Submission for assignment Object-Oriented Design:

Team 7 anh tài members:

Nguyễn Hoàng Hạo - 10422095

Nguyễn Trần Tuấn Anh - 10422005

Nguyễn Tiến Khoa - 10422035

Tạ Lê Khôi Vĩ - 10422083

Lê Đức Thanh Kim - 10422105

Trần Ngọc Anh Toàn - 10422118

Trần Nguyễn Minh Trí – 10422119

1. **This is for analyzing model and relationship for each designed object: docs**

Step 1: Convert Analysis Classes to Design Classes

The analysis classes—**Menu, Menu Item, Order, Customer, Waiter**—are conceptual. In the design phase, we refine them and add supporting classes for complete functionality.

Design Classes:

Menu → Menu

- Represents all food/drink items
- Aggregates MenuItem objects

Menu Item → MenuItem

- Now includes: image, description, prepTime
- Added getDetails() method

Order → Order + OrderItem

- Split into main Order and line items
- Added OrderStatus enum for tracking

Customer → Customer

- Lightweight implementation
- Added order history tracking

Waiter → Waiter

- Now implements RestaurantUser interface
- Added wearable device integration

Additional Supporting Classes:

- **RestaurantServer**: Central coordination
- **TabletUI**: Customer ordering interface
- **WearableUI**: Waiter device interface
- **Payment** (Abstract): Payment processing
- **StorageManager**: Database handling
- **OrderItem**: Individual order components

Step 2: Identify Key Class Relationships

Inheritance/Implementation:

- Payment → CreditCardPayment, CashPayment
- UI → TabletUI, WearableUI
- RestaurantUser → Waiter, KitchenStaff

Aggregation:

- Menu contains MenuItem
- Order contains OrderItem

Association:

- Order ↔ Customer (who placed it)
- Order ↔ Waiter (who delivers it)
- RestaurantServer ↔ TabletUI/WearableUI

Step 3: Define Class Attributes and Methods

StorageManager

Attributes:

- database: DatabaseConnection

Methods:

- saveOrder(order: Order): void
- loadMenu(): Menu
- getOrder(orderId: int): Order

Order

Attributes:

- orderId: int
- items: List<OrderItem>
- status: OrderStatus
- customer: Customer

Methods:

- calculateTotal(): double
- updateStatus(): void

Payment (Abstract)

Attributes:

- amount: double

Methods:

- processPayment(): boolean ○ Implemented by:
  - CreditCardPayment
  - CashPayment

Step 4: Ensure Open/Closed Principle (OCP)

