Unit Testing Plan

for POS System

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

Project Team

Team 5

Date

2017-11-01

Team Information

201211355 손지웅

201610401 손하영

201611303 조정익

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 시스템으로 구현 하는 것이다.

1.2 Background

POS System이란 판매와 관련한 데이터를 일괄적으로 관리하고, 고객정보를 수집하여 부가 가치를 향상시키는 시스템이다.

Unit Test는 기본적인 실행 가능한 코드에 대한 테스트 코드를 작성하여, 코드 별로 Test를 하는 작업이다. 요구사항을 기준으로 입력과 출력을 정의하여 그 의도대로 올바 른 결과가 나오는지 Test를 한다.

1.3 Scope

POS System이란 판매와 관련한 데이터를 일괄적으로 관리하고, 고객정보를 수집하여 부가 가치를 향상시키는 시스템이다. 본 프로젝트는 전체 POS System 중 POS 단말기 만을 대상으로 구현하는 것으로 규모를 제한한다. 모든 시스템은 SW 만으로 구현하고 HW가 필요한 부분은 SW 모듈을 만들어 가상의 HW를 구현한다.

1.4 Project plan

Post 시스템의 SRA, SDA를 바탕으로 CUnit툴을 이용하여 유닛 테스트를 실행한다.

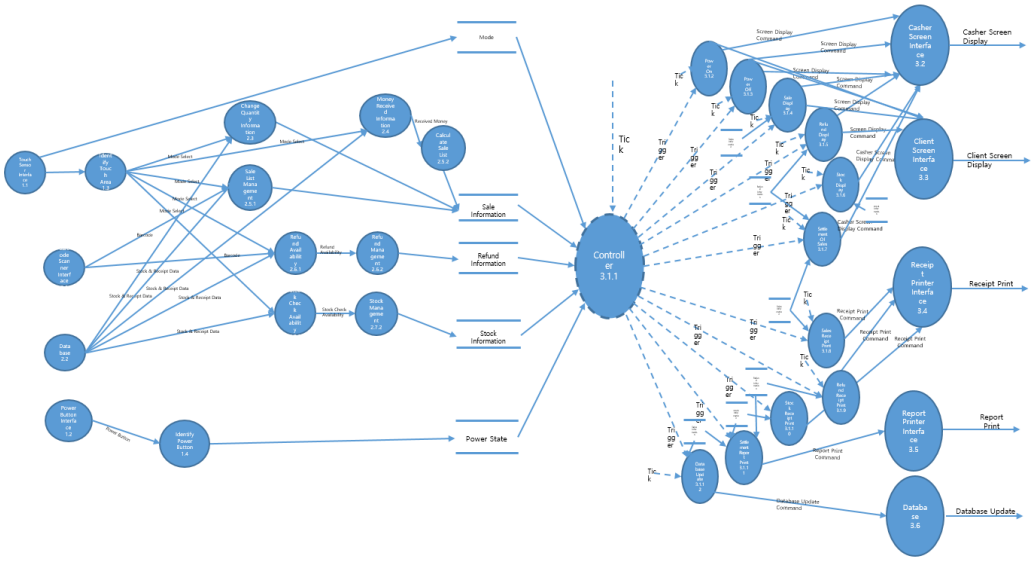
유닛 테스트는 Cygwin환경에서 실행된다.

