customer data.
- Demonstrate abstraction, inheritance, and polymorphism.

a. Class Structure

- **User**: Abstract base class for all users.
- **Customer**: Derived from User, base for Regular and Premium customers.
- **RegularCustomer and PremiumCustomer**: Offer different billing logic.
- **Admin**: Handles login, order viewing, and search.
- **FoodItem**: Represents individual food products.
- **Menu**: Displays all items and retrieves them by ID.
- **DeliveryPartner**: Provides delivery simulation.
- **KitchenSystem**: Main driver class, handling user interactions.

b. Class Diagram (Suggested for Visuals)

```
User (Abstract)
   |
   ---> Customer (Abstract)
```

```
          ---> RegularCustomer
          ---> PremiumCustomer
   ---> Admin

Menu --> FoodItem
KitchenSystem --> Menu, DeliveryPartner, Customer
```

4. OOP Concepts Applied

a. Abstraction

```
class User {
public:
    virtual void displayProfile() = 0;
};
```

b. Inheritance

Customer and Admin both inherit from User. RegularCustomer and PremiumCustomer extend Customer.

c. Polymorphism

```
float calculateBill(float baseAmount) {
    return baseAmount + 50; // Regular
}

float calculateBill(float baseAmount) {
    return baseAmount * 0.9; // Premium
}
```

d. Encapsulation

Private/protected members ensure modularity and data safety. Data access is handled via public methods.

5. Features

- **Customer Mode**: Place orders and view profiles.
- **Admin Mode**: Password protected; view/search orders.
- **Two Customer Types**:
  - Regular – with delivery fee
  - Premium – with discount and no delivery fee
- **Persistent Order Storage**: Text file-based logging.
- **Menu System**: Vector-based food item storage.
- **Delivery Partner Simulation**: Adds real-world feel.

6. Code Walkthrough

A. Menu Initialization

```
Menu() {
    items.push_back(FoodItem(1, "Zinger Burger", 350));
    ...
}
```

B. Order Flow

- Choose type (Regular/Premium)
- Enter name/contact
- Show menu → Select items
- Bill calculated with polymorphism
- Show + Save order history

C. Admin Functionalities

```
if (a.login(pass)) {
    a.viewAllOrders();
    a.searchCustomerOrder(key);
}
```

7. Advantages

- **Reusability**
- **Scalability**
- **Maintainability**
- **Practical Learning**
- **File Persistence**

8. Limitations

- Console only (no GUI)
- No DB (file-based)

- No input validation
- Static menu
- Weak password security

9. Suggested Improvements

- Add encryption
- Add exception handling
- Use database (MySQL/SQLite)
- GUI interface (Qt/WinAPI)
- API integration

10. Conclusion

This project demonstrates the real-world use of OOP in C++ through a simulated cloud kitchen system. It applies core principles like inheritance, encapsulation, and polymorphism to deliver a scalable, modular solution. Despite its console limitations, the system offers a strong foundation for future improvements and learning.

11. References

- C++ Primer by Stanley B. Lippman
- Object-Oriented Programming using C++ by E. Balagurusamy
- `https://www.geeksforgeeks.org/c-plus-plus/`