OOPT Stage 2040 - Design

: Distributed vending machine

202211252 고승우
202211300 박연주
202211349 이채유
202214202 김민석

Activity 2041. Design Real Use Cases

Use case	1. Show Menu
Actor	None
Purpose	사용자에게 자판기 모드 화면을 보여준다.
Overview	음료 선택, 인증코드 입력, 관리자 모드 중 하나를 선택할 수 있는 메뉴를 출력한다.
Type	Primary
Cross Reference	System Functions: R1.2 Use cases: “Select Menu”
Pre-Requisites	N/A
Typical Courses of Events	(S): System 1. (S) 시스템이 초기화면으로 음료 선택, 인증코드 입력, 관리자 모드 중 하나를 선택할 수 있는 메뉴를 출력한다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	N/A

Activity 2041. Design Real Use Cases

Use case	2. Select Menu
Actor	User
Purpose	사용자가 자판기의 모드를 선택한다.
Overview	시스템이 출력한 메뉴인 음료 선택, 인증코드 입력, 관리자 모드에서 하나를 선택한다.
Type	Primary
Cross Reference	System Functions: R1.1, R1.3, R3.5, R4.1 Use cases: “Show Menu”, “Show item”, “Check Authentication Code”, “Admin Login”
Pre-Requisites	시스템 인터페이스에 음료 선택, 인증코드 입력, 관리자 모드 중 하나를 선택할 수 있는 메뉴가 출력된 상태여야 한다.
Typical Courses of Events	(A): Actor 1. (A) 사용자가 음료 선택, 인증코드 입력, 관리자 모드 중 원하는 버튼을 선택한다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	N/A

Activity 2041. Design Real Use Cases

Use case	3. Show Item
Actor	None
Purpose	사용자가 원하는 음료를 선택할 수 있게 한다.
Overview	DVM에서 판매하는 음료의 목록들과 그의 개수를 출력한다.
Type	Primary
Cross Reference	System Functions: R1.2 R1.4 Use cases: “Select Mode”, “Choose Item”
Pre-Requisites	사용자가 ‘음료 선택’ 메뉴를 선택한 상태여야 한다.
Typical Courses of Events	(S): System 1. (S) 시스템이 DVM에 있는 음료의 목록과 선택 가능한 개수를 출력한다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	N/A

Activity 2041. Design Real Use Cases

Use case	4. Choose Item
Actor	User
Purpose	사용자가 원하는 음료를 선택한다.
Overview	DVM에서 판매하는 음료의 목록 중 사용자가 원하는 음료의 종류와 그 음료의 개수를 선택한다.
Type	Primary
Cross Reference	System Functions: R1.3 R2.1 R2.3 Use cases: “Show Item”, “Pay”, “Check Stock”
Pre-Requisites	시스템이 정상적으로 메뉴를 출력한 상황이어야 한다.
Typical Courses of Events	(A): Actor 1. (A) 사용자가 원하는 음료의 종류와 개수를 선택한다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	N/A

Activity 2041. Design Real Use Cases

Use case	5. Provide Item
Actor	None
Purpose	사용자가 원하는 음료를 제공한다.
Overview	사용자가 선택한 음료를 개수에 맞게 사용자에게 제공한다.
Type	Primary
Cross Reference	System Functions: R3.5 R2.1 R2.4 Use cases: “Check Authentication Code”, “Pay”, “Update Stock”
Pre-Requisites	결제가 정상적으로 진행된 상태이거나 사용자가 입력한 인증코드의 검사가 올바르게 끝나야 한다.
Typical Courses of Events	(S): System 1. (S) 시스템은 사용자에게 제공해야하는 음료를 개수에 맞춰서 제공한다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	N/A

Activity 2041. Design Real Use Cases

Use case	6. Pay
Actor	None
Purpose	사용자의 결제를 진행한다.
Overview	사용자에게 입력받은 카드 정보를 바탕으로 결제를 진행한다.
Type	Primary
Cross Reference	System Functions: R1.4 R2.1 R2.2 R2.5 Use cases: “Valid Card”
Pre-Requisites	1. DVM에 사용자가 선택한 음료의 재고가 있어야 한다. 2. 결제할 카드의 정보가 유효하고 잔액이 충분한지 확인이 되어야한다.
Typical Courses of Events	(S): System (C): Card Company 1. (S) 시스템은 카드사에게 결제 요청을 보낸다. 2. (C) 카드사는 카드의 잔액에서 결제 요청 금액을 차감한다. 3. (C) 카드사는 결제가 정상적으로 진행되었다고 시스템에게 알린다. 4. (S) 시스템은 사용자에게 결제가 정상 완료 되었다고 알린다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	N/A

Use case	7. Valid Card
Actor	User
Purpose	사용자가 입력한 카드정보가 유효한지 확인하고, 음료를 구매하기에 잔액이 충분한지 검증한다.
Overview	사용자에게 입력받은 카드 정보가 유효한지 확인하고, 입력받은 카드에 얼마 만큼의 잔액이 남았는지 계산한다. 이때, 잔액이 충분하면 결제에 성공하고 잔액이 충분하지 않으면 결제에 실패한다.
Type	Primary
Cross Reference	System Functions: R2.1 R2.2 R2.3 R3.1 Use cases: “Pay”, “Prepay”
Pre-Requisites	사용자가 구매할 음료의 재고가 우리 DVM 또는 다른 DVM에서 확인 되어야한다.
Typical Courses of Events	(A): Actor (S): System (C): Card Company 1. (S) 시스템은 사용자에게 결제를 진행할 카드 정보를 입력을 요청한다. 2. (A) 사용자는 결제를 진행할 카드 정보를 입력한다. 3. (S) 시스템은 카드 정보가 유효한지 카드사에 요청한다. 4. (C) 카드사는 카드 정보가 유효한지 확인한다. 5. (C) 카드 정보가 유효한지 여부에 대한 메세지를 시스템에게 보낸다. (E1) 5. (S) 시스템은 사용자에게 입력받은 카드 정보를 바탕으로 카드사에게 잔액 확인요청과 결제 금액을 보내준다. 6. (C) 카드사는 그 카드에 잔액이 얼마 있는지 확인한다. 7. (C) 잔액이 충분한지에 대한 메세지를 시스템에게 보낸다. (E2)
Alternatives Courses of Events	N/A
Exceptions Courses of Events	E1. 유효하지않은 카드 정보일 경우: 1. 사용자가 유효하지 않은 카드 정보를 입력하면 결제를 진행하지 않고 사용자에게 다시 카드 정보 입력을 요청한다. 2. 세 번 이상 잘못 입력 시, 음료 선택 시스템 화면을 출력한다. E2. 카드의 잔액이 부족할 경우: 1. 사용자가 잔액이 부족한 카드정보를 제공하면 오류 메시지를 띄우고 음료 선택 시스템 화면을 출력한다.

Activity 2041. Design Real Use Cases

Use case	8. Check Stock
Actor	None
Purpose	DVM에 해당하는 음료의 재고가 있는지 파악한다.
Overview	사용자가 선택한 음료 혹은 다른 DVM에서 요청한 음료의 재고가 있는지 파악한다.
Type	Primary
Cross Reference	System Functions: R1.4 R2.2 R 2.3 R3.2 R3.3 Use cases: “Valid Card”, “Check other DVMs”, “Request from other DVMs”
Pre-Requisites	사용자가 System에게 음료를 요청하거나, 다른 DVM에게서 음료의 재고 확인의 요청이 있어야 한다.
Typical Courses of Events	(S): System 1.(S) 시스템은 사용자가 요청한 음료의 재고가 있는지 확인한다. (E1) 2. 결제 화면을 띄운다.
Alternatives Courses of Events	(S): System 1. (S) 시스템은 다른 DVM이 요청한 음료의 재고가 있는지 확인한다. (E2) 2. (S) 요청한 음료의 재고 응답 메세지를 다른 DVM에게 보낸다.
Exceptions Courses of Events	E1. 재고가 충분하지 않을 경우 : 다른 dvm에서 구매할 것인지 묻는다. E2. Other DVM에서 잘못된 형식의 message가 왔을 경우 요청을 처리하지 않는다.

Activity 2041. Design Real Use Cases

Use case	9. Update Stock
Actor	None
Purpose	음료의 재고를 실시간으로 업데이트한다.
Overview	사용자가 결제에 성공해 음료를 제공하거나 다른 DVM에서 선결제해 system이 음료를 제공해야 하는 경우 음료의 재고를 업데이트한다.
Type	Primary
Cross Reference	System Functions: R1.5 R3.3 Use cases: “Provide Item”, “Request from other DVMs”
Pre-Requisites	사용자가 결제에 성공해서 음료를 제공하거나 다른 DVM에서 사용자에게 음료를 제공하도록 요청 받은 경우여야 한다.
Typical Courses of Events	(S): System 1. (S) 시스템은 사용자에게 제공한 음료의 종류와 개수를 파악한다. 2. (S) 시스템은 그 음료의 개수를 재고에서 차감한다.
Alternatives Courses of Events	(S): System 1. (S) 시스템은 다른 DVM에서 요청한 음료의 종류와 개수를 파악한다. 2. (S) 시스템은 그 음료의 개수를 재고에서 차감한다.
Exceptions Courses of Events	N/A

Activity 2041. Design Real Use Cases

Use case	10. Prepay
Actor	None
Purpose	사용자의 선결제를 진행한다.
Overview	사용자가 선택한 음료의 재고가 현재 DVM에 없어, 재고가 확인된 다른 DVM에서 음료를 받을 수 있도록 선결제를 진행한다.
Type	Primary
Cross Reference	System Functions: R2.2 R3.1 R3.3 R3.4 Use cases: “Valid Card”
Pre-Requisites	다른 DVM에서 사용자가 요청한 음료의 재고가 확인 되어야 한다. 카드 정보가 유효하고 잔액이 충분해야한다.
Typical Courses of Events	(S): System (C): Card Company 1. (S) 시스템은 카드사에게 결제 요청을 보낸다. 2. (C) 카드사는 카드의 잔액에서 결제 요청 금액을 차감한다. 3. (C) 카드사는 결제가 정상적으로 진행되었다고 시스템에게 알린다. 4. (S) 시스템은 사용자에게 결제가 정상 완료 되었다고 알린다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	

Activity 2041. Design Real Use Cases

Use case	11. Check other DVMs
Actor	None
Purpose	다른 DVM에게 요청한 음료의 재고와 다른 DVM의 위치를 파악한다.
Overview	사용자가 선택한 음료가 DVM에 없을 때, 다른 DVM과 통신하여 사용자가 구매하려는 음료의 재고를 확인하고 위치 정보를 응답 받는다.
Type	Primary
Cross Reference	System Functions: R2.3 Use cases: “Check Stock”
Pre-Requisites	사용자가 요청한 음료의 재고가 현재 DVM에 없어야 한다.
Typical Courses of Events	(S): System (O): Other DVM 1. (S) 시스템은 Other DVM들에게 사용자가 요청한 음료의 재고 정보를 요청을 한다. 2. (O) Other DVM은 시스템에게 요청한 재고 정보와 자신의 위치정보를 넘겨준다. 3. (S) 시스템은 재고 정보를 이용해 다른 DVM들에서 사용자에게 음료를 제공할만큼 재고가 충분한지 판단한다. 4. (S) 재고가 있는 DVM이 하나라도 존재하면 선결제를 진행한다. (E1) 5. (S) 재고가 충분하다고 판단된 DVM들의 직선 거리를 계산해서, 가장 가까운 DVM 순으로 구매요청 메세지를 보낸다. 선결제 요청에는 음료종류, 개수, 생성한 인증코드 정보를 전달한다. 6. (O) 선결제 요청에 대해 가능 여부를 응답한다. 7. (S) 선결제가 가능하다고 응답한 DVM이 있으면 해당 과정을 완료한다. (E2)
Alternatives Courses of Events	N/A
Exceptions Courses of Events	E1. 재고가 있는 DVM이 없다면 선결제가 불가능함을 사용자에게 알리고 메인화면으로 돌아간다. E2. 선결제 요청을 받을 수 있는 DVM이 없다면 선결제 금액을 사용자에게 환불해준다.

Activity 2041. Design Real Use Cases

Use case	12. Request from other DVMs
Actor	None
Purpose	다른 DVM에서 보낸 요청을 처리한다.
Overview	다른 DVM에서 온 요청에 대해 응답한다. 재고 확인 요청이 오면 요청받은 재고 정보와 위치를 반환하고, 선결제 요청이 왔을 때 선결제가 가능하다면, 인증코드를 받고 음료를 판매된 것으로 처리한다.
Type	Secondary
Cross Reference	System Functions: R2.3 Use cases: “Check Stock”
Pre-Requisites	다른 DVM에서 요청이 와야한다.
Typical Courses of Events	(S): System (O): Other DVM 1. (O) Other DVM은 시스템에게 재고 확인 요청을 보낸다. (E1) 2. (S) 시스템은 Other DVM에게 요청한 재고 정보와 위치정보를 넘겨준다. 3. (O) Other DVM은 시스템에게 사용자가 요청한 음료에 대해 선결제 요청을 한다. (E2) 4. (S) 시스템은 선결제 요청을 받아 선결제가 가능하다고 판단되면, 인증코드를 저장하고 음료를 판매된 것으로 처리한다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	E1.Other DVM이 잘못된 형식의 재고확인 요청을 보내면 요청을 처리하지 않는다. E2. a. Other DVM이 잘못된 형식의 선결제 요청을 보내면 요청을 처리하지 않는다. b. Other DVM이 우리 DVM이 제공하는 음료의 개수를 초과하면 요청을 처리하지 않는다

Activity 2041. Design Real Use Cases

Use case	13. Provide Authentication Code
Actor	None
Purpose	인증코드를 사용자에게 제공한다.
Overview	선결제가 완료된 다음, 다른 DVM에서 음료를 받을 수 있게 인증코드를 사용자에게 제공한다.
Type	Primary
Cross Reference	System Functions: R3.1 R3.4 Use cases:
Pre-Requisites	선결제가 완료되어야 한다.
Typical Courses of Events	(A) :Actor (S): System 1. (S) 시스템이 사용자의 선결제에 대한 인증코드를 발급하여 사용자에게 새로운 팝업창으로 10초동안 띄워준다. 2. (A) 사용자는 인증코드를 받는다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	N/A

Activity 2041. Design Real Use Cases

Use case	14. Provide Available DVM Location
Actor	None
Purpose	사용자가 선결제 한 음료를 받을 수 있는 DVM의 위치를 사용자에게 제공한다.
Overview	선결제가 완료된 후, 그 음료를 제공해줄 수 있는 다른 DVM의 위치를 사용자에게 제공한다.
Type	Primary
Cross Reference	System Functions: R3.1 R3.5 Use cases:
Pre-Requisites	선결제가 완료되어야 한다.
Typical Courses of Events	(A) :Actor (S): System 1. (S) 시스템이 사용자의 선결제에 대해 음료를 가져갈 다른 DVM의 위치정보를 사용자에게 새로운 팝업창으로 10초동안 띄워준다. 2. (A) 사용자는 위치 정보를 받는다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	N/A

