Unit Testing Plan

for Point Of Sale System

- **Test Plan**
- **Test Design Specification**
- **Test Cases Specification**

Project Team

T6

Date

2016-11-06

Team Information

201311264 김병식

201610379 김나연

201611248 강병성

201610070 김지우

Table of Contents

1 Introduction

1.1 Objectives

본 문서는 2017년 건국대학교의 소프트웨어공학 개론 강의의 실습과제를 설명한다. 실습 과제는 Point Of Sale (POS) System을 소프트웨어만을 이용한 가상의 시스템을 unit 단위로 구현하여 unit testing 하기 위한 계획 문서이다. Test 수행을 위한 testing Pass/Fail Criteria를 정의하고 이를 수행하기 위한 test design & test cases를 제작한다.

1.2 Background

(1) 개발 시스템 : POS System이란 판매와 관련한 데이터를 일괄적으로 관리하고, 고객 정보를 수집하여 부가가치를 향상시키는 시스템이다.

(2) Unit test : Unit test는 시스템을 구성하는 최소 단위 모듈들을 대상으로 하는 test이며, 시스템의 성 능을 좌우하는 요소들이 요구사항을 만족하는지를 확인할 수 있는 기본적인 Test approach이다.

1.3 Scope

Point Of Sale (POS) System에 대한 unit test 수행을 위한 자원, 절차, 접근, 기술, 그리고 환경 및 도구 등을 정의한다. Unit test는 SASD 단계에서 디자인된 시스템 기능 관련 데이터 프로세스나 컨트롤러에 중점을 두며, SASD 단계에서 명세하지 않기로 정한 구체적인 파일 시스템을 포함한 외부 기기로부터의 입출력과 같은 기능, 외부 시스템에서 동작하는 관련 프로그램(포스기 서버 프로그램) 등은 test 대상에서 제외한다.