1.5 Configuration management plan

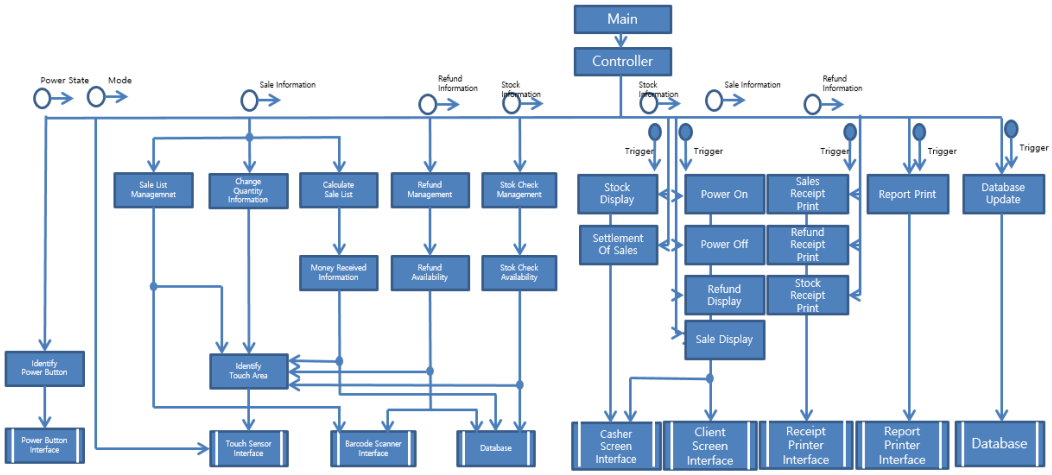
1.6 References

2 Test items

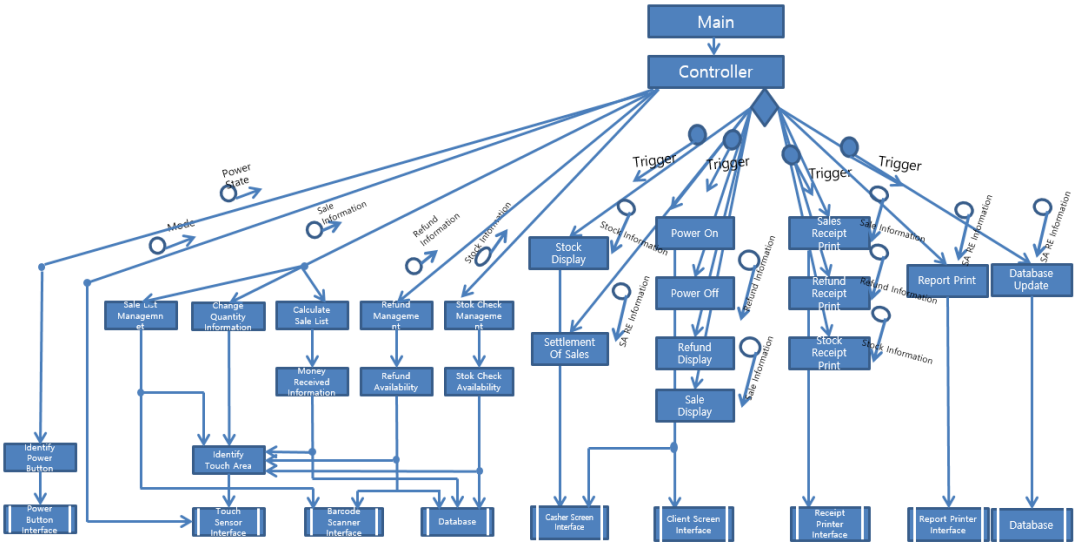
2.1 Team 5가 SASD 기법을 이용하여 개발한 Point Of Sale System의 단위 모듈들이 요구사항을 만족하는지, 입출력이 올바른지, 예외처리가 제대로 되었는지 등을 테스트한다. Test item은 SRA를 바탕으로 작성되었다.



<Figure 1 Overall DFD>



<Figure 2 Basic Structure Chart>



<Figure 3 Advanced Structure Chart>

3 Features to be tested

ID	Name	Description
1.3	Identify Touch Area	Touch Area를 바탕으로 그에 맞는 Mode정보를 보낸다. Input : mode Output : 함수
1.4	Identify Power Button	Power On/Off 상태를 Power State로 보낸다. Input: temp Output: Power state
2.1	Barcode Scnner Interface	바코드 정보를 받는다. Input: 바코드 int[] Output: index
2.3	Change Quantity Information	Mode 가 수량 변경 모드일 때 이루어진다. 수량 변경된 것을 Manage Sale에 알린다. Input: index, stock_info->상품.quantity Output: sale_info->상품.quantity stock_info->상품.quantity sale_info->total_price
2.4	Money Received Information	Mode가 돈 수령 모드일 때 이루어진다. 수령한 돈을 Manage Sale에 알린다. Input: received money, total price, index Output: received money, total price, sended money