Activity 2041. Design Real Use Cases

Use case	15. Check Authentication Code
Actor	User
Purpose	사용자가 입력한 인증코드를 검증한다.
Overview	사용자가 입력한 인증코드를 다른 DVM에게서 선결제 완료 후 전달받은 인증코드와 같은 지 검사한다.
Type	Primary
Cross Reference	System Functions: R1.2 R1.5 R3.6 Use cases: “Select Menu”, “Provide Item”
Pre-Requisites	선결제가 완료되어야 한다.
Typical Courses of Events	(A) :Actor (S): System 1. (A) 사용자가 인증코드를 입력한다. (E1) 2. (S) 시스템은 입력받은 인증코드를 검증하여 선결제 인증코드와 일치하는 지 확인한다. (E2)
Alternatives Courses of Events	N/A
Exceptions Courses of Events	E1. 인증코드 형식이 맞지 않을 시: 1. 사용자가 형식에 맞지 않는 인증코드를 입력하면 인증코드 불일치로 판단하고 오류 메시지를 띄운 후, 다시 인증번호 입력 요청을 한다. 2. 3번 이상 틀릴 시 시스템 초기화면으로 돌아간다. E2. 인증코드가 다를 시: 1. 사용자가 올바르지 않은 인증코드를 입력하면 인증코드 불일치로 오류 메시지를 띄운 후, 다시 인증번호 입력 요청을

Activity 2041. Design Real Use Cases

Use case	16. Admin Login
Actor	Administrator
Purpose	관리자 모드로 로그인 한다.
Overview	음료에 대한 재고 및 가격을 조정할 수 있는 관리자 모드로 로그인한다.
Type	Secondary
Cross Reference	System Functions: R1.2 R4.1 R4.2 R4.3 R4.4 Use cases: “Select Menu”, “Admin Logout”, “Set Item Stock”, “Set Item Price”
Pre-Requisites	메뉴 선택에서 관리자 로그인을 선택해야한다.
Typical Courses of Events	(A) :Administrator (S): System 1. (A) 관리자가 관리자 아이디와 비밀번호를 입력한다. (E1) 2. (S) 시스템은 입력받은 아이디와 비밀번호를 검증하여 일치하면, 관리자 메뉴를 출력한다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	E1. 관리자가 잘못된 관리자 아이디 비밀번호를 입력하면 오류 메시지를 띄우고 입력창을 재출력한다.

Activity 2041. Design Real Use Cases

Use case	17. Admin Logout
Actor	Administrator
Purpose	관리자 모드에서 로그아웃한다.
Overview	관리자 모드에서 로그아웃한다.
Type	Secondary
Cross Reference	System Functions: R4.1 R4.2 Use cases:
Pre-Requisites	관리자로 로그인이 된 상태여야 한다.
Typical Courses of Events	(A) :Administrator (S): System 1. (A) 관리자 로그아웃 버튼을 누른다. 2. (S) 로그아웃을 진행해 관리자 모드에서 나와 메인 메뉴화면으로 돌아간다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	N/A

Activity 2041. Design Real Use Cases

Use case	18. Set Item Stock
Actor	Administrator
Purpose	음료의 재고를 조정한다.
Overview	관리자모드일 때, 관리자가 음료에 대한 재고를 조정할 수 있다.
Type	Primary
Cross Reference	System Functions: R4.1 R4.3 Use cases:
Pre-Requisites	관리자로 로그인이 된 상태여야 한다.
Typical Courses of Events	(A) :Administrator (S): System 1. (S) 재고를 조정할 수 있는 DVM의 음료 목록을 출력한다. 2. (A) 재고를 조정할 음료 버튼을 선택한다. 3. (A) 재고를 얼마만큼 조정할 것인지 개수를 선택한다. 4. (S) 관리자가 입력한 만큼 해당 음료의 재고를 조정한다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	

Activity 2041. Design Real Use Cases

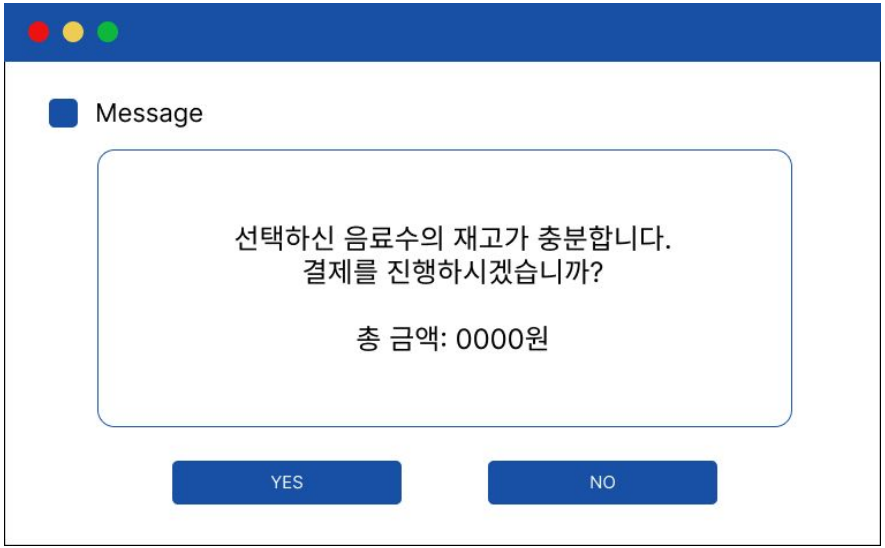
Use case	19. Set Item Price
Actor	Administrator
Purpose	음료에 대한 가격을 조정한다.
Overview	관리자모드일 때, 관리자가 음료에 대한 가격을 조정할 수 있다.
Type	Primary
Cross Reference	System Functions: R4.1 R4.4 Use cases:
Pre-Requisites	관리자로 로그인이 된 상태여야 한다.
Typical Courses of Events	(A) :Administrator (S): System 1. (S) 가격을 조정할 수 있는 DVM의 음료 목록을 출력한다. 2. (A) 가격을 조정할 음료 버튼을 선택한다. 3. (A) 가격을 얼마로 조정할 것인지 입력한다. (E1) 4. (S) 관리자가 입력한 만큼 해당 음료의 가격을 조정한다.
Alternatives Courses of Events	N/A
Exceptions Courses of Events	E1. 잘못된 형식의 가격 입력: 1. 관리자가 잘못된 형식의 음료 가격을 입력하면 오류 메시지를 띄우고 다시 음료 가격 입력 요청을 한다.