1.4 Project plan

1.5 Configuration management plan

1.6 References

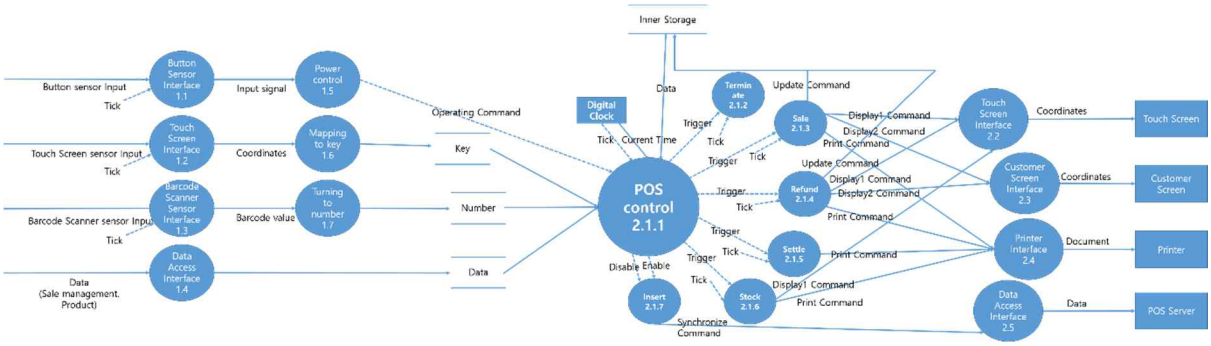
[2017SE_B][TP#2]SRA_T6_ver3

[2017SE_B][TP#2]SDS_T6_ver2

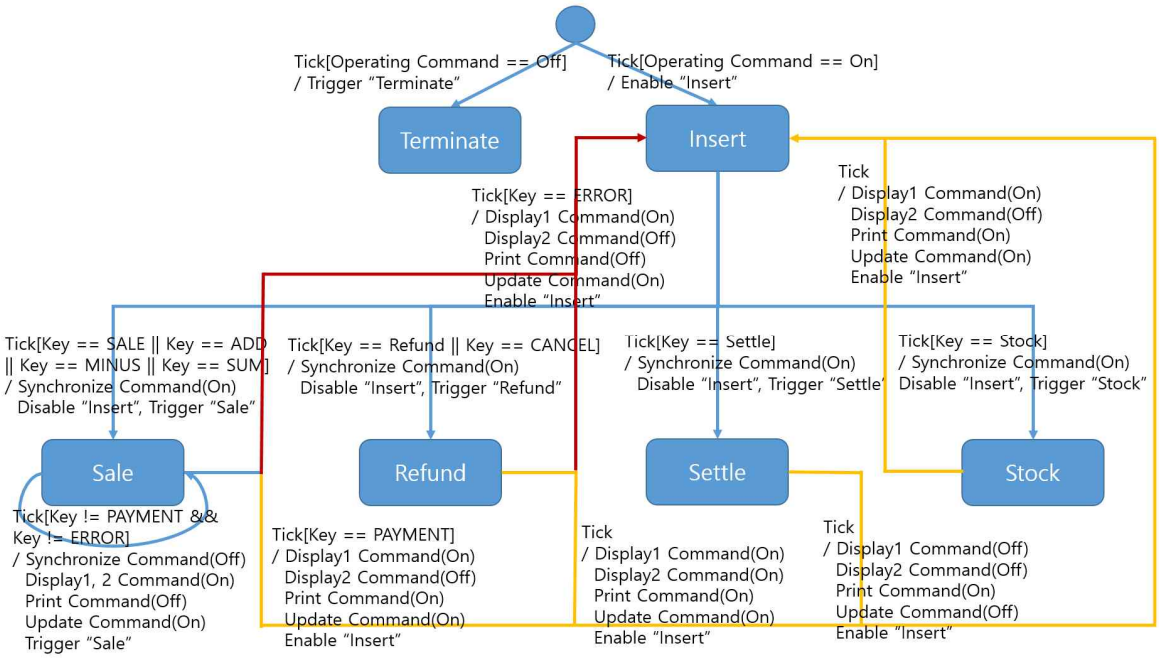
2 Test items

SASD 단계에서 디자인된 Point Of Sale (POS) System을 testing한다. SA에서 명세한대로 data process가 요구사항을 만족하는지, 정상적인 입출력 결과가 나오는지, SRS문서에 명시된 예외처리가 잘 동작하는지 testing을 수행한다.

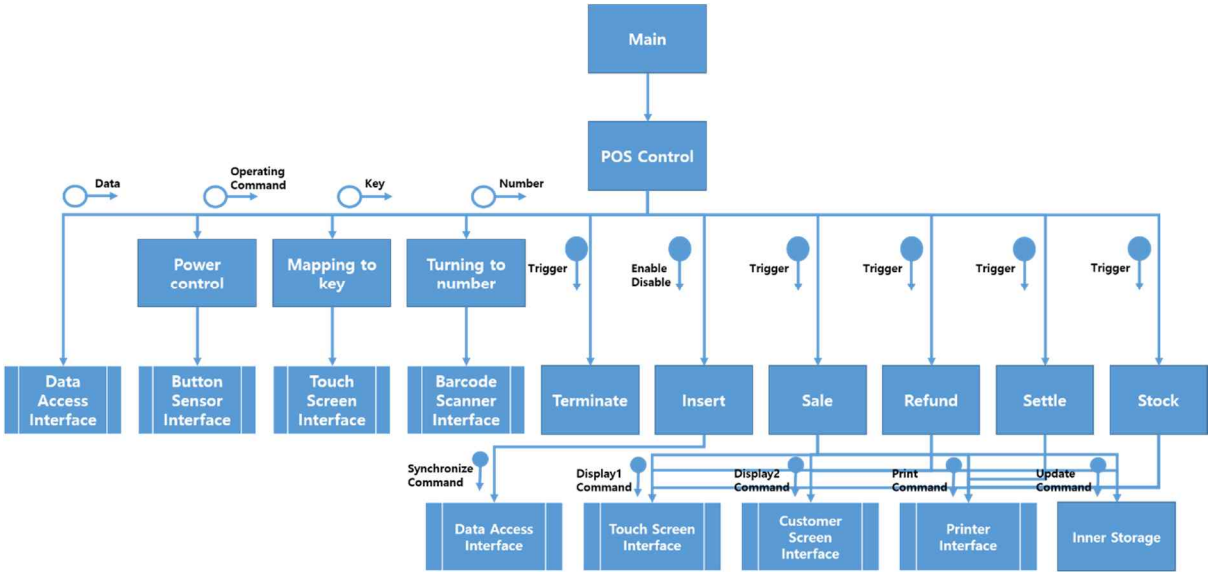
Overall DFD와 Structured Chart를 참고하여 모듈 간 데이터의 흐름을 잘 반영했는지 테스트한다. 또한 이전에 설계한 STD를 통해 상태 전이가 의도한대로 진행되는지 평가한다.



