# Design Overview for Swin-Bite

Name: Show Wai Yan

Student ID: 105293041

# Summary of Program

The Food Delivery Management System is a comprehensive platform that connects customers, restaurants, and delivery drivers in a seamless food ordering and delivery ecosystem. The program allows customers to browse restaurants, view menus, place orders, and track deliveries in real-time. Restaurants can manage their menus, process incoming orders, and update order statuses, while delivery drivers can accept delivery assignments, update their location, and complete deliveries efficiently.

The system implements advanced design patterns including the Observer pattern for real-time notifications, ensuring all stakeholders receive immediate updates about order status changes, delivery assignments, and completion notifications. The platform handles payment processing, delivery service management, and provides comprehensive tracking capabilities for enhanced user experience. Key features include intelligent driver assignment based on location, multi-user notification system, order history management, and secure payment processing through integrated bank accounts.

# Sketch of sample output

A computer screen with a menu

AI-generated content may be incorrect.

# Required Roles

## User (Abstract Base Class)

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Store user identification | userId: int, username: string, email: string | Primary user data |
| Handle authentication | password: string, isAuthenticated: bool | Security management |
| Manage user type | userType: UserType | Polymorphic behavior |
| Process payments | bankAccount: BankAccount | Financial transactions |
| User operations | Login(): bool, Logout(): void, UpdateProfile(): bool | Core user functions |
| Observer management | Subscribe(), Unsubscribe(), NotifyObservers() | Notification system |

## Customer (Inherits User)

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Shopping management | shoppingCart: ShoppingCart, orderHistory: List<Order> | Order management |
| Notification handling | notifications: List<Notification> | Personalized alerts |
| Food ordering | AddToCart(), PlaceOrder(): bool | Core customer functionality |
| Restaurant browsing | BrowseRestaurants(): List<Restaurant> | Discovery features |
| Order tracking | TrackOrder(orderId: int): string | Real-time monitoring |
| Observer implementation | Update(notification: Notification): void | Receives notifications |

## Restaurant (Inherits User)

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Restaurant details | restaurantId: int, name: string, address: string, rating: double | Business information |
| Menu management | menu: List<Food>, category: string, operatingHours: string | Food catalog |
| Order processing | ProcessOrder(): bool, UpdateOrderStatus(): bool | Order management |
| Menu operations | AddMenuItem(): bool, UpdateMenu(): bool | Content management |
| Observer implementation | Update(notification: Notification): void | Receives order notifications |

## DeliveryDriver (Inherits User)

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Driver identification | driverId: int, licenseNumber: string | Professional credentials |
| Vehicle management | vehicleType: VehicleType | Delivery capacity |
| Location tracking | currentLocation: Location, UpdateLocation(): void | GPS functionality |
| Availability status | isAvailable: bool | Work status |
| Delivery management | activeDeliveries: List<Delivery>, rating: double | Performance tracking |
| Delivery operations | AcceptDelivery(): bool, CompleteDelivery(): bool | Core driver functions |
| Observer implementation | Update(notification: Notification): void | Receives delivery assignments |

## Food (Abstract Base Class)

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Food identification | foodId: int, name: string, description: string | Product details |
| Pricing | price: decimal | Cost information |
| Categorization | category: FoodCategory, restaurant: Restaurant | Organization |
| Availability | isAvailable: bool, prepTime: int | Status management |
| Information display | GetDetails(): string | Customer interface |

## Dish (Inherits Food)

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Serving details | servingSize: string, spiceLevel: int | Portion information |
| Nutritional info | calories: int, ingredients: List<string> | Health data |

## Drink (Inherits Food)

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Beverage specs | volume: string, temperature: Temperature | Physical properties |
| Drink properties | isCarbonated: bool, hasAlcohol: bool | Type classification |

## Snack (Inherits Food)

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Package details | packageSize: string | Portion information |
| Health classification | isHealthy: bool, allergens: List<string> | Dietary information |

## ShoppingCart

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Item storage | items: List<Food>, quantities: Dictionary<Food, int> | Cart contents |
| Cart operations | AddItem(), RemoveItem(), Clear(): void | Manipulation functions |
| Cost calculation | CalculateTotal(): decimal | Price computation |
| Order conversion | ConvertToOrder(): Order | Checkout process |