Activity 2042. Define Reports, UI, and Storyboards

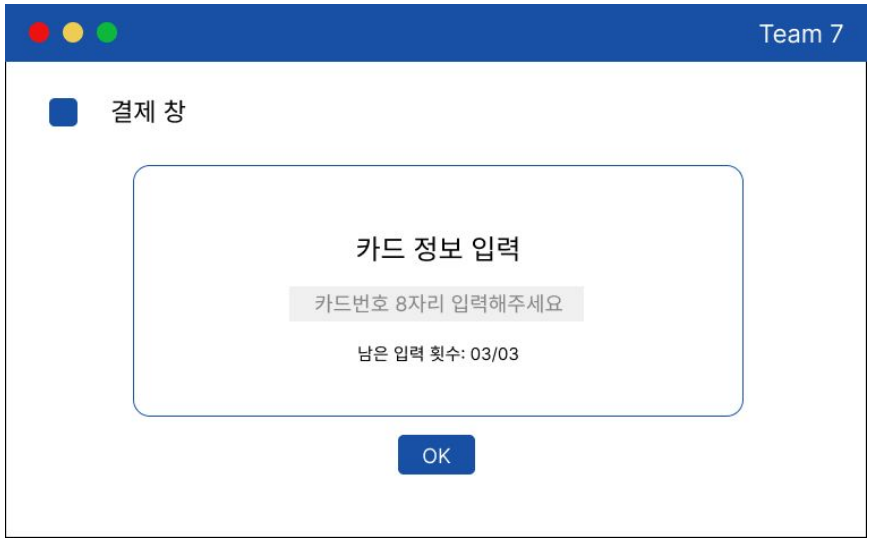
시스템 작동 시 메인 화면

[1. 음료 선택] 선택 시 음료 선택 화면

Activity 2042. Define Reports, UI, and Storyboards

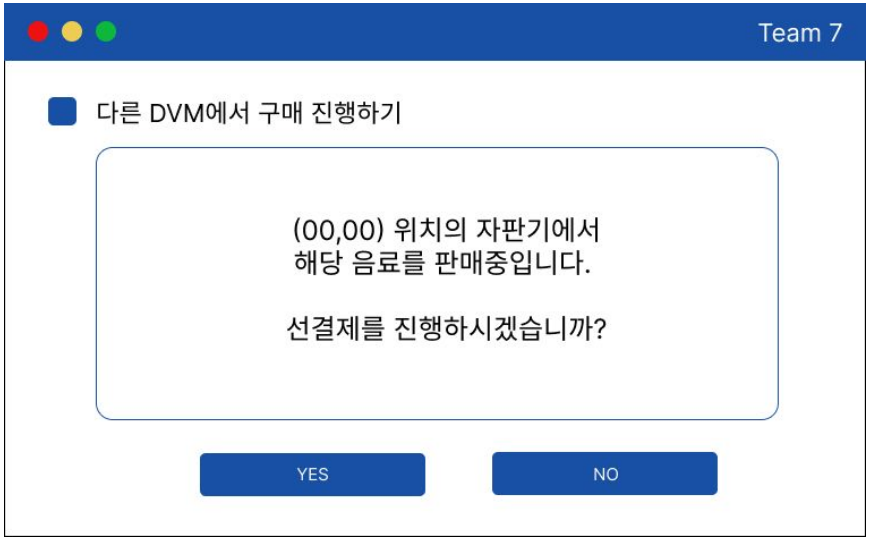


재고 충분 -> 결제 진행 여부

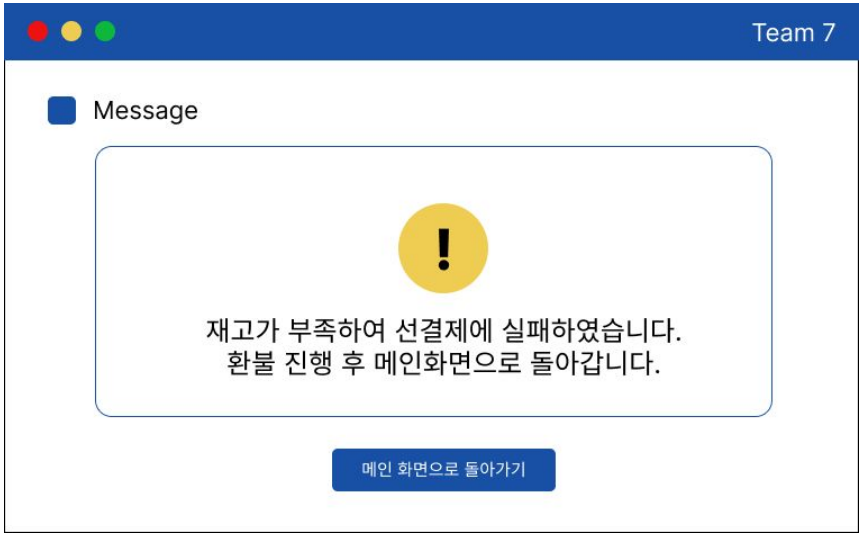


선택한 음료의 재고가 충분할 경우

Activity 2042. Define Reports, UI, and Storyboards

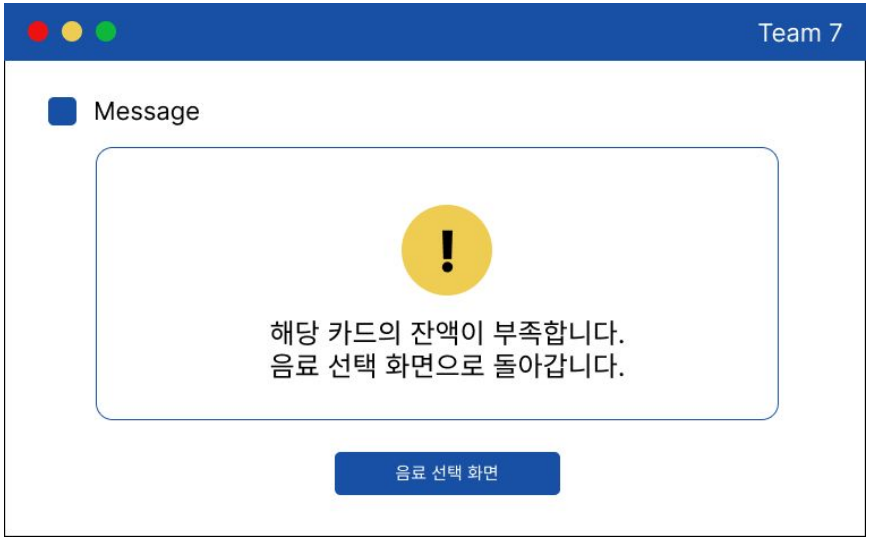


다른 DVM에서 구매할 경우

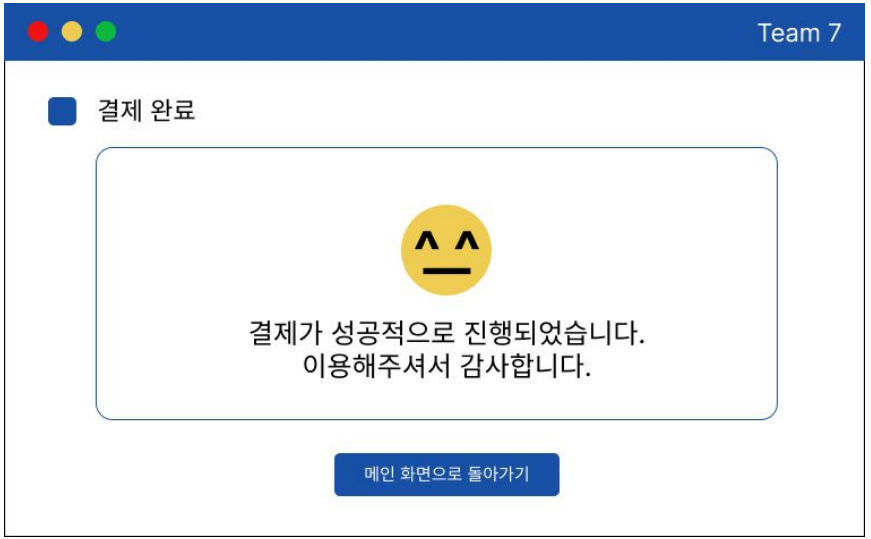


최종 선결제에 실패한 경우

Activity 2042. Define Reports, UI, and Storyboards

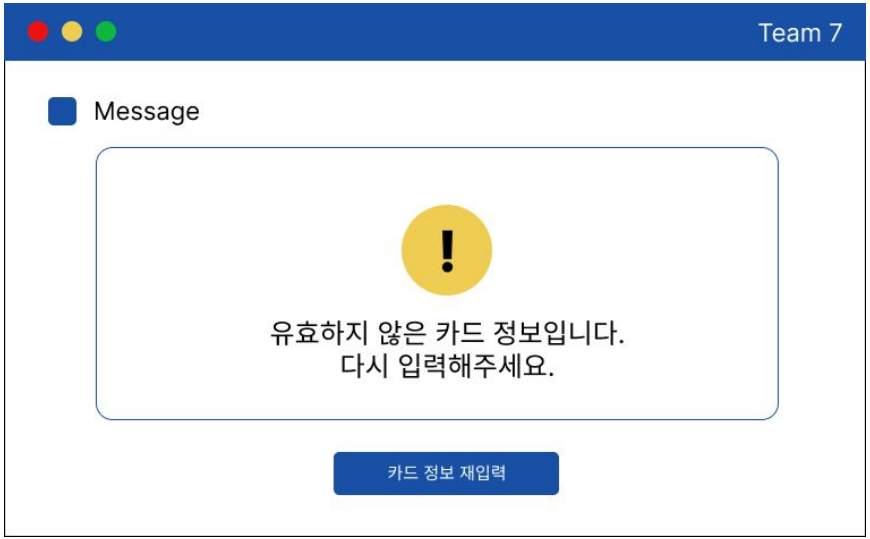


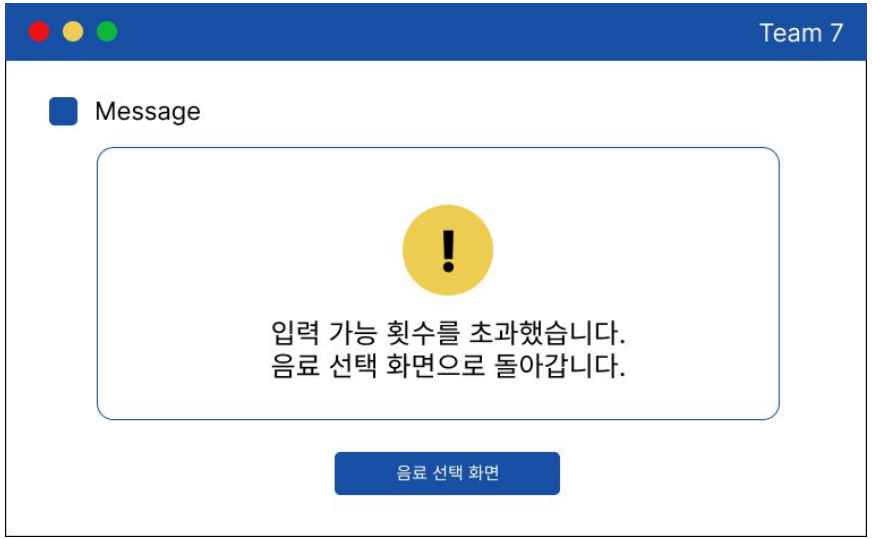
카드 정보가 유효한 경우 - 잔액 X



카드 정보가 유효한 경우 - 잔액 O
-> 결제완료

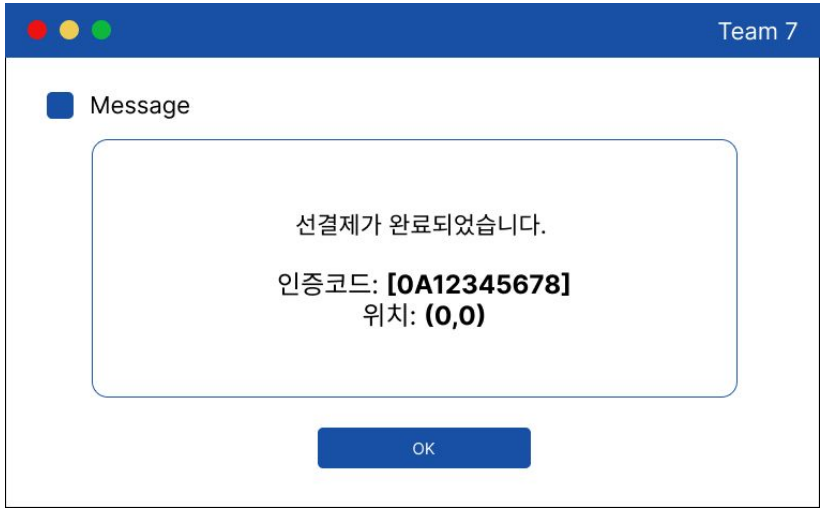
Activity 2042. Define Reports, UI, and Storyboards