✓ **Payment System**: Add MobilePayment without changes

✓ **Menu Items**: Extend with ComboMenuItem ✓ **UI Components**: Add

ManagerDashboard easily

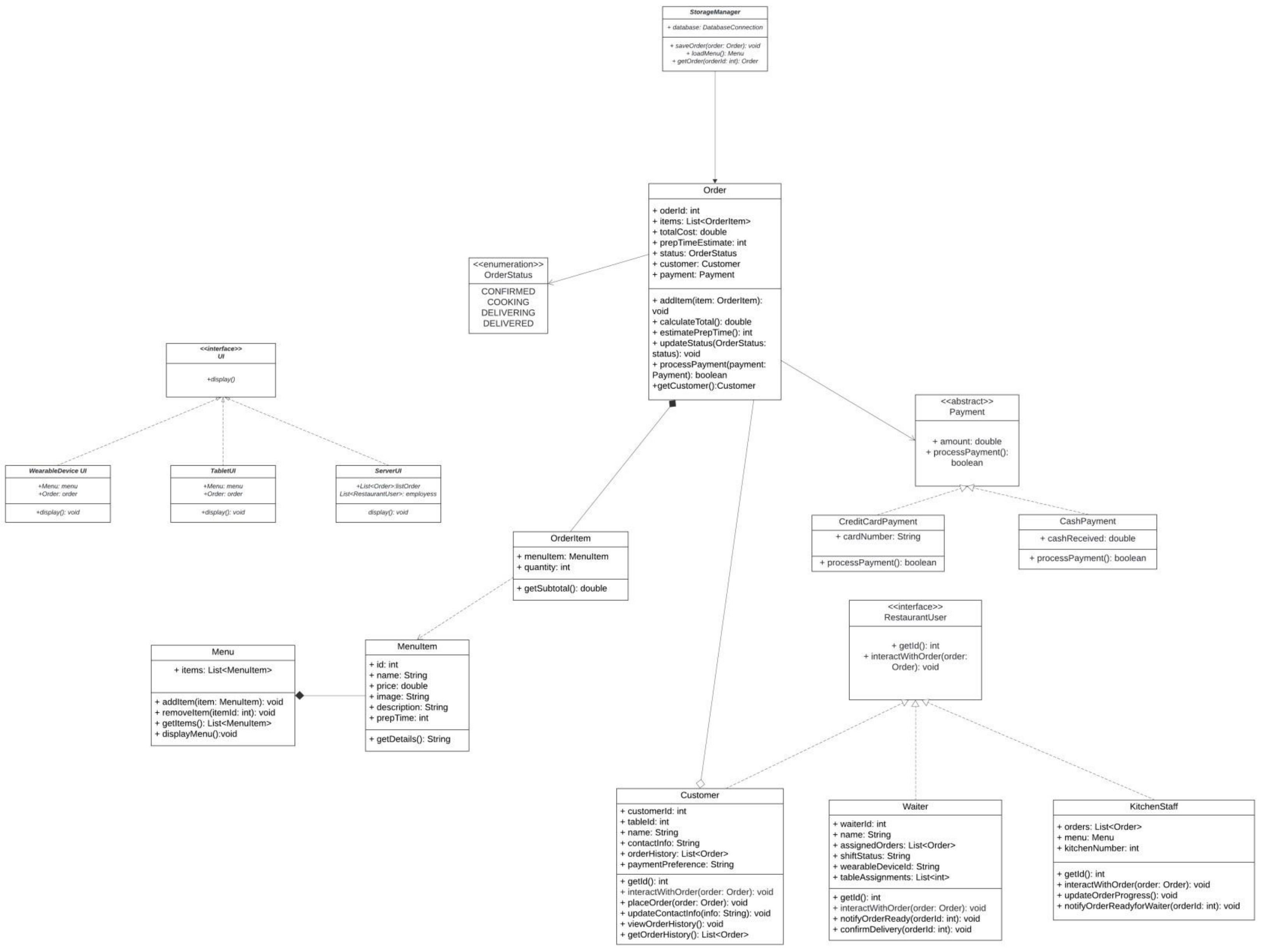