<Overall DFD>



<STD>



<Structured Chart>

3 Features to be tested

시스템의 모듈마다 입출력이 디자인과 일치하는지, 디자인 단계에서 설계한대로 프로세스가 작동하는지, 요구 사항을 토대로 작성된 예외 처리 동작에 중점을 두고 테스트한다.

4 Features not to be tested

SASD 단계에서 명세하지 않기로 정한 구체적인 파일 시스템을 포함한 외부 기기로부터의 입출력과 같은 기능, 외부 시스템에서 동작하는 관련 프로그램(포스기 서버 프로그램) 등은 test 대상에서 제외한다.

5 Approach

Point Of Sale (POS) System의 소스 코드의 컴파일 및 unit test는 Cygwin에서 바로 사용 가능한 CuTest-1.5를 사용한다.

6 Item pass/fail criteria

7 Unit test design specification

7.1 Test design specification identifier

7.2 Features to be tested

7.3 Approach refinements

SRA 문서에 언급된 각각의 프로세스의 상세 내용을 기준으로 test design 및 test case 를 작성한다.

7.4 Test identification

Identifier	Feature
Control Suite	
POS_UTC001	2.1.1의 POS control에 관한 Test Suite이다.
POS_UTC001_001	외부기기로부터 전달 받은 데이터에 대응되는 작업 또는 시스템 내부에서 조건이 충족되었을 때 작업을 키 값으로 할당하여 동작하게 한다.
POS_UTC001_002	Key가 TERMINATE가 되었을 때 자원을 정리하고 정상적으로 포스기를 종료한다.
POS_UTC001_003	Key가 SALE이 되었을 때 입력 받은 상품 번호를 상품 판매 리스트에 추가하고 내부 저장소의 상태를 변경한다.
POS_UTC001_004	Key가 REFUND가 되었을 때 입력 받은 바코드 번호를 식별하고 유효한 영수증일 때 판매 영수증 단위로 상품의 수량을 변경하고 판매 관리에 환불 기록을 추가한다. 바뀐 상태들을 내부 저장소에 반영한다.
POS_UTC001_005	Key가 SETTLE이 되었을 때 내부 저장소의 상태를 연결된 외부 인터페이스로 보내고 재고를 100개로 초기화한다.
POS_UTC001_006	Key가 STOCK이 되었을 때 STOCK하는 시각을 포함하여 내부 저장소의 상태를 연결된 외부 인터페이스로 보낸다.
POS_UTC001_007	Key가 PAYMENT로 되었을 때 상품 코드 입력 받아 상품 판매 리스트에 추가하는 것을 중단하고 상품 판매 영수증을 연결된 외부 기기로 전달한다
Display Suite	
POS_UTC002	2.2의 Touch Screen Interface와 2.3 Customer Screen Interface에 관한 Test Suite이다.
POS_UTC002_001	전달받은 데이터의 속성에 따라(sale unit, refund unit, inventory) 해당 포맷에 맞춰 콘솔에 출력해준다.
POS_UTC002_002	전달받은 데이터의 속성에 따라(sale unit, refund unit) 해당 포맷에 맞춰 콘솔에 출력해준다.
Print Suite	
POS_UTC003	2.4의 Printer Interface에 관한 Test Suite이다.
POS_UTC003	전달받은 데이터의 속성에 따라(sale unit, refund unit, settle unit, stock unit) 해당 포맷에 맞춰 파일을 생성한다.
Server Suite	

POS_UTC004	2.5의 Data Access Interface에 관한 Test Suite이다.
POS_UTC004	하루 단위로 내부 스토리지에 저장된 재고 상태, 판매 관리 데이터를 파일로 생성해준다.