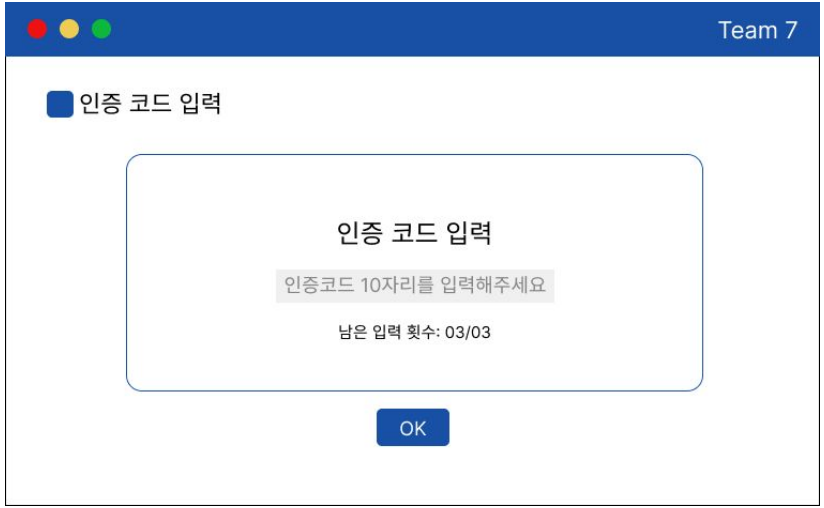


카드 정보가 유효하지 않을 경우

Activity 2042. Define Reports, UI, and Storyboards

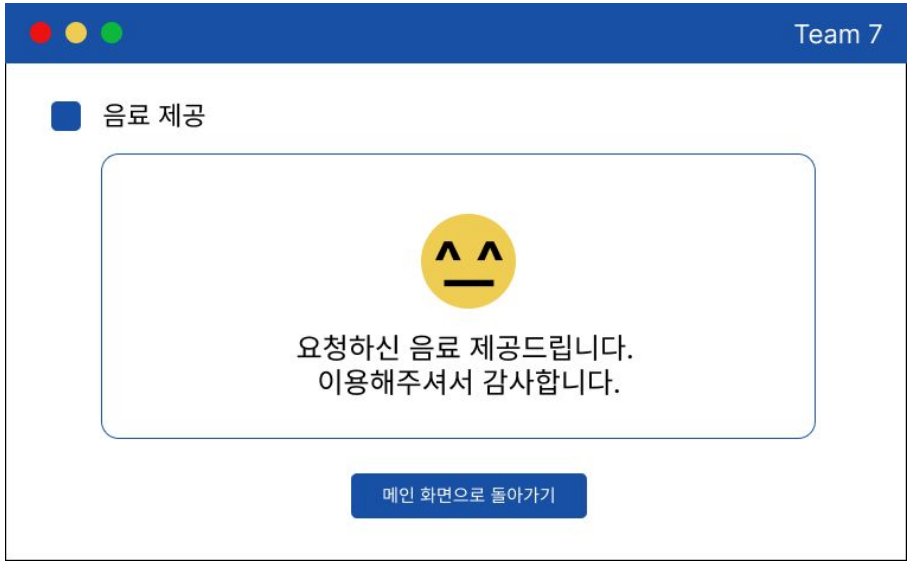


선결제
완료

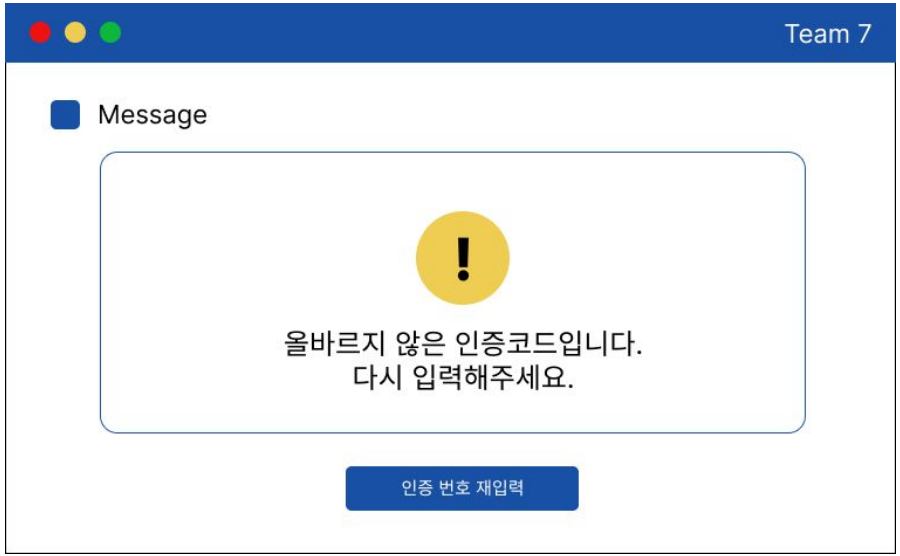


[2. 인증번호 입력] 선택 시
화면

Activity 2042. Define Reports, UI, and Storyboards

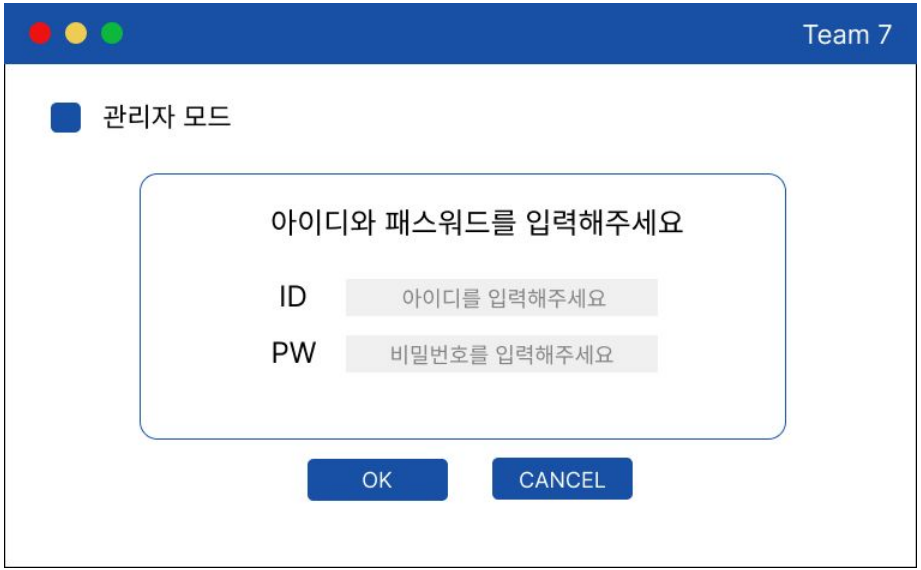


인증코드 인증 성공



인증코드 인증 실패

Activity 2042. Define Reports, UI, and Storyboards



[3. 관리자 모드] 선택 시 화면

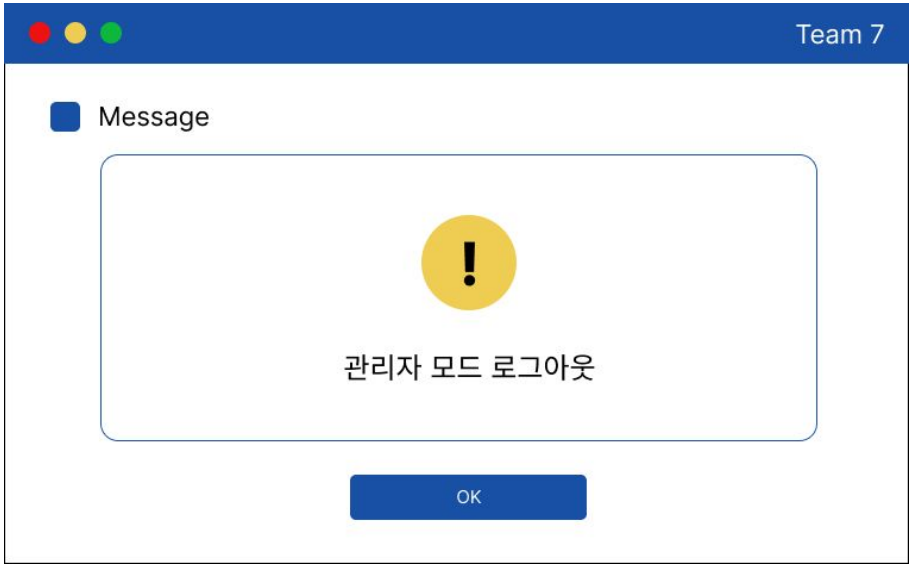
관리자 모드 메뉴 선택

Activity 2042. Define Reports, UI, and Storyboards

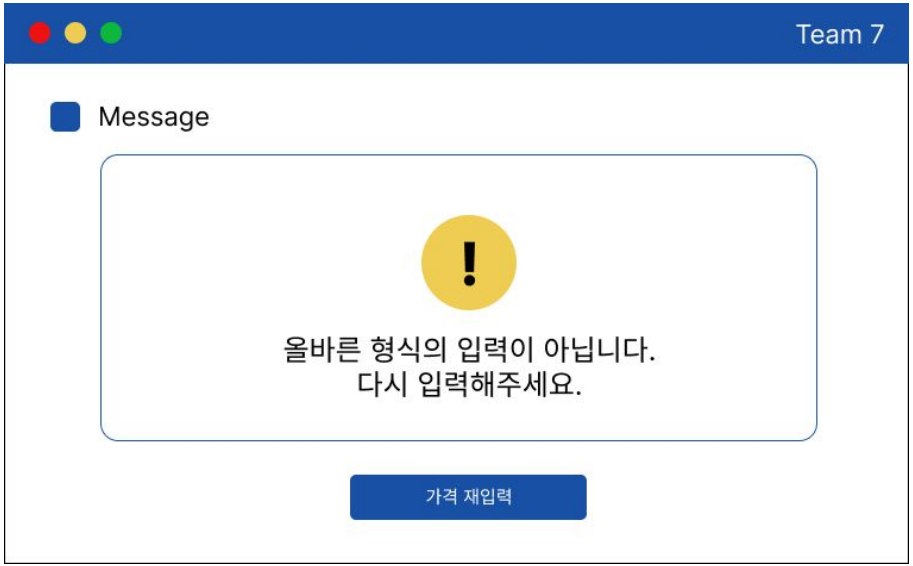
음료 재고 관리

음료 가격 관리

Activity 2042. Define Reports, UI, and Storyboards

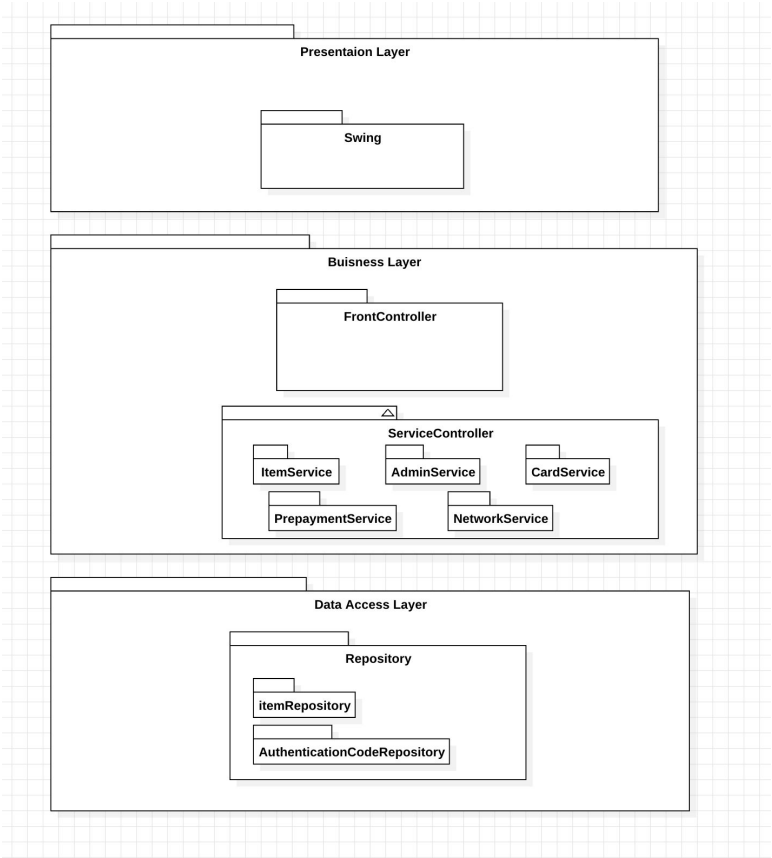


관리자 모드
로그아웃



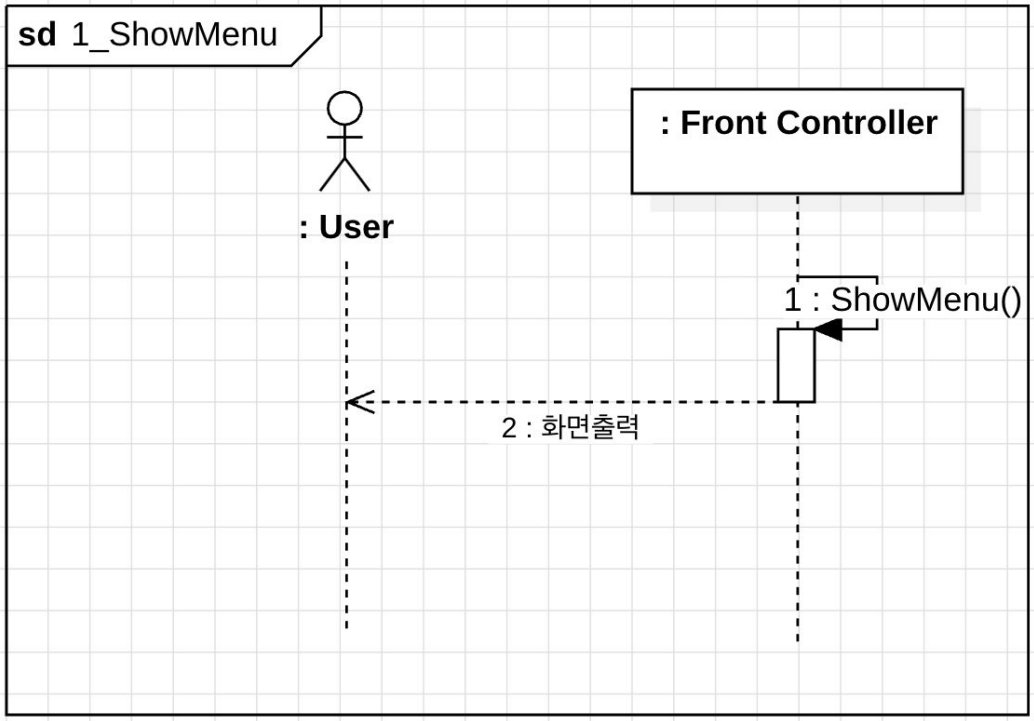
입력 형식 오류

Activity 2043. Refine System Architecture



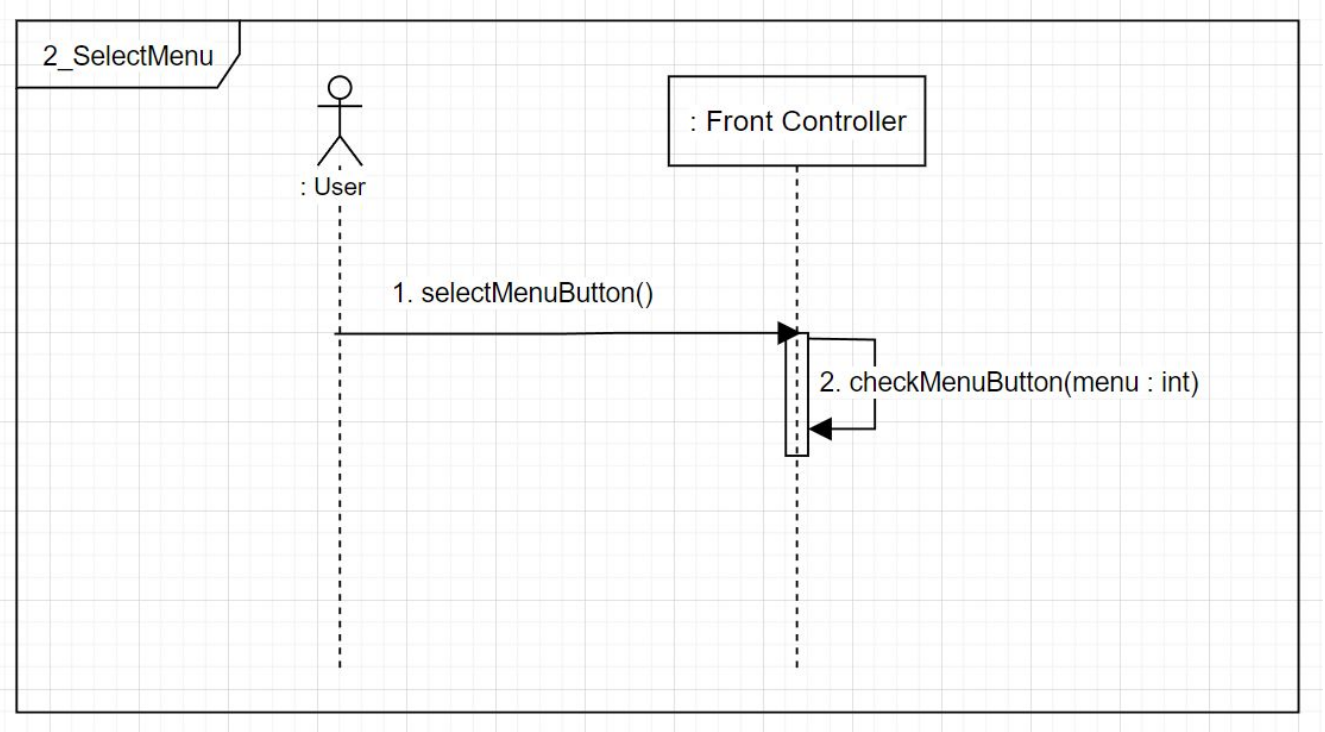
Activity 2044. Define Interaction Diagrams

Use Case 1: Show Menu



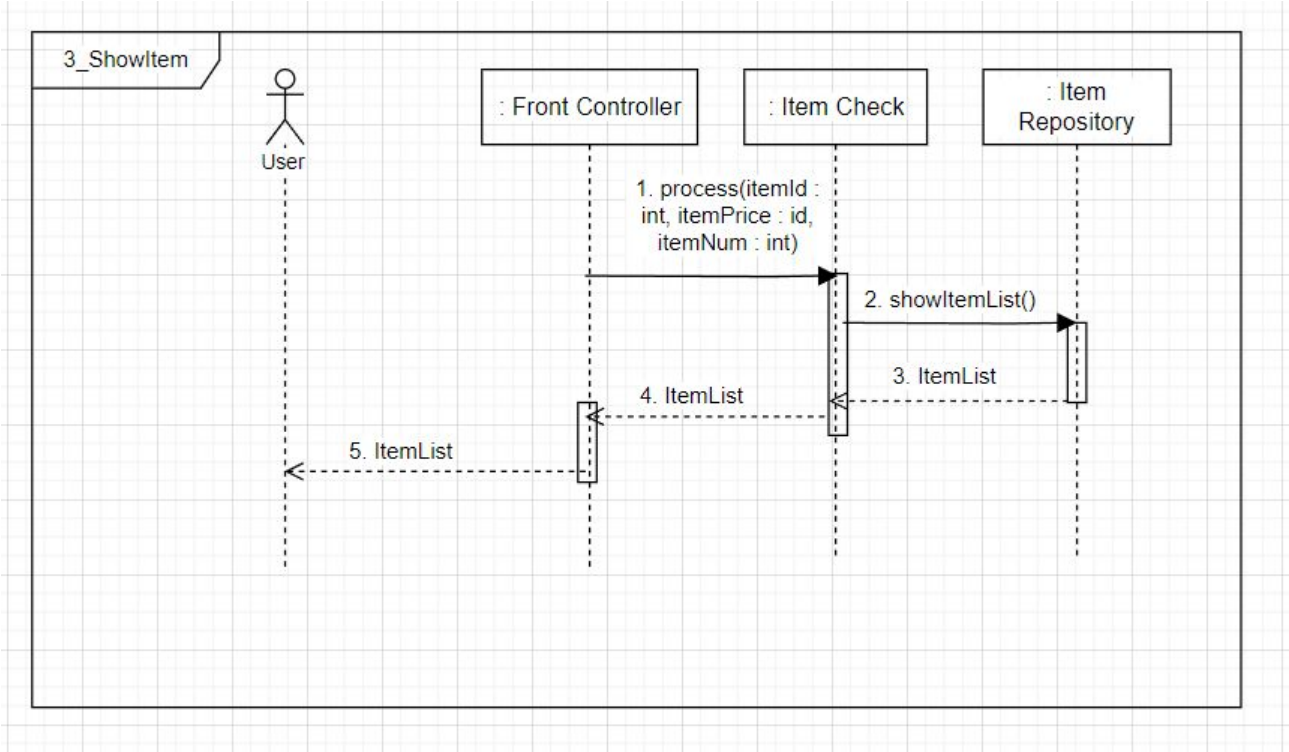
Activity 2044. Define Interaction Diagrams

Use Case 2: Select Menu



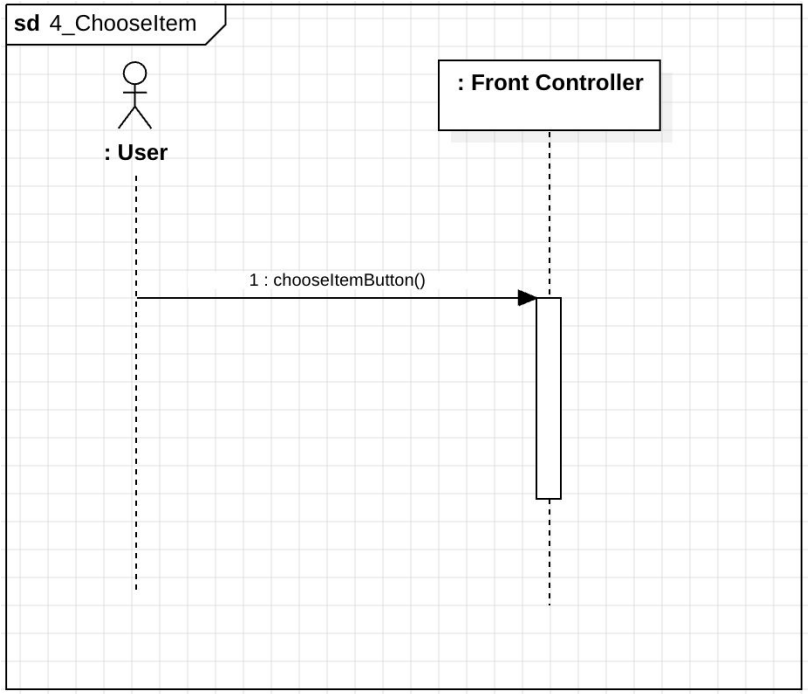
Activity 2044. Define Interaction Diagrams