## Order

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Order identification | orderId: int, orderDate: DateTime | Tracking information |
| Participants | customer: Customer, restaurant: Restaurant | Relationship management |
| Order contents | orderItems: List<Food>, quantities: Dictionary<Food, int> | Product details |
| Financial data | totalAmount: decimal | Cost tracking |
| Status management | status: OrderStatus, UpdateStatus(): void | Workflow control |
| Delivery integration | delivery: Delivery, deliveryAddress: string | Location services |
| Observer pattern | observers: List<Observer>, NotifyObservers(): void | Real-time updates |

## Delivery

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Delivery identification | deliveryId: int, trackingNumber: string | Tracking system |
| Assignment | order: Order, driver: DeliveryDriver | Relationship management |
| Location management | pickupLocation: Location, deliveryLocation: Location | GPS coordinates |
| Time tracking | estimatedTime: DateTime, actualTime: DateTime | Performance metrics |
| Cost management | deliveryFee: decimal | Pricing |
| Status control | status: DeliveryStatus, UpdateStatus(): void | Workflow management |
| Driver operations | AssignDriver(): bool, GetTrackingInfo(): string | Assignment functions |

## DeliveryService

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Driver pool | availableDrivers: List<DeliveryDriver> | Resource management |
| Active tracking | activeDeliveries: List<Delivery> | Current operations |
| Driver assignment | FindAvailableDriver(): DeliveryDriver | Optimization logic |
| Delivery creation | AssignDelivery(): Delivery | Order processing |
| Cost calculation | CalculateDeliveryFee(): decimal | Pricing algorithm |
| Tracking services | TrackDelivery(): string | Customer interface |

## Observer (Interface)

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Notification handling | Update(notification: Notification): void | Observer pattern contract |

## NotificationService

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Observer management | observers: List<Observer> | Subscription handling |
| Subscription control | Subscribe(), Unsubscribe(): void | Observer registration |
| Mass notifications | NotifyAll(notification: Notification): void | Broadcast functionality |
| Targeted messaging | SendNotification(userId: int, message: string): void | Individual notifications |

## Notification

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Message identification | notificationId: int, userId: int | Tracking data |
| Content management | message: string, timestamp: DateTime | Message details |
| Classification | type: NotificationType, isRead: bool | Organization and status |
| User operations | MarkAsRead(): void, GetContent(): string | Interface functions |

## Location

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Coordinate storage | latitude: double, longitude: double | GPS data |
| Address management | address: string | Human-readable location |
| Distance calculation | CalculateDistance(other: Location): double | Spatial operations |
| Coordinate display | GetCoordinates(): string | String representation |

## BankAccount

| **Responsibility** | **Type Details** | **Notes** |
| --- | --- | --- |
| Account details | accountNumber: string, pin: string | Financial credentials |
| Balance management | balance: decimal, isActive: bool | Account status |
| Payment processing | ProcessPayment(amount: decimal): bool | Transaction handling |
| Balance inquiry | GetBalance(): decimal | Account information |

## UserType (Enumeration)

| **Value** | **Notes** |
| --- | --- |
| Customer | End users placing orders |
| RestaurantOwner | Business owners managing restaurants |
| DeliveryDriver | Drivers handling deliveries |
| Admin | System administrators |

## FoodCategory (Enumeration)

| **Value** | **Notes** |
| --- | --- |
| MainCourse | Primary dishes |
| Beverage | Drinks |
| Snack | Light items |

## OrderStatus (Enumeration)

| **Value** | **Notes** |
| --- | --- |
| Placed | Initial order state |
| Confirmed | Restaurant accepted |
| Preparing | Being cooked |
| Ready | Available for pickup |
| OutForDelivery | In transit |
| Delivered | Completed successfully |
| Cancelled | Order terminated |

## DeliveryStatus (Enumeration)

| **Value** | **Notes** |
| --- | --- |
| Assigned | Driver allocated |
| PickedUp | Order collected |
| InTransit | On the way |
| Delivered | Successfully completed |
| Failed | Delivery unsuccessful |