7.5 Feature pass/fail criteria

Point Of Sale (POS) System의 각 모듈(프로세스)은 요구사항을 바탕으로 정의한 입출력 데이터, 프로세스 기능을 모두 만족해야만 한다. 각 모듈의 상세는 SRA의 Process Description 과 State Transition Diagram을 참조한다

8 Unit test case specification

8.1 Test case specification identifier

Identifier	Input	Expected Output
Control Suite		
POS_UTC001	2.1.1의 POS control에 관한 Test Suite이다.	
POS_UTC001_001	SETTLE_SIGN = 0; key = (any key value except for SETTLE);	SETTLE_SIGN → 1 key → SETTLE
POS_UTC001_002	key = TERMINATE;	프로그램 종료
POS_UTC001_003	key = SALE인 상황에 대한 Test Suite	

POS_UTC001_003_C	barcode = 1;
	control = ADD;
	su->items[0].count == 1
		su->items[k].count
		→ su->items[k].count + 1
		(0 ≤ K < 7)

	barcode = 2;
	control = ADD;
	su->items[1].count == 1	

	barcode = 3;
	control = ADD;
	su->items[2].count == 1	

	barcode = 4;
	control = ADD;
	su->items[3].count == 1	

	barcode = 5;
	control = ADD;
	su->items[4].count == 1	

	barcode = 6;	

	control = ADD; su->items[5].count == 1	

	barcode = 7;
	control = ADD;
	su->items[6].count == 1	

POS_UTC001_003_V	barcode = 1;
	control = ADD;
	innerStorage->list[0].volume == 1
		innerStorage->list[k].volume
		→ innerStorage->list[k].volume
		+ 1
		(0 ≤ K < 7)

	barcode = 2;
	control = ADD;
	innerStorage->list[1].volume == 1	

	barcode = 3;
	control = ADD;
	innerStorage->list[2].volume == 1	

	barcode = 4;
	control = ADD;
	innerStorage->list[3].volume == 1	

	barcode = 5;
	control = ADD;
	innerStorage->list[4].volume == 1	

	barcode = 6;
	control = ADD;
	innerStorage->list[5].volume == 1	

	barcode = 7;
	control = ADD;
	innerStorage->list[6].volume == 1	

POS_UTC001_003_1S	barcode = 1;
	control = ADD;
	innerStorage->total_sales == 1000
		innerStorage->total_sales
		→ innerStorage->total_sales + su->items[0].price

POS_UTC001_003_2S	barcode = 2;
	control = ADD;
	innerStorage->total_sales == 1500
		innerStorage->total_sales
		→ innerStorage->total_sales + su->items[1].price

POS_UTC001_003_3S	barcode = 3;
	control = ADD;
	innerStorage->total_sales == 3000
		innerStorage->total_sales
		→ innerStorage->total_sales + su->items[2].price

POS_UTC001_003_4S	barcode = 4;
	control = ADD;
	innerStorage->total_sales == 500
		innerStorage->total_sales
		→ innerStorage->total_sales + su->items[3].price

POS_UTC001_003_5S	barcode = 5;
	control = ADD;
	innerStorage->total_sales == 800
		innerStorage->total_sales → innerStorage->total_sales + su->items[4].price

POS_UTC001_003_6S	barcode = 6;
	control = ADD;
	innerStorage->total_sales == 1200
		innerStorage->total_sales → innerStorage->total_sales + su->items[5].price

POS_UTC001_003_7S	barcode = 7;
	control = ADD;
	innerStorage->total_sales == 2000
		innerStorage->total_sales → innerStorage->total_sales + su->items[6].price

POS_UTC001_003_1A	barcode = 1;
	control = ADD;
	su->amount == 1000
		su->amount → su->amount + su->items[0].price

POS_UTC001_003_2A	barcode = 2;
	control = ADD;
	su->amount == 1500
		su->amount → su->amount + su->items[1].price