Use Case 3 : Show Item

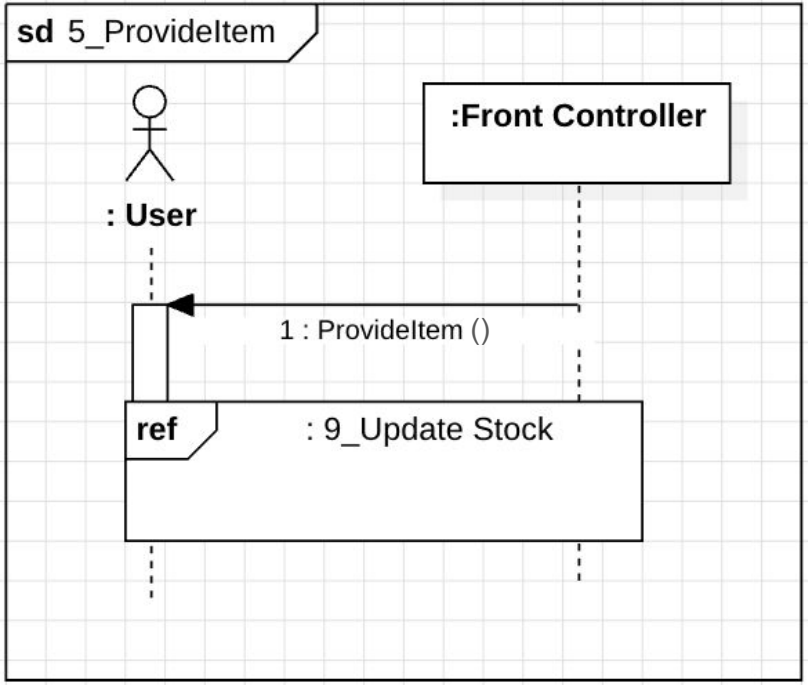


Activity 2044. Define Interaction Diagrams

Use Case 4: Choose Item

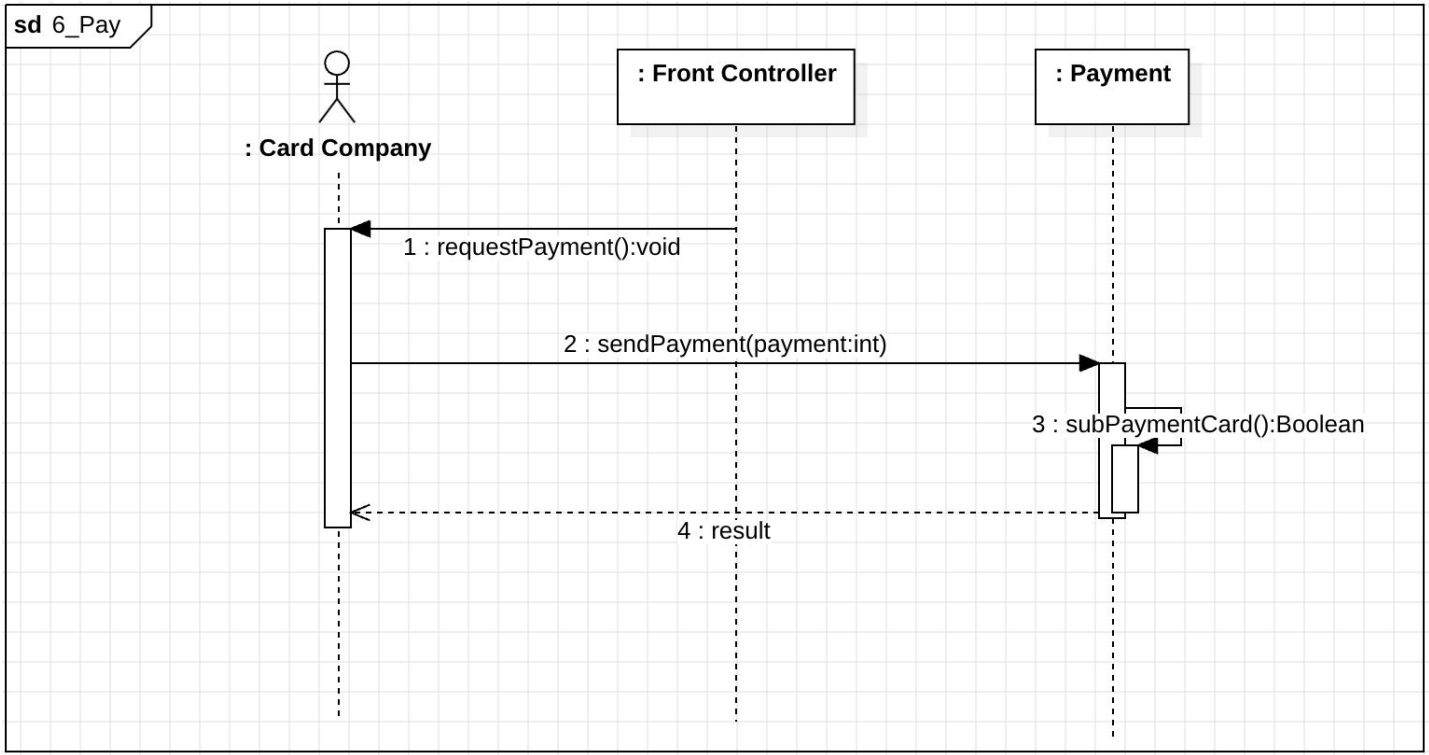


Use Case 5: Provide Item



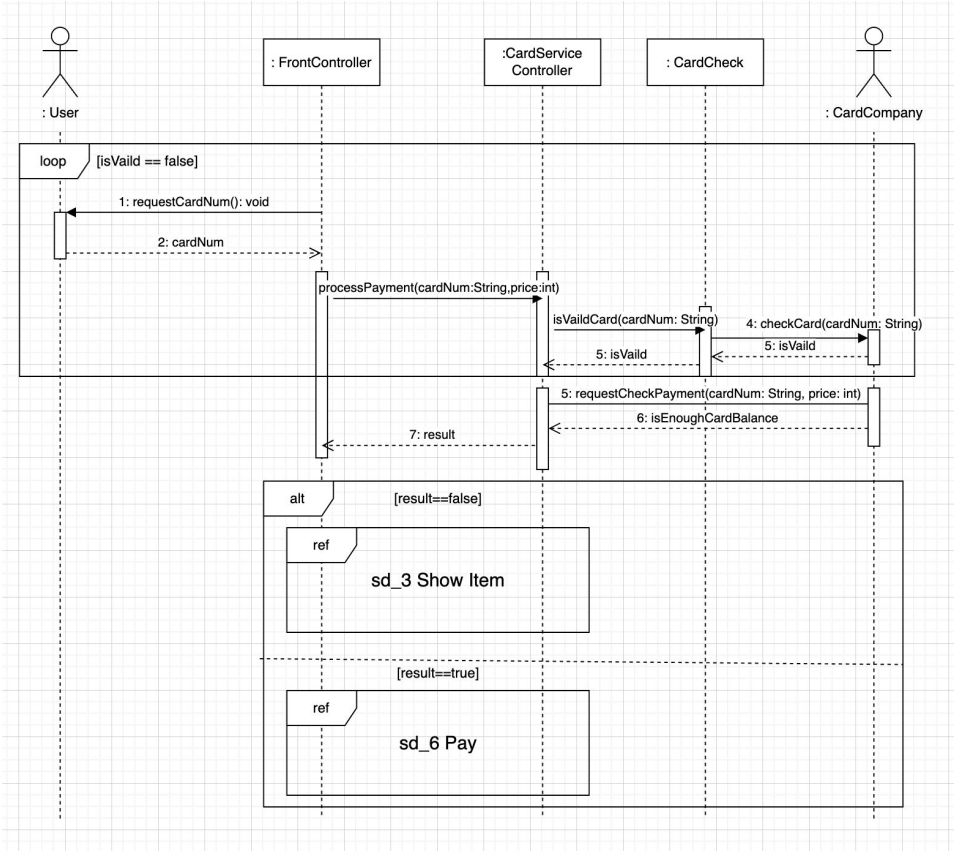
Activity 2044. Define Interaction Diagrams

Use Case 6: Pay



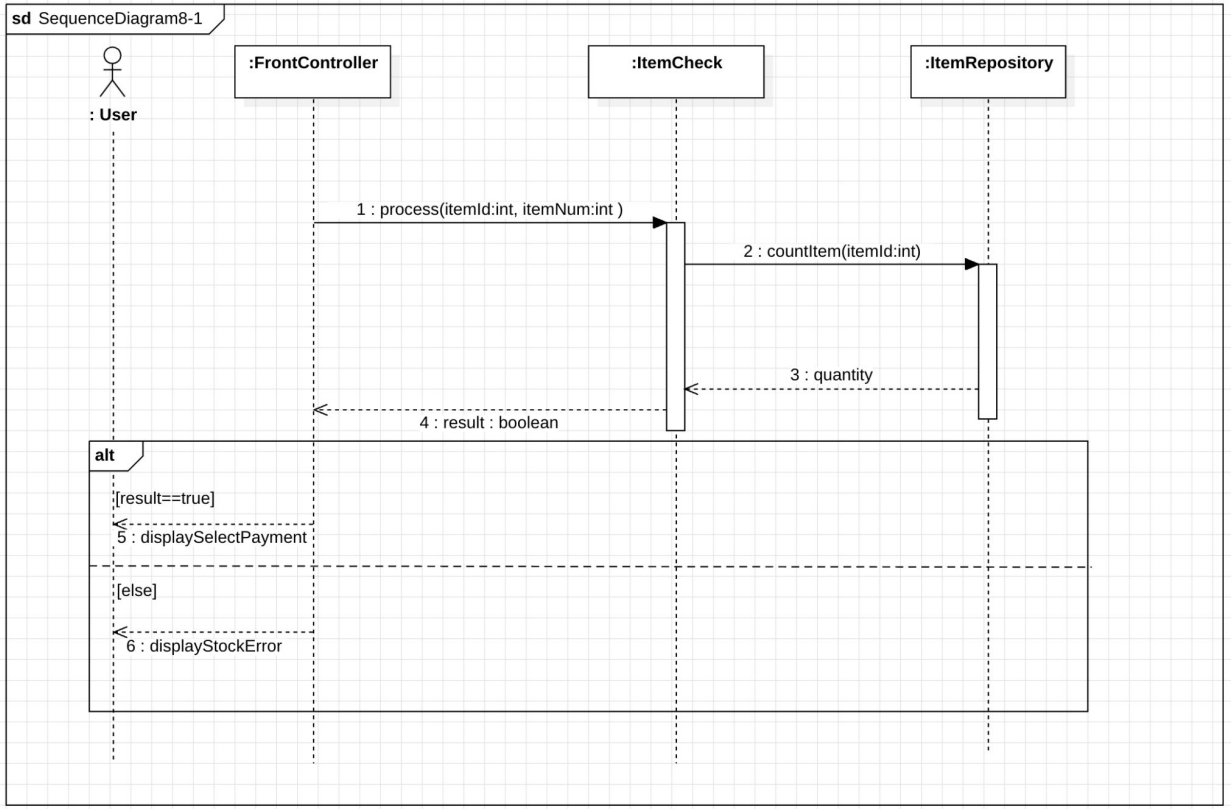
Activity 2044. Define Interaction Diagrams

Use Case 7: Valid Card



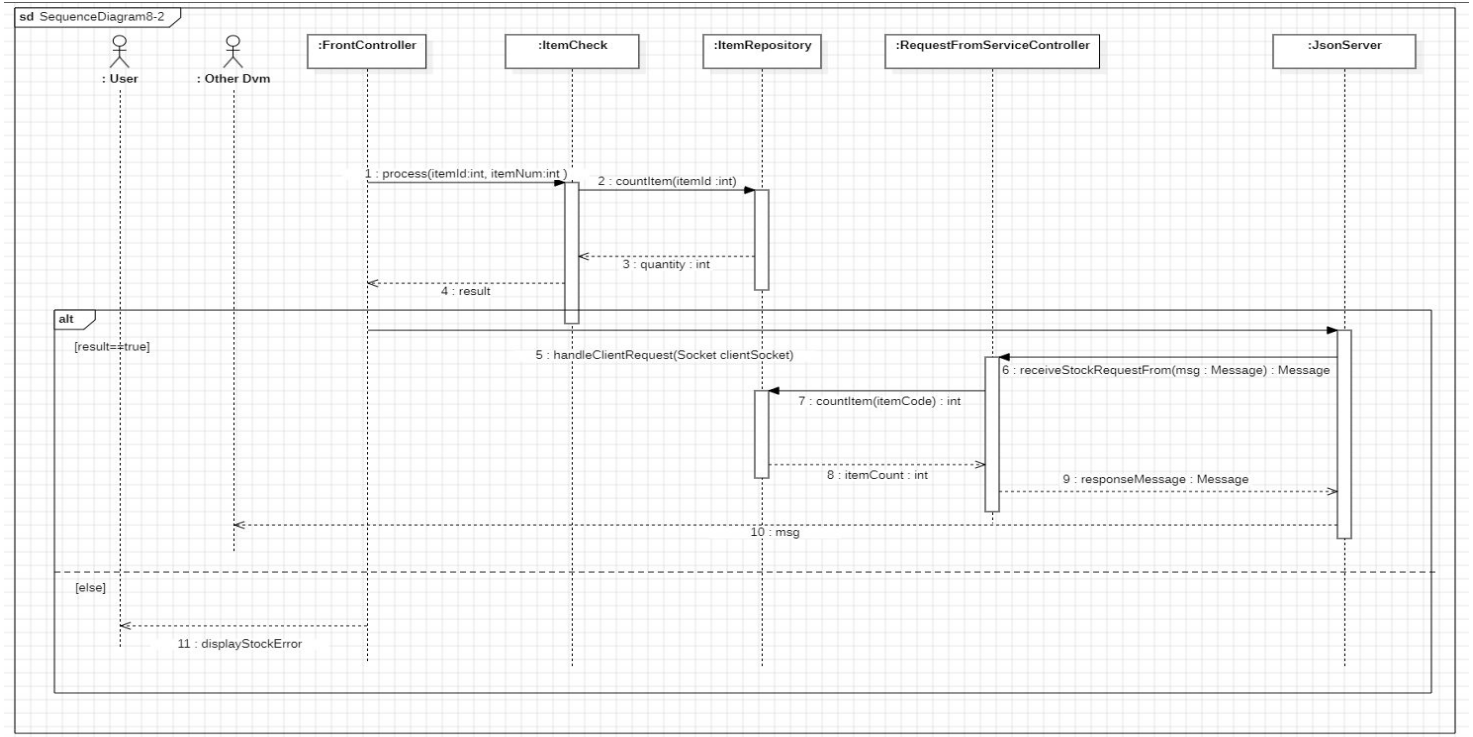
Activity 2044. Define Interaction Diagrams

Use Case 8: Check Stock



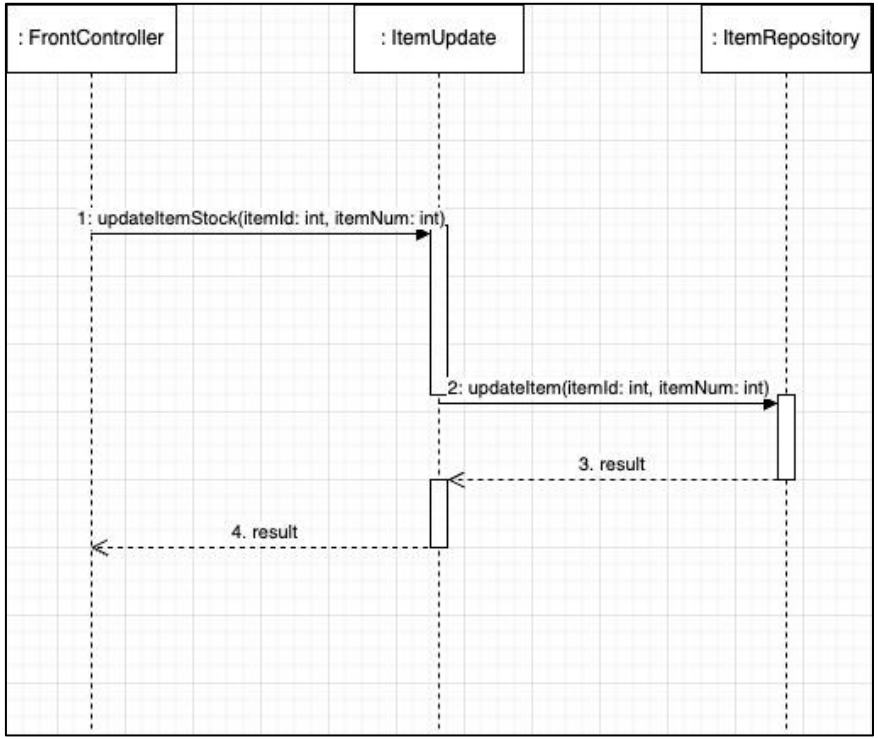
Activity 2044. Define Interaction Diagrams

