Mini Project Report

Online Food Delivery System (Like Zomato / Swiggy)

Name: Sakshi
UID: 24BCA10162
Course: BCA
Subject: Object Oriented Programming (OOPs)

Section: 24BCA-2 ”A”

Introduction

In today’s digital world, technology has transformed the way we live, work, and eat. One of the most convenient examples of this transformation is **online food delivery systems** such as **Zomato**, **Swiggy**, and **Uber Eats**. These platforms have made it possible for people to order food from their favorite restaurants with just a few clicks, without stepping out of their homes.

This project titled “**Online Food Delivery System**” aims to create a simple version of such a platform using **Object-Oriented Programming (OOPs)** concepts. It focuses on how different entities like **customers**, **restaurants**, **delivery personnel**, and **orders** interact with each other in a structured and logical way.

The system provides an easy interface for customers to browse restaurants, view menus, select food items, place orders, and make payments. It also allows restaurants to

manage their menu items and track incoming orders. The purpose of this project is not only to simulate a real-world application but also to demonstrate how OOP concepts can simplify complex systems through modular design and reusability.

Objective

The main objective of this project is to design a software model that:

- Allows customers to order food online conveniently.
- Connects restaurants and delivery partners through a single platform.
- Implements **OOP concepts** such as **encapsulation, inheritance, abstraction,** and **polymorphism**.
- Demonstrates how data and functionality can be grouped into classes for better structure and maintainability.
- Provides an efficient, user-friendly, and scalable solution to food delivery needs.

Modules of the System

The Online Food Delivery System consists of the following key modules:

1. **User Module –** Handles registration, login, and profile management for customers.
2. **Restaurant Module –** Manages restaurant information, menu items, and order processing.
3. **Menu Module –** Displays the available food items with prices and categories.
4. **Order Module –** Allows customers to add food to the cart, place orders, and calculate bills.
5. **Delivery Module –** Assigns delivery personnel and tracks order delivery status.
6. **Payment Module –** Manages payment options like Cash on Delivery (COD) or online payment.

OOPs Concepts Used

1. **Class and Object:**
   Used to represent real-world entities such as Customer, Restaurant, FoodItem, and Order.
2. **Encapsulation:**
   Data and related functions are combined within a class, ensuring data security and abstraction.
3. **Inheritance:**
   Enables reusability. For example, Customer and RestaurantOwner classes can inherit from a parent User class.
4. **Polymorphism:**
   Allows the same function (e.g., displayInfo()) to behave differently depending on the object calling it.
5. **Abstraction:**
   Hides complex logic and exposes only necessary information to the user, making the system easier to use.

Example Code (C++)

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;
```

```
class User {
protected:
    string name, email, contact;
public:
    void setUser(string n, string e, string c) {
        name = n;
        email = e;
        contact = c;
    }
    void displayUser() {
        cout << "Name: " << name
             << "\nEmail: " << email
             << "\nContact: " << contact << endl;
    }
};

class Customer : public User {
    string address;
public:
    void setAddress(string a) {
        address = a;
    }
    void placeOrder() {
        cout << "Order placed successfully!\n";
    }
};

class FoodItem {
public:
    string itemName;
    float price;
```

```
    FoodItem(string n, float p) {
        itemName = n;
        price = p;
    }

    void displayItem() {
        cout << itemName << " - Rs." << price << endl;
    }
};

class Order {
    vector<FoodItem> items;
    float total = 0;
public:
    void addItem(FoodItem f) {
        items.push_back(f);
        total += f.price;
    }

    void showOrder() {
        cout << "\nOrder Details:\n";
        for (auto i : items)
            i.displayItem();
        cout << "Total Bill: Rs." << total << endl;
    }
};

int main() {
    Customer c;
    c.setUser("Sakshi", "sakshi@email.com", "9876543210");
    c.setAddress("Patna, Bihar");
```

```
    c.displayUser();
    c.placeOrder();

    FoodItem f1("Pizza", 250);
    FoodItem f2("Burger", 150);

    Order o;
    o.addItem(f1);
    o.addItem(f2);
    o.showOrder();

    return 0;
}
```

System Working

1. The customer creates an account and logs into the system.
2. The system displays a list of nearby restaurants.
3. The customer browses the menu and selects food items.
4. The selected items are added to the cart.
5. The customer confirms the order and chooses a payment method (COD or Online).
6. The restaurant receives the order and starts preparing the food.

7. A delivery person is assigned automatically.
8. The delivery person picks up and delivers the food to the customer.
9. The order status updates as **Preparing** → **Out for Delivery** → **Delivered**.

Features

- Easy registration and login for users.
- Search and filter options for restaurants.
- Real-time order tracking.
- Multiple payment options.
- User-friendly interface.
- Receipt generation after successful order.

Advantages

- Saves time and effort for both customers and restaurants.
- Increases restaurant sales through digital visibility.
- Provides real-time tracking and updates.
- Encourages cashless transactions.
- Promotes safety and convenience.

Conclusion

This project successfully demonstrates the application of **Object-Oriented Programming concepts** to develop a real-world system.
By using OOPs principles, we can break down a complex system into smaller, manageable classes, making it easier to understand and maintain.

The **Online Food Delivery System** not only shows how customers interact with restaurants and delivery partners but also teaches how real-world objects can be represented through classes, attributes, and methods in programming. It can be further enhanced with database integration, a web interface, and APIs for real-world deployment.

Summary

The **Online Food Delivery System** project is a complete demonstration of OOPs in action.

It combines practical features (like user login, order placement, and tracking) with core OOP principles (like encapsulation and inheritance).
This project gives students real insight into how modern applications like Zomato or Swiggy are designed and helps them understand structured and reusable programming approaches.