POS_UTC001_003_3A	barcode = 3;
	control = ADD;
	su->amount == 3000
		su->amount → su->amount + su->items[2].price

POS_UTC001_003_4A	barcode = 4;
	control = ADD;
	su->amount == 500
		su->amount → su->amount + su->items[3].price

POS_UTC001_003_5A	barcode = 5;
	control = ADD;
	su->amount == 800
		su->amount → su->amount + su->items[4].price

POS_UTC001_003_6A	barcode = 6;
	control = ADD;
	su->amount == 1200
		su->amount → su->amount + su->items[5].price

POS_UTC001_003_7A	barcode = 7;
	control = ADD;
	su->amount == 2000
		su->amount → su->amount + su->items[6].price

POS_UTC001_003_E	barcode = 1;
	control = ADD;
	innerStorage->list[0].volume = 0;
		key → ERROR

	barcode = 2;
	control = ADD;
	innerStorage->list[1].volume = 0;	

	barcode = 3; control = ADD;	

	innerStorage->list[2].volume = 0;	

	barcode = 4;
	control = ADD;
	innerStorage->list[3].volume = 0;	

	barcode = 5;
	control = ADD;
	innerStorage->list[4].volume = 0;	

	barcode = 6;
	control = ADD;
	innerStorage->list[5].volume = 0;	

	barcode = 7;
	control = ADD;
	innerStorage->list[6].volume = 0;	

POS_UTC001_003_2E	barcode = (0 ≤ K < 7)이외의 정수 control = ADD;	key → ERROR
POS_UTC001_004	key = REFUND인 상황에 대한 Test Suite	
POS_UTC001_004_V	receipt = 유효한 영수증 번호 innerStorage.list[0].volume == 100	innerStorage->list[k].volume → innerStorage->list[k].volume+1 (0 ≤ K < 7)
	receipt = 유효한 영수증 번호 innerStorage.list[1].volume == 100	
	receipt = 유효한 영수증 번호 innerStorage.list[2].volume == 100	
	receipt = 유효한 영수증 번호 innerStorage.list[3].volume == 100	
	receipt = 유효한 영수증 번호 innerStorage.list[4].volume == 100	
	receipt = 유효한 영수증 번호 innerStorage.list[5].volume == 100	
	receipt = 유효한 영수증 번호 innerStorage.list[6].volume == 100	
POS_UTC001_004_F	receipt = 유효한 영수증 번호 sm->slist[0].sale->flag == '1'	sm->slist[k].sale->flag → '1' (0 ≤ K < 7)
	receipt = 유효한 영수증 번호 sm->slist[1].sale->flag == '1'	
	receipt = 유효한 영수증 번호 sm->slist[2].sale->flag == '1'	
	receipt = 유효한 영수증 번호 sm->slist[3].sale->flag == '1'	

	receipt = 유효한 영수증 번호 sm->slist[4].sale->flag == '1'	
	receipt = 유효한 영수증 번호 sm->slist[5].sale->flag == '1'	
	receipt = 유효한 영수증 번호 sm->slist[6].sale->flag == '1'	
POS_UTC001_004_1T	receipt = 유효한 영수증 번호 innerStorage.total_refund == 1000	innerStorage.total_refund → innerStorage.total_refund + sm->slist[0].sale->amount;
POS_UTC001_004_2T	receipt = 유효한 영수증 번호 innerStorage.total_refund == 1500	innerStorage.total_refund → innerStorage.total_refund + sm->slist[1].sale->amount;
POS_UTC001_004_3T	receipt = 유효한 영수증 번호 innerStorage.total_refund == 3000	innerStorage.total_refund → innerStorage.total_refund + sm->slist[2].sale->amount;
POS_UTC001_004_4T	receipt = 유효한 영수증 번호 innerStorage.total_refund == 500	innerStorage.total_refund → innerStorage.total_refund + sm->slist[3].sale->amount;
POS_UTC001_004_5T	receipt = 유효한 영수증 번호 innerStorage.total_refund == 800	innerStorage.total_refund → innerStorage.total_refund + sm->slist[4].sale->amount;
POS_UTC001_004_6T	receipt = 유효한 영수증 번호 innerStorage.total_refund == 1200	innerStorage.total_refund → innerStorage.total_refund + sm->slist[5].sale->amount;
POS_UTC001_004_7T	receipt = 유효한 영수증 번호 innerStorage.total_refund == 2000	innerStorage.total_refund → innerStorage.total_refund + sm->slist[6].sale->amount;
POS_UTC001_004_C	receipt = 유효한 영수증 번호 sm->rcount == 1	sm->rcount → sm->rcount +1
POS_UTC001_004_E	sm->slist[k].sale->flag = '1' (0 ≤ K < sm->scount) *이미 환불 완료된 영수증 입력상황을 말함	key → ERROR
POS_UTC001_004_CS	receipt = 유효한 영수증 번호 ru->receipt.code.string이 YYYYMMDDHHmm의 포맷 코드인지 확인	ru->receipt.code.string → "YYYYMMDDHHHmm"
POS_UTC001_005	key = SETTLE인 상황에 대한 Test Suite	