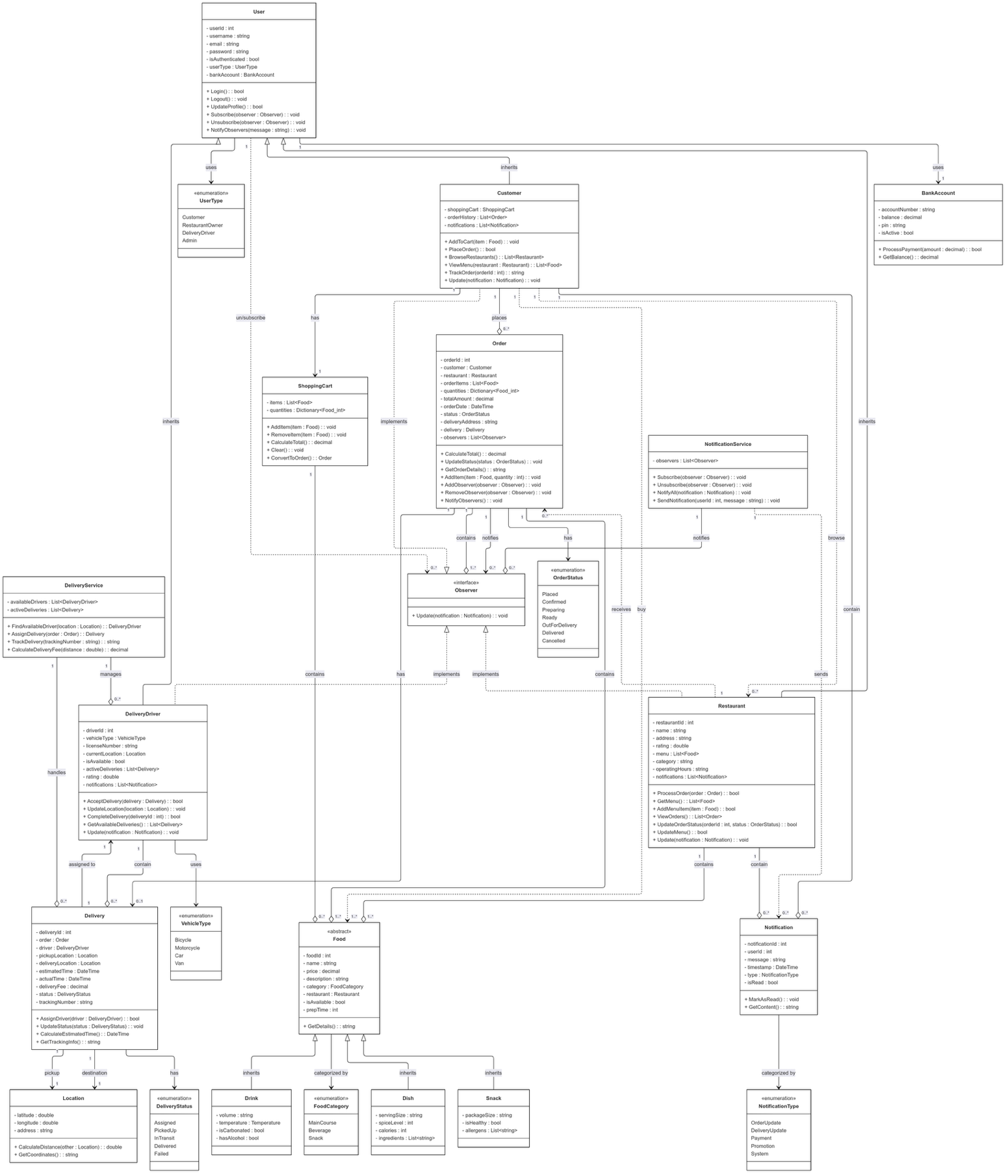
## VehicleType (Enumeration)

| **Value** | **Notes** |
| --- | --- |
| Bicycle | Eco-friendly, short distance |
| Motorcycle | Fast, medium distance |
| Car | Weather protection, longer distance |
| Van | Large orders, commercial |

## NotificationType (Enumeration)

| **Value** | **Notes** |
| --- | --- |
| OrderUpdate | Order status changes |
| DeliveryUpdate | Delivery progress |
| Payment | Financial transactions |
| Promotion | Marketing messages |
| System | Administrative alerts |

# Class Diagram



[Click here if Diagram is low resolution](https://drive.google.com/file/d/17M01kAfFd-48ZImpXQzyqNUOVSkoUMB2/view?usp=sharing)

# Sequence Diagram

A white sheet with yellow sticky notes

AI-generated content may be incorrect.

[Click here if diagram is low resolution](https://drive.google.com/file/d/1Zw-Cq3eSh8jKT6vqKGrf8tGUD-F4-Wr9/view?usp=sharing)