2.5.1	Sale List Management	Mode가 Sale모드 때 혹은 돈 수령모드나 수량 변경모드가 이뤄지고 난 이후에 이루어진다. 현재 담겨진 전체 상품 정보들을 Calculate Sale List 로 보낸다. Input: index, sale_info->상품.quantity, stock_info->상품.quantity, sale_info->total_price Ouput: sale_info->상품.quantity, stock_info->상품.quantity, sale_info->total_price
2.5.2	Calculte Sale List	담겨 진 상품 정보들을 Sale List Management로 받고 Received Money를 통해 받은 금액으로 총 계산 된 Sale Information을 만든다. Input: index, sale_info_quantity =0, sale_info_total_price, s_r[0][index], s_r[0][7] Output: sale_display(), database_update(), receipt_list_file(), sale_receipt_print(), s_r[0][index], s_r[0][7]
2.6.1	Refund Availability	Mode가 Refund모드 때 이루어진다. 바코드를 통해 환불 가능 여부를 판단해서 Refund Management 로 넘겨준다. Input: index, list_data, check Output: Refund Availability
2.6.2	Refund Management	Refund Availablity 에 따라 처리 유무를 결정한다. 유효한 영수증이면 바코드 정보를 바탕으로 환불 정보를 생성한다. Input: stock_info, barcode, total_price, refund_stock.상품.quantity Output: stock_info->상품.quantity
2.7.1	Stock Check Availablity	Mode가 Stock 모드 때 이루어진다. 판매나 환불상태인지 확인하고 재고 확인 가능 여부를 넘겨준다. Input: settle_flag Output: stock check Availablity
3.1.4	Sale Display	판매 정보를 보여준다. Input: index, sale_info_recv_money, sale_info_snd_money sale_info_s_arr[index].name, sale_info_s_arr[index].quantity sale_info_s_arr[index].price ,sale_info_total_price Output: 출력
3.1.5	Stock Display	재고 정보를 보여준다. 재고 확인 모드이고 판매 혹은 환불 모드가 아닐 때 수행된다. Input: index, stock_info_s_arr[index].quantity stock_info_s_arr[index].name, stock_info_s_arr[index].price

		Output: 화면 출력
3.1.6	Refund Display	환불 정보를 보여준다 Input: index I, year1, month1, day1, hour1, min1, t_price refund_info[i].name, refund_info[i].price refund_info[i].quantity Output: 출력
3.1.7	Database Update	데이터베이스를 업데이트한다. Input: index I, sale_info, stock_info_s_arr[i].name stock_info_s_arr[i].price, stock_info_s_arr[i].quantity year1, month1, day1, hour1, min1 Output: 출력
3.1.8	Sales Receipt Print	판매 영수증을 출력한다. Input: index I, sale_info_s_arr[i].name sale_info_s_arr[i].price, sale_info_s_arr[i].quantity sale_info_total_price, year1, month1, day1, hour1,min1 Output: 출력
3.1.9	Refund Receipt Print	환불 영수증을 출력한다. Input: index I, refund_info_quantity refund_info_name, refund_info_price year1, month1, day1, hour1, min1 Output: 출력
3.1.10	Stock Receipt Print	재고 확인 영수증을 출력한다. Input: index I, stock_info_s_arr[i].name stock_info_s_arr[i].price, stock_info_s_arr[i].quantity year1, month1, day1, hour1, min1 Output: 출력
3.1.11	Report Print	정산 보고서를 출력한다. Input: index I, day1, month1, year1 sale_info_temp[i].name, sale_info_temp[i].price sale_info_temp[i].quantity refund_info_temp[i].name, refund_info_temp[i].price refund_info_quantity total_sale_price, total_refund_price Output: 출력
3.1.12	Settlement Of Sales	3분(T%180 == 0)마다 정산을 진행한다. 판매 혹은 환불 모드가 아닐 때 수행된다.

4 Features not to be tested

단순 데이터 전달 프로세스, 단순 입력에 대한 출력 값 저장, 센서, 단순 데이터 취합은 Unit Test에서 제외된다.

ID	Name	Description
1.1	Touch Sensor Interface	Touch Sensor로부터 입력 정보를 받는다. 입력 정보(좌표)를 바탕으로 어느 위치가 입력 되었는지 Identify Touch Area로 보낸다.
1.2	Power Button Interface	Power Button으로부터 입력을 받는다. Power Button의 입력 정보를 Identify Power Button으로 보낸다.
2.2	Database	Database로부터 재고 및 영수증 정보를 읽어 온다. 재고 및 영수증 정보를 동시에 보내고 받는 프로세스에서 해당 정보가 자신의 맞는 것인지 확인 하고 처리한다.
2.7.2	Stock Management	Stock Check Availability에 따라 처리 유무를 결정한다. 재고확인이 가능하면 재고 정보를 생성한다
3.1.1	Controller	Data Store의 data들을 바탕으로 출력 및 정산을 제어한다.
3.1.2	Power On	POS기의 전원을 킨다. (Main 상태)
3.1.3	Power Off	POS기의 전원을 끈다.
3.2	Casher Screen Interface	Main Control로부터 Command를 받고 Casher Screen을 출력한다.
3.3	Client Screen Interface	Main Control로부터 Command를 받고 Client Screen을 출력한다.
3.4	Receipt Printer Interface	Main Control로부터 Command를 받고 판매 영수증을 출력한다.
3.5	Report Printer Interface	Main Control로부터 Command를 받고 정산 보고서를 출력한다.
3.6	Database	Main Control로부터 Command를 받고 Database를 업데이트한다.