POS_UTC001_005_V	innerStorage.list[0].volume == 100 innerStorage.list[1].volume == 100 innerStorage.list[2].volume == 100 innerStorage.list[3].volume == 100 innerStorage.list[4].volume == 100 innerStorage.list[5].volume == 100 innerStorage.list[6].volume == 100	innerStorage->list[k].volume → 100 (0 ≤ K < 7)
POS_UTC001_005_S	innerStorage.total_sales == 0	innerStorage->total_sales → 0
POS_UTC001_005_R	innerStorage.total_refund == 0	innerStorage.total_refund → 0
POS_UTC001_006	key = STOCK인 상황에 대한 Test Suite	
POS_UTC001_006_T	innerStorage->time == posTime	TRUE
POS_UTC001_007	key = PAYMENT인 상황에 대한 Test Suite	
POS_UTC001_007_1C	su->money - su->amount == 0	su->change → su->money – su->amount su->code.string → YYYYMMDDHHmm 형태의 문자열 삽입 sm->slist에 item이 하나 추가 sm->scount → sm->scount + 1
POS_UTC001_007_2C		
POS_UTC001_007_3C		
POS_UTC001_007_4C		
POS_UTC001_007_5C		
POS_UTC001_007_6C		
POS_UTC001_007_7C		
POS_UTC001_007_S	sm->scount == 1	sm->scount → sm->scount + 1
POS_UTC001_007_P	sm->slist is NOT NULL	TRUE
POS_UTC001_007_CS	su->code.string이 YYYYMMDDHHmm의 포맷 코드인지 확인	su->code.string → “YYYYMMDDHHHmm”
Library Suite		
POS_UTC002	2.2의 Touch Screen Interface와 2.3 Customer Screen Interface에 관한 Test Suite이다.	

POS_UTC002_001_1	kb = NULL; su = NULL;
	ru = NULL; innerStorage = NULL;
	cmd = 0;
		캐셔 스크린의 기본 형태를 화면에 출력

POS_UTC002_001_2	kb = NULL; su = NULL;
	ru = NULL; innerStorage = NULL;
	cmd = 1;
		캐셔가 입력할 수 있는 메뉴 창을 출력

POS_UTC002_001_3	kb = NULL; su = NULL;
	ru = NULL; innerStorage = NULL;
	cmd = 2;
		판매/환불 취소 여부를 묻는 메시지를 출력

POS_UTC002_001_4	key = SALE
	kb = NULL; su = NULL;
	ru = NULL; innerStorage = NULL;
	cmd = 3;
		판매 상황에서 캐셔가 입력할 수 있는 메뉴를 출력
		입력 받은 값에 의해
		*kb → ADD/MINUS

POS_UTC002_001_5	key = SALE
	kb = NULL; su에 출력할 상품 판매 리스트 삽입
	ru = NULL; innerStorage = NULL;
	cmd = 4;
		입력 받은 상품을 실시간으로 판매 리스트에 추가, 감소된 화면을 출력

POS_UTC002_001_6	key = PAYMENT
	kb = NULL; su에 출력할 상품 판매 리스트 삽입
	ru = NULL; innerStorage = NULL;
	cmd = 4;
		su의 member를 포맷에 맞춰 출력