Use Case 8: Check Stock



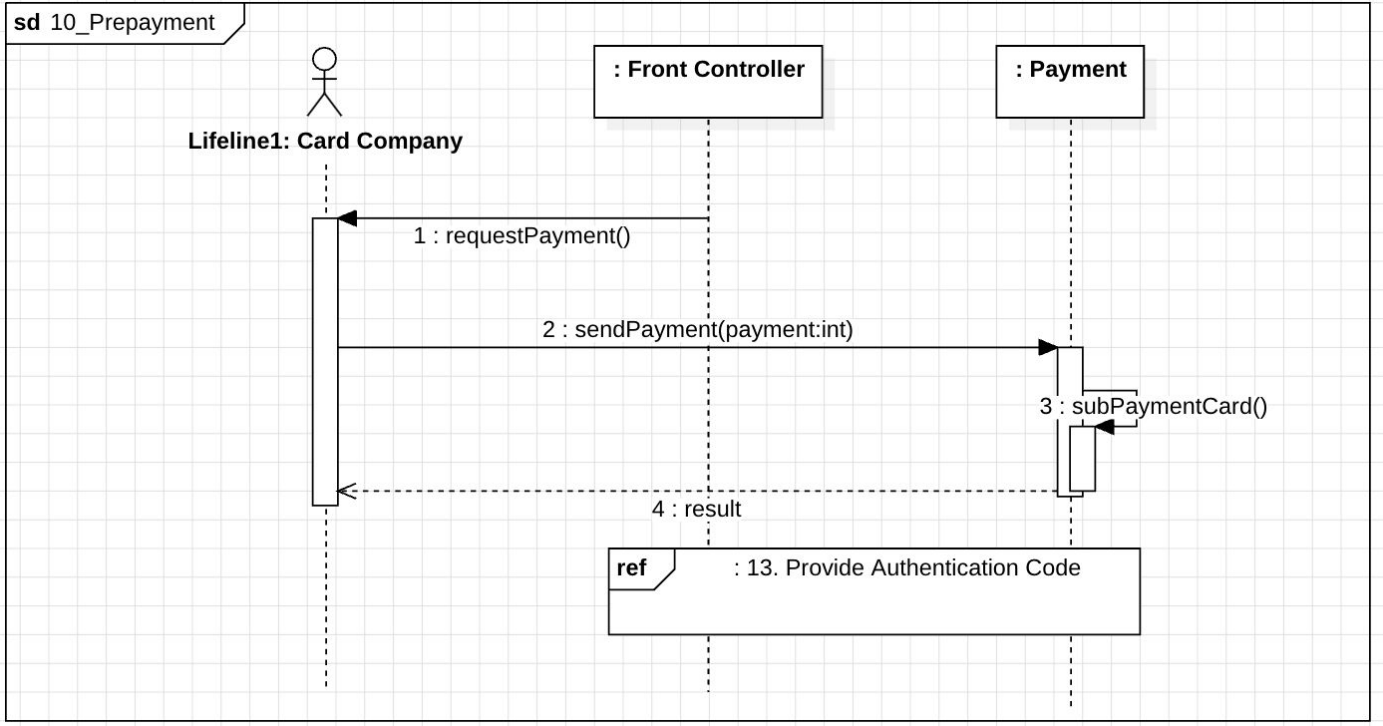
Activity 2044. Define Interaction Diagrams

Use Case 9: Update Stock



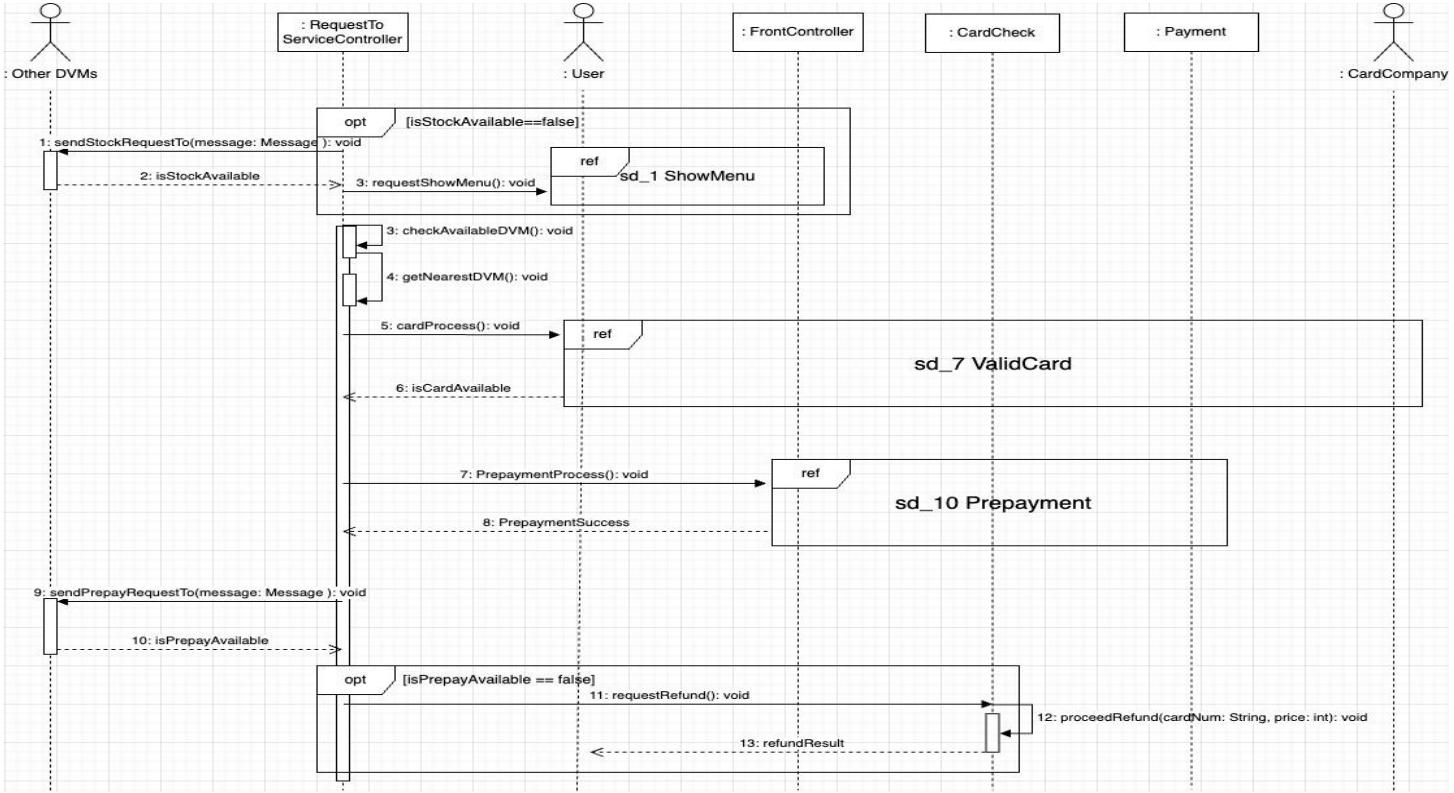
Activity 2044. Define Interaction Diagrams

Use Case 10: Prepayment



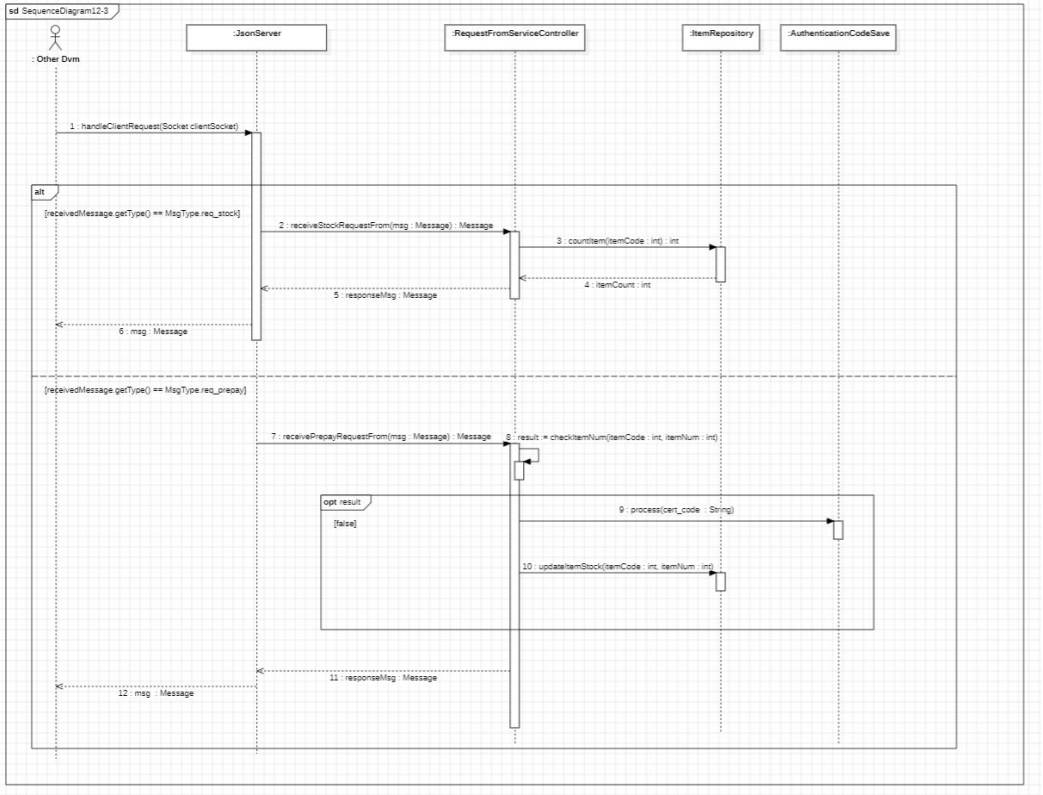
Activity 2044. Define Interaction Diagrams

Use Case 11: Check other DVMs



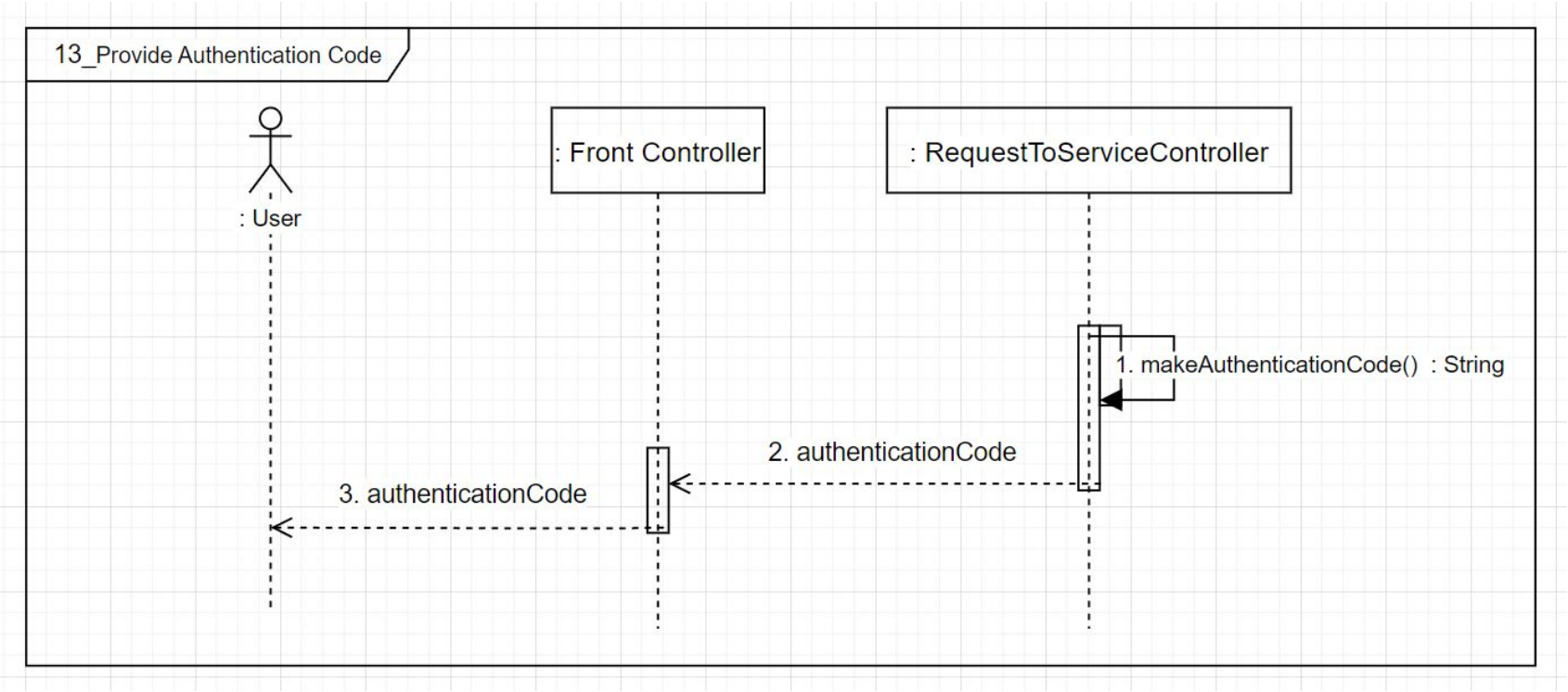
Activity 2044. Define Interaction Diagrams

Use Case 12: Request from other DVMs



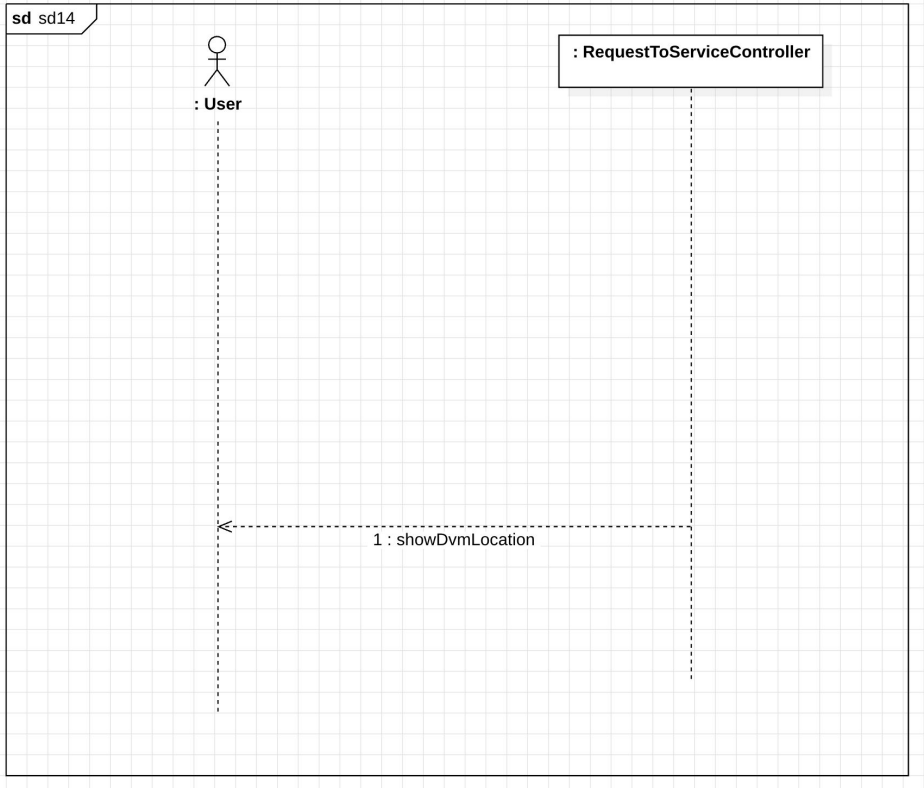
Activity 2044. Define Interaction Diagrams