Storage Implementation:

- **RestaurantServer** handles in-memory storage
- Can extend with **StorageManager** for databases

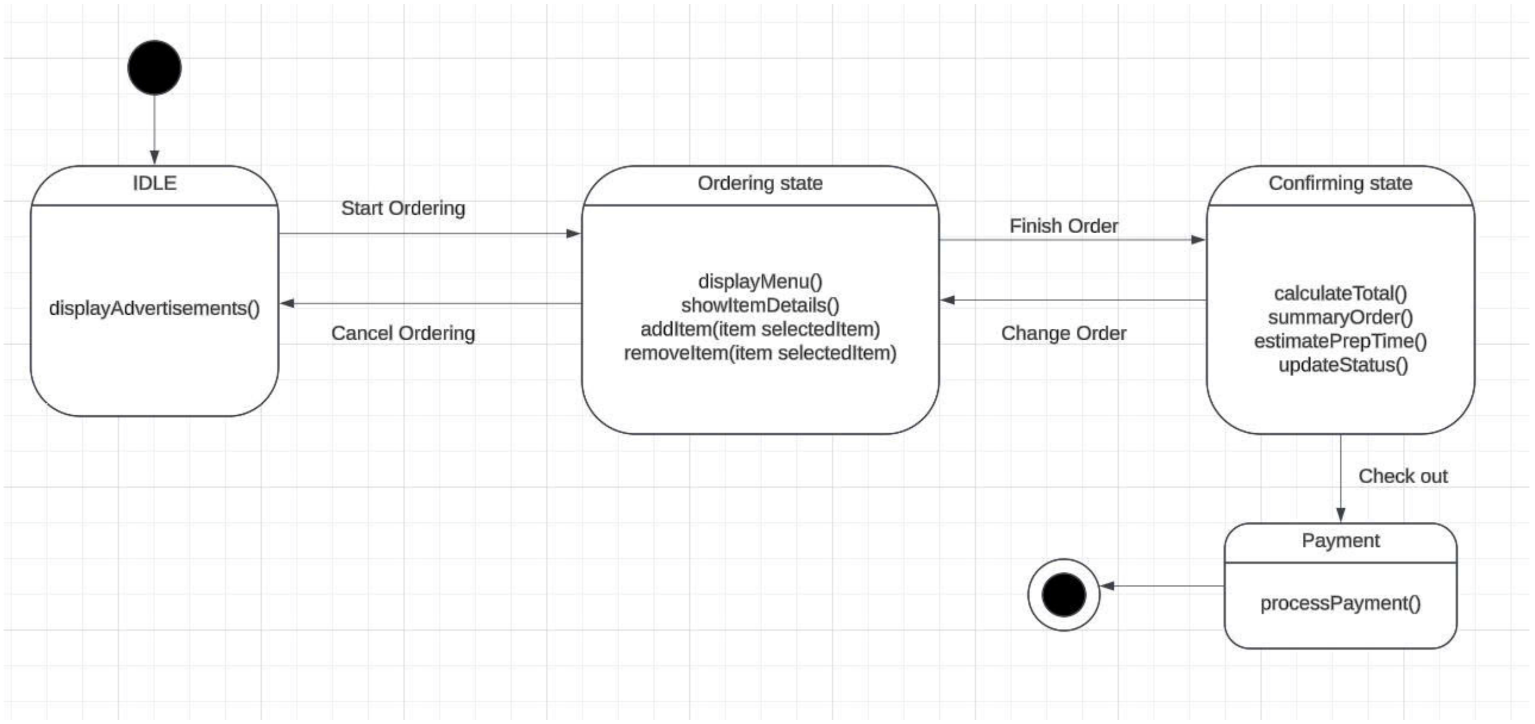
Network Implementation:

- **RestaurantServer** manages basic communication
- Can add **NetworkManager** for explicit handling

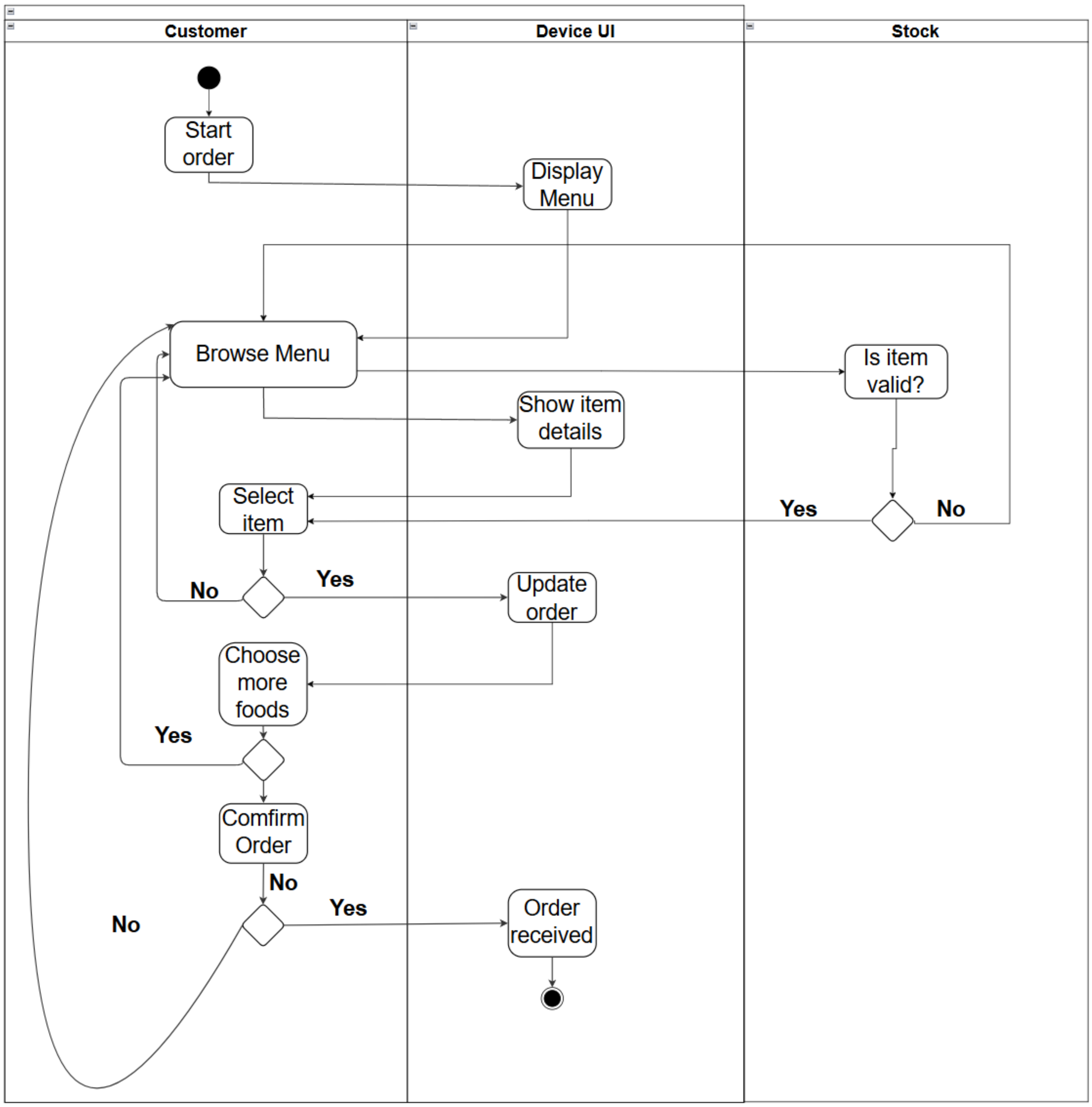
UML Diagram


StorageManager
+ database: DatabaseConnection
+ saveOrder(order: Order): void
+ loadMenu(): Menu
+ getOrder(orderId: int): Order
Order
+ oderId: int
+ items: List<OrderItem>
+ totalCost: double
+ prepTimeEstimate: int
+ status: OrderStatus
+ customer: Customer
+ payment: Payment
+ addItem(item: OrderItem): void
+ calculateTotal(): double
+ estimatePrepTime(): int
+ updateStatus(OrderStatus: status): void
+ processPayment(payment: Payment): boolean
+getCustomer():Customer
<<enumeration>>
OrderStatus
CONFIRMED
COOKING
DELIVERING
DELIVERED
<<interface>>
UI
+display()
WearableDevice UI
+Menu: menu
+Order: order
+display(): void
TabletUI
+Menu: menu
+Order: order
+display(): void
ServerUI
+List<Order>:listOrder
List<RestaurantUser>: employess
display(): void
<<abstract>>
Payment
+ amount: double
+ processPayment(): boolean
CreditCardPayment
+ cardNumber: String
+ processPayment(): boolean
CashPayment
+ cashReceived: double
+ processPayment(): boolean
OrderItem
+ menuItem: MenuItem
+ quantity: int
+ getSubtotal(): double
Menu
+ items: List<MenuItem>
+ addItem(item: MenuItem): void
+ removeItem(itemId: int): void
+ getItems(): List<MenuItem>
+ displayMenu():void
MenuItem
+ id: int
+ name: String
+ price: double
+ image: String
+ description: String
+ prepTime: int
+ getDetails(): String
<<interface>>
RestaurantUser
+ getId(): int
+ interactWithOrder(order: Order): void
Customer
+ customerId: int
+ tableId: int
+ name: String
+ contactInfo: String
+ orderHistory: List<Order>
+ paymentPreference: String
+ getId(): int
+ interactWithOrder(order: Order): void
+ placeOrder(order: Order): void
+ updateContactInfo(info: String): void
+ viewOrderHistory(): void
+ getOrderHistory(): List<Order>
Waiter
+ waiterId: int
+ name: String
+ assignedOrders: List<Order>
+ shiftStatus: String
+ wearableDeviceId: String
+ tableAssignments: List<int>
+ getId(): int
+ interactWithOrder(order: Order): void
+ notifyOrderReady(orderId: int): void
+ confirmDelivery(orderId: int): void
KitchenStaff
+ orders: List<Order>
+ menu: Menu
+ kitchenNumber: int
+ getId(): int
+ interactWithOrder(order: Order): void
+ updateOrderProgress(): void
+ notifyOrderReadyforWaiter(orderId: int): void

2. State Machine Diagram



3.Activity Diagram


Customer
Device UI
Stock
Start order
Display Menu
Browse Menu
Is item valid?
Show item details
Select item
Yes
No
No
Yes
Update order
Choose more foods
Yes
Comfirm Order
No
Yes
No
Order received