POS_UTC002_001_7	key = REFUND;
	kb = NULL; su = NULL;
	ru에 출력할 환불 상품 리스트 삽입 innerStorage = NULL;
	cmd = 3;
		ru의 member를 포맷에 맞춰 출력

POS_UTC002_001_8	key = STOCK;
	kb = NULL; su = NULL;
	ru = NULL; innerStorage에 현재 재고 상태를 삽입
	cmd = 3;
		innerStorage의 member를 포맷에 맞춰 출력

POS_UTC002_001_9	key = CANCEL;
	kb = NULL; su = NULL;
	ru = NULL; innerStorage = NULL;
	cmd = 3;
		판매/환불 동작이 정상적으로 취소됨을 출력

POS_UTC002_001_10	key = ERROR;
	kb = NULL; su = NULL;
	ru = NULL; innerStorage = NULL;
	cmd = 3;
		에러 상황을 출력

POS_UTC002_001_11	key = TERMINATE;
	kb = NULL; su = NULL;
	ru = NULL; innerStorage = NULL;
	cmd = 3;
		포스기가 자원 정리를 정상적으로 마치고 종료됨을 출력

POS_UTC002_002_1	su = NULL; ru = NULL; cmd = 0;	고객 스크린의 기본 형태를 화면에 출력
POS_UTC002_002_2	key = SALE; ru = NULL; cmd = 1; su = su에 출력할 상품 판매 리스트 삽입	입력 받은 상품을 실시간으로 판매 리스트에 추가, 감소된 화면을 출력

POS_UTC002_002_3	key = PAYMENT; ru = NULL;
	cmd = 1;
	su = su에 출력할 상품 판매 리스트 삽입
		su의 member를 포맷에 맞춰 출력

POS_UTC002_002_4	key = REFUND; su = NULL;
	cmd = 1;
	ru = ru에 출력할 환불 상품 리스트 삽입
		ru의 member를 포맷에 맞춰 출력

Print Suite		
POS_UTC003	2.4의 Printer Interface에 관한 Test Suite이다.	

POS_UTC003_1	key = PAYMENT; ru = NULL;
	su = su에 파일로 생성할 상품 판매 리스트 삽입
	innerStorage = NULL; sm = NULL;
		sale_YYYYMMDDHHmm.txt 형태의 파일 생성
		(YYYYMMDDHHmm → su->code.string)

POS_UTC003_2	key = REFUND; su = NULL;
	ru = ru에 파일로 생성할 환불 상품 리스트 삽입
	innerStorage = NULL; sm = NULL;
		refund_YYYYMMDDHHmm.txt 형태의 파일 생성
		(YYYYMMDDHHmm → ru->receipt.code.string)

POS_UTC003_3	key = STOCK; su = NULL; ru = NULL;
	innerStorage = innerStorage에 현재 재고 상태를 삽입
	sm = NULL;
		stock_YYYYMMDD.txt 형태의 파일 생성
		(YYYYMMDD → posTime)

POS_UTC003_4	key = SETTLE; su = NULL; ru = NULL; innerStorage = innerStorage에 현재 재고 상태를 삽입 sm = sm에 하루 동안 남긴 판매 기록에 대한 데이터를 삽입;	settle_YYYYMMDD.txt 형태의 파일 생성 (YYYYMMDD → posTime)

Server Suite		
POS_UTC004	2.5의 Data Access Interface에 관한 Test Suite이다.	
POS_UTC004	innerStorage = innerStorage에 서버와 동기화할 내부 스토리지 상태를 삽입 sm = sm에 서버에 업데이트 할 판매 기록을 삽입	YYYYMMDD_product.txt 형태의 파일 생성 YYYYMMDD_sale_management.txt형태의 파일 생성 (YYYYMMDD → posTime)

8.2 Test items

상위 표 참조

8.3 Input specifications

상위 표 참조

8.4 Output specifications

상위 표 참조

9 Testing tasks

10 Environmental needs

-gcc 5.4.0버전

-cygwin64

-CuTest-1.5

11 Unit Test deliverables

12 Schedules