Use Case 13: Provide Authentication Code



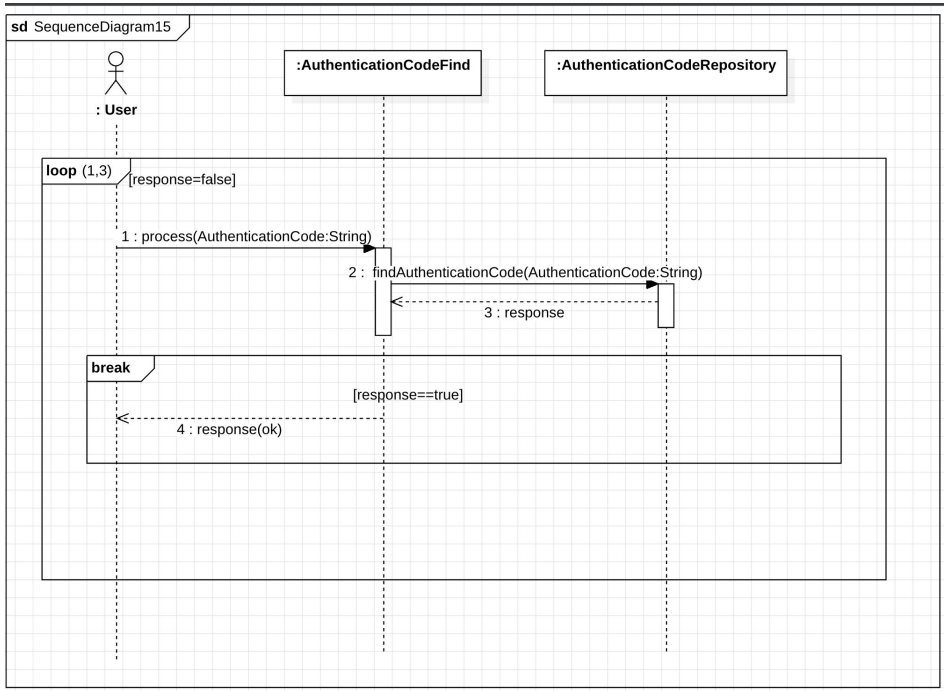
Activity 2044. Define Interaction Diagrams

Use Case 14: Provide Available DVM Location



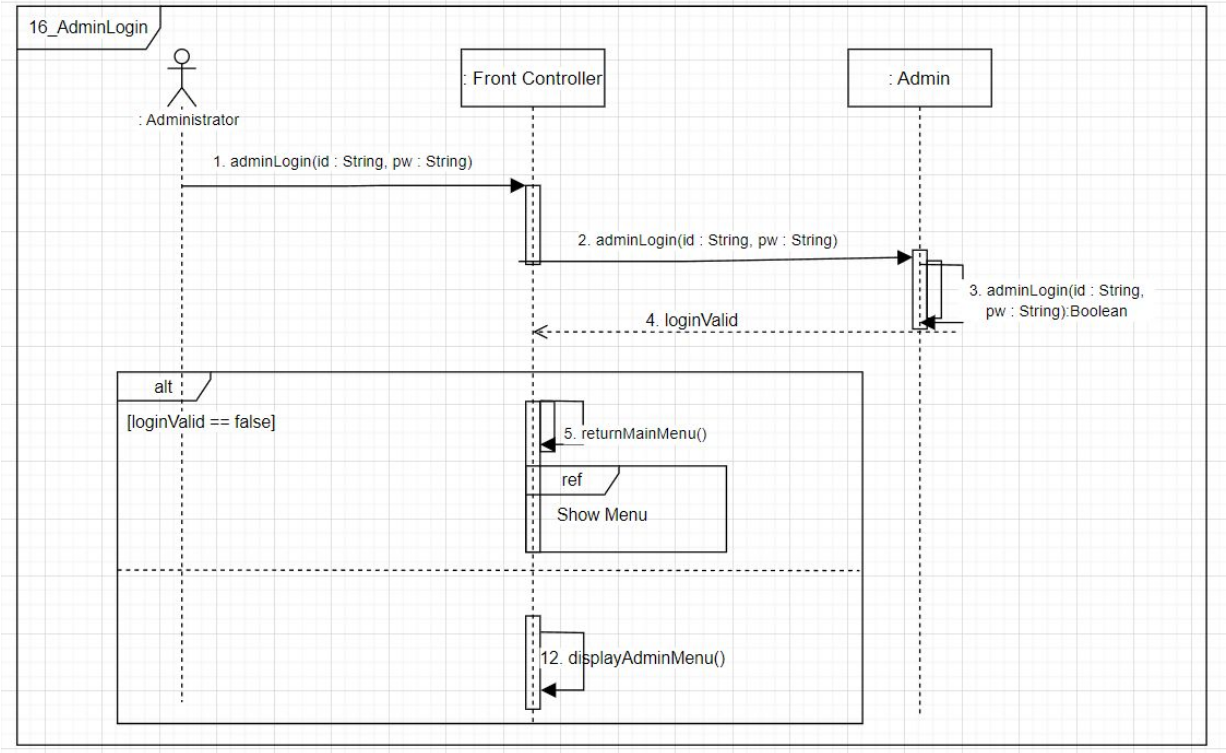
Activity 2044. Define Interaction Diagrams

Use Case 15: Check Authentication Code



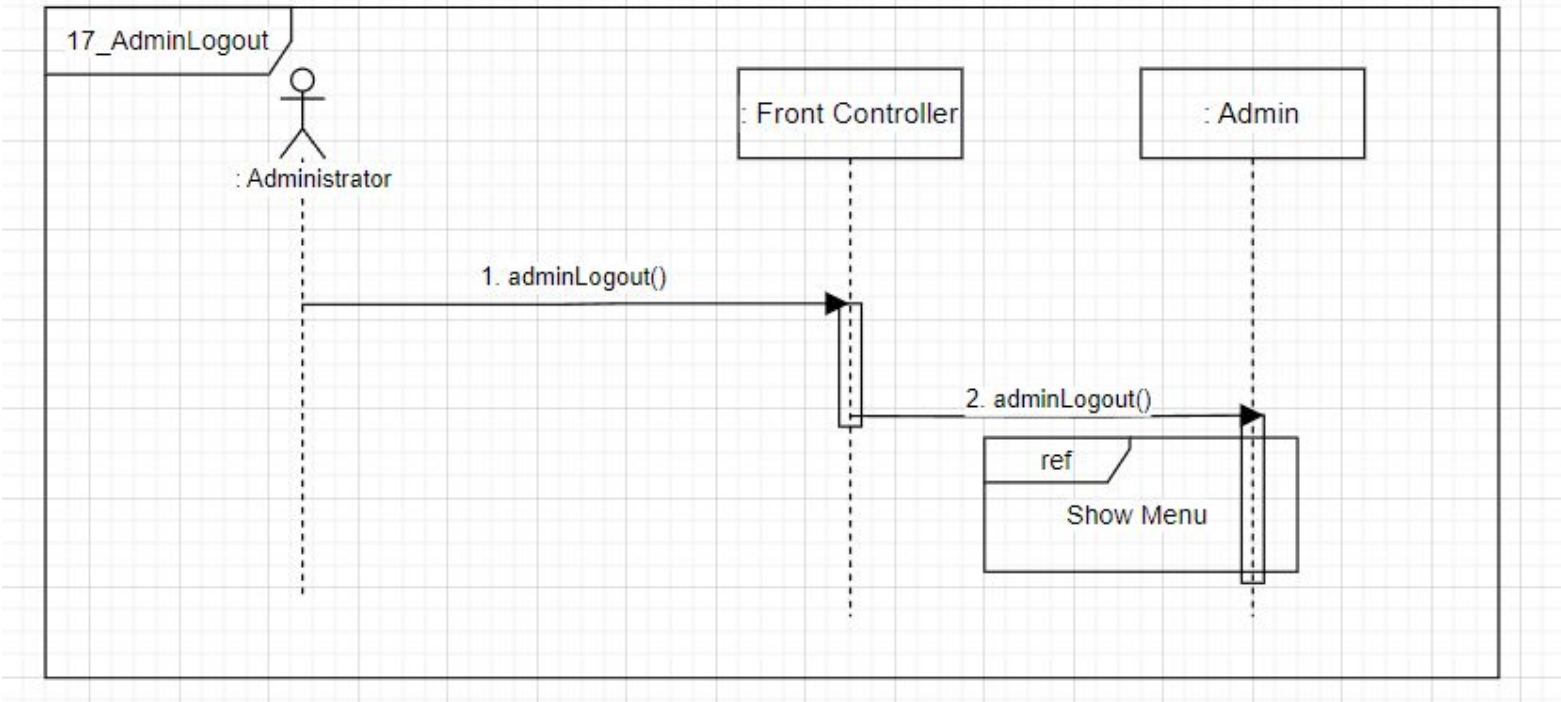
Activity 2044. Define Interaction Diagrams

Use Case 16: Admin Login



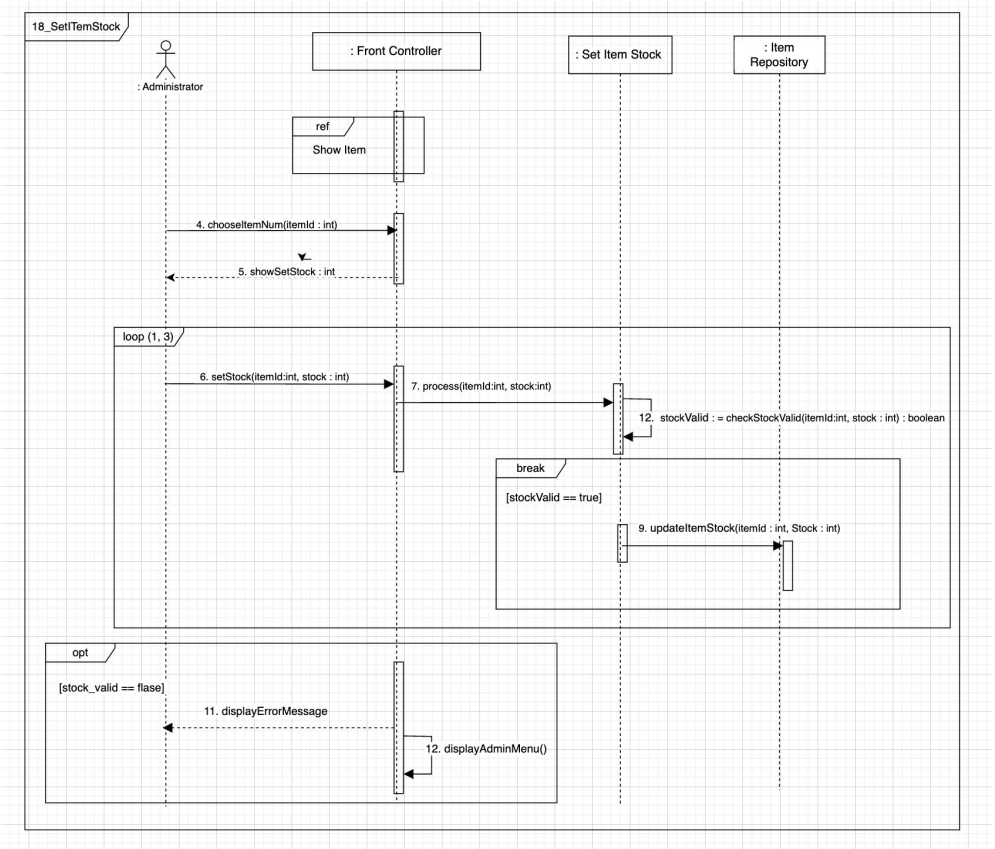
Activity 2044. Define Interaction Diagrams

Use Case 17: Admin Logout



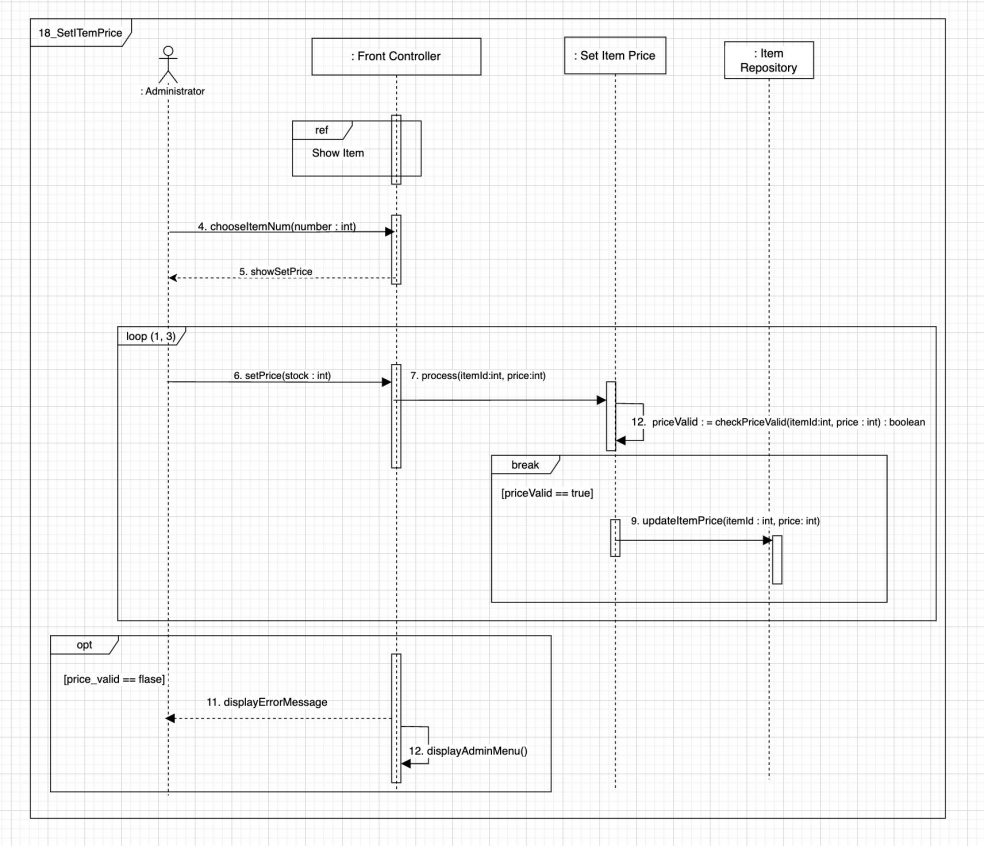
Activity 2044. Define Interaction Diagrams

Use Case 18: Set Item Stock

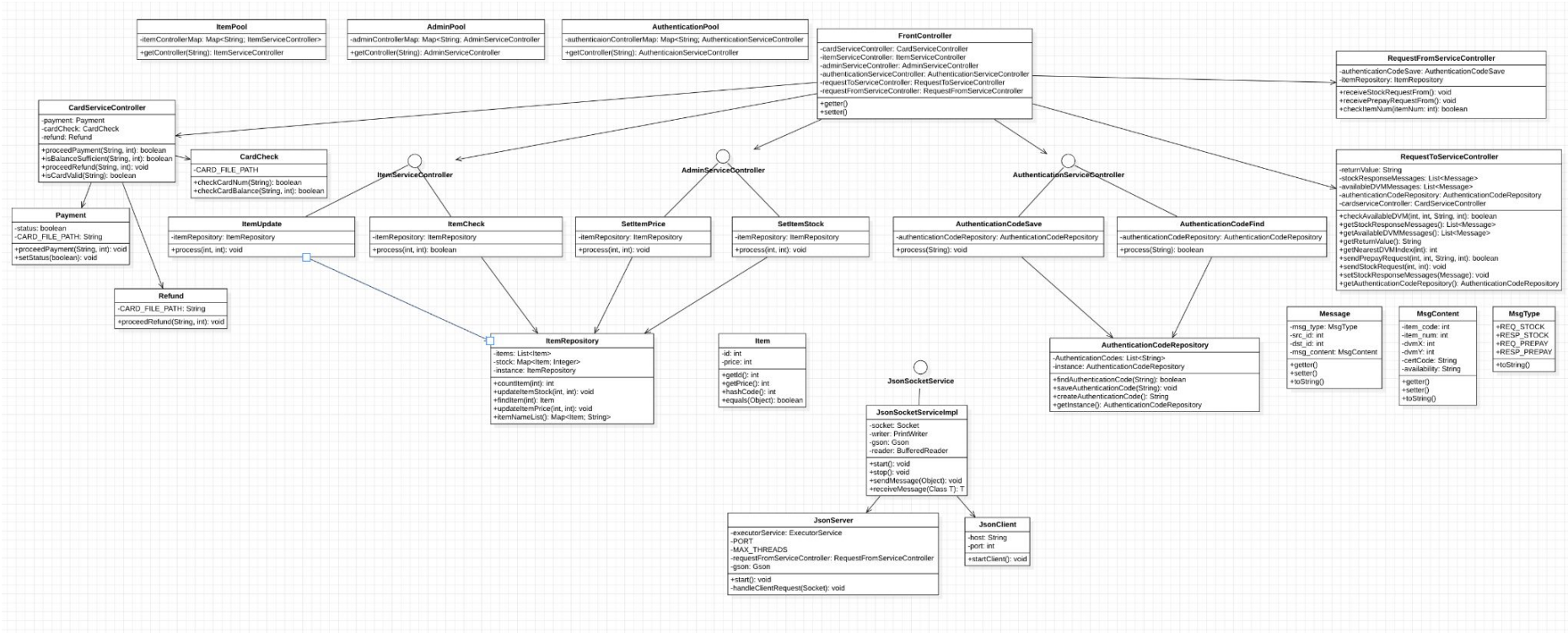


Activity 2044. Define Interaction Diagrams

Use Case 19: Set Item Price



Activity 2045. Define Design Class Diagrams

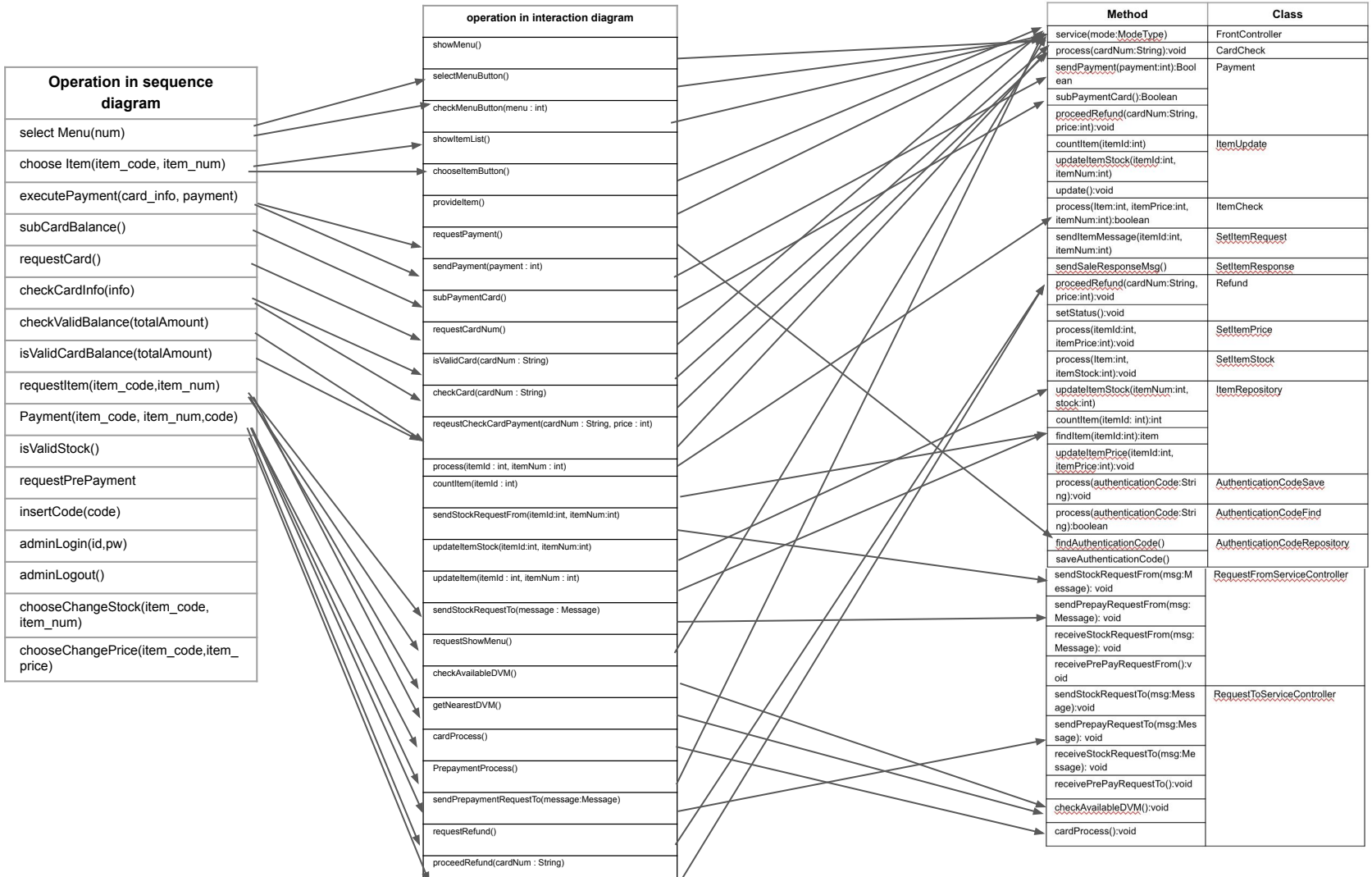


Activity 2046. Design Traceability Analysis

Ref.	System Function
R 1.1	Show Menu
R 1.2	Select Menu
R 1.3	Show Item
R 1.4	Choose Item
R 1.5	Provide Item
R 2.1	Pay
R 2.2	Valid Card
R 2.3	Check Stock
R 2.4	Update Stock
R 3.1	Prepay
R 3.2	Check other DVMs
R 3.3	Request from others DVMs
R 3.3	Provide Authentication Code
R 3.4	Provide Available DVM Location
R 3.5	Check Authentication Code
R 4.1	Admin Login
R 4.2	Admin Logout
R 4.3	Set Item Stock
R 4.4	Set Item Price

num	UseCase
1	Show Menu
2	Select Menu
3	Show Item
4	Choose Item
5	Provide Item
6	Pay
7	Valid Card
8	Check Stock
9	Update Stock
10	Prepay
11	Check other DVMs
12	Request from others DVMs
13	Provide Authentication Code
14	Provide Available DVM Location
15	Check Authentication Code
16	Admin Login
17	Admin Logout
18	Set Item Stock
19	Set Item Price

Operation in sequence diagram
select Menu(num)
choose Item(item_code, item_num)
executePayment(card_info, payment)
subCardBalance()
requestCard()
cardInfo(info)
checkCardInfo(info)
checkValidBalance(totalAmount)
isValidCardBalance(totalAmount)
executePayment(card_info.payment)
subCardBalance()
requestItem(item_code,item_num)
requestPrePayment(item_code, item_num,code)
isValidStock()
requestPrePayment
insertCode(code)
adminLogin(id,pw)
adminLogout()
chooseChangeStock(item_code, item_num)
chooseChangePrice(item_code,item_price)


Operation in sequence diagram
select Menu(num)
choose Item(item_code, item_num)
executePayment(card_info, payment)
subCardBalance()
requestCard()
checkCardInfo(info)
checkValidBalance(totalAmount)
isValidCardBalance(totalAmount)
requestItem(item_code,item_num)
Payment(item_code, item_num,code)
isValidStock()
requestPrePayment
insertCode(code)
adminLogin(id,pw)
adminLogout()
chooseChangeStock(item_code, item_num)
chooseChangePrice(item_code,item_price)
operation in interaction diagram
showMenu()
selectMenuButton()
checkMenuButton(menu : int)
showItemList()
chooseItemButton()
provideItem()
requestPayment()
sendPayment(payment : int)
subPaymentCard()
requestCardNum()
isValidCard(cardNum : String)
checkCard(cardNum : String)
reqeustCheckCardPayment(cardNum : String, price : int)
process(itemId : int, itemNum : int)
countItem(itemId : int)
sendStockRequestFrom(itemId:int, itemNum:int)
updateItemStock(itemId:int, itemNum:int)
updateItem(itemId : int, itemNum : int)
sendStockRequestTo(message : Message)
requestShowMenu()
checkAvailableDVM()
getNearestDVM()
cardProcess()
PrepaymentProcess()
sendPrepaymentRequestTo(message:Message)
requestRefund()
proceedRefund(cardNum : String)
Method
Class
service(mode:ModeType)
FrontController
process(cardNum:String):void
CardCheck
sendPayment(payment:int):Boolean
Payment
subPaymentCard():Boolean
proceedRefund(cardNum:String, price:int):void
countItem(itemId:int)
ItemUpdate
updateItemStock(itemId:int, itemNum:int)
update():void
process(Item:int, itemPrice:int, itemNum:int):boolean
ItemCheck
sendItemMessage(itemId:int, itemNum:int)
SetItemRequest
sendSaleResponseMsg()
SetItemResponse
proceedRefund(cardNum:String, price:int):void
Refund
setStatus():void
process(itemId:int, itemPrice:int):void
SetItemPrice
process(Item:int, itemStock:int):void
SetItemStock
updateItemStock(itemNum:int, stock:int)
ItemRepository
countItem(itemId: int):int
findItem(itemId:int):item
updateItemPrice(itemId:int, itemPrice:int):void
process(authenticationCode:String):void
AuthenticationCodeSave
process(authenticationCode:String):boolean
AuthenticationCodeFind
findAuthenticationCode()
AuthenticationCodeRepository
saveAuthenticationCode()
sendStockRequestFrom(msg:Message): void
RequestFromServiceController
sendPrepayRequestFrom(msg:Message): void
receiveStockRequestFrom(msg:Message): void
receivePrePayRequestFrom():void
sendStockRequestTo(msg:Message):void
RequestToServiceController
sendPrepayRequestTo(msg:Message): void
receiveStockRequestTo(msg:Message): void
receivePrePayRequestTo():void
checkAvailableDVM():void
cardProcess():void

Operation in sequence diagram
select Menu(num)
choose Item(item_code, item_num)
executePayment(card_info, payment)
subCardBalance()
requestCard()
cardInfo(info)
checkCardInfo(info)
checkValidBalance(totalAmount)
isValidCardBalance(totalAmount)
executePayment(card_info.payment)
subCardBalance()
requestItem(item_code,item_num)
requestPrePayment(item_code, item_num,code)
isValidStock()
requestPrePayment
insertCode(code)
adminLogin(id,pw)
adminLogout()
chooseChangeStock(item_code, item_num)
chooseChangePrice(item_code,item_price)

operation in interaction diagram
receiveStockRequestFrom()
process(itemId : int, itemNum : int)
receiveprepayRequestFrom()
checkItemNum(itemNum : int)
saveAuthenticationCode(AuthenticationCode : String)
makeAuthenticationCode()
process(AuthenticationCode : String)
findAuthenticationCode : String
adminLogin(id : String, pw : String)
returnMainMenu()
displayAdminMenu()
adminLogout()
chooseItemNum(number : int)
setStock()

Method	Class
service(mode:ModeType)	FrontController
process(cardNum:String):void	CardCheck
sendPayment(payment:int):Boolean	Payment
subPaymentCard():Boolean	
proceedRefund(cardNum:String, price:int):void	
countItem(itemId:int)	ItemUpdate
updateItemStock(itemId:int, itemNum:int)	
update():void	
process(Item:int, itemPrice:int, itemNum:int):boolean	ItemCheck
sendItemMessage(itemId:int, itemNum:int)	SetItemRequest
sendSaleResponseMsg()	SetItemResponse
proceedRefund(cardNum:String, price:int):void	Refund
setStatus():void	
process(itemId:int, itemPrice:int):void	SetItemPrice
process(Item:int, itemStock:int):void	SetItemStock
updateItemStock(itemNum:int, stock:int)	ItemRepository
countItem(itemId: int):int	
findItem(itemId:int):item	
updateItemPrice(itemId:int, itemPrice:int):void	
process(authenticationCode:String):void	AuthenticationCodeSave
process(authenticationCode:String):boolean	AuthenticationCodeFind
findAuthenticationCode()	AuthenticationCodeRepository
saveAuthenticationCode()	
sendStockRequestFrom(msg:Message): void	RequestFromServiceController
sendPrepayRequestFrom(msg:Message): void	
receiveStockRequestFrom(msg:Message): void	
receivePrePayRequestFrom():void	
sendStockRequestTo(msg:Message):void	RequestToServiceController
sendPrepayRequestTo(msg:Message): void	
receiveStockRequestTo(msg:Message): void	
receivePrePayRequestTo():void	
checkAvailableDVM():void	
cardProcess():void	