5 Approach

POS System의 Program Source Code 및 Unit Test를 위한 Test Code는 Cygwin + gcc 환경에서 CUnit으로 이루어지며, Program source code/test code의 변경 및 수정사항은 지속적으로 통합하여 test하며 각 모듈의 Unit test를 마무리 짓는다.

6 Item pass/fail criteria

각 Process/Module은 요구사항을 만족해야 하며, Unit test의 함수들이 특정 input에 대하여 예상된 Output이 나올 경우 pass, 예상값 이외의 오류가 발생할 시 fail이다.

7 Unit test design specification

7.1 Test design specification identifier

POS_Unit번호_시도횟수

7.2 Features to be tested

본 문서 3. Features to be tested 참조

7.3 Approach refinements

각 모듈이 요구사항을 만족하는지를 확인하기 위하여, 요구사항에 정의된 내용에 기반하여 test code를 작성한다. 그 이외의 상황에 대해서는 test code를 작성하지 않는다.

7.4 Test identification

<Table3: test identification>

Identifier	Feature	Valid/ Invalid Value
POS_1.3_000	1.3 Identify Touch Area	Input : mode = 1 (Sale Mode) Output : settle_flag = 0, Sale Mode
POS_1.3_001	1.3 Identify Touch Area	Input : mode = 2 (Quantity Changed Mode) Output : quantity_change() , Quantity Changed Mode
POS_1.3_002	1.3 Identify Touch Area	Input : mode = 3 (Money Received Mode) Output : settle_flag = 1, Money Received Mode
POS_1.3_003	1.3 Identify Touch Area	Input : mode = 4 (Refund Mode) && refund_availability = 0 Output: settle_flag = 1, Refund Mode
POS_1.3_004	1.3 Identify Touch Area	Input : mode = 4(Refund Mode) && (refund_availability > 1 || refund_availability < 1) Output : settle_flag = 1, NULL
POS_1.3_005	1.3 Identify Touch Area	Input : mode = 5 (Stock Check Mode) && (stock_check_availability= 0) Output : settle_falg = 0, Can not check Stock

POS_1.3_006	1.3 Identify Touch Area	Input : mode = 5 && (stock_check_availability>0|| stock_check_availability < 0) Output: settle_flag = 1, Stock Check Mode
POS_1.3_007	1.3 Identify Touch Area	mode < 1 || mode >5 Output : Reselect Mode
POS_1.4_000	1.4 Identify Power Button	Input : temp = 1 Output : 1 (Power On)
POS_1.4_001	1.4 Identity Power Button	Input : temp > 1 || temp < 1 Output : 0 (Power Off)
POS_2.1_000	2.1 Barcode Scanner Interface	Input : char* barcode = 001 Output : 1
POS_2.1_001	2.1 Barcode Scanner Interface	Input : char* barcode = 010 Output : 2
POS_2.1_002	2.1 Barcode Scanner Interface	Input : char* barcode = 011 Output: 3
POS_2.1_003	2.1 Barcode Scanner Interface	Input : char* barcode = 100 Output: 4
POS_2.1_004	2.1 Barcode Scanner Interface	Input: char* barcode = 101 Output: 5
POS_2.1_005	2.1 Barcode Scanner Interface	Input: char* barcode = 110 Output: 6
POS_2.1_006	2.1 Barcode Scanner Interface	Input: char* barcode = 111 Output: 7
POS_2.3_000	2.3 Change Quantity Information	mode = 2일 때 발생 Input: index = 1 && (stock_info->상품1의 수량 == 0) Output: "상품1 이름 is sold out"
POS_2.3_001	2.3 Change Quantity Information	Input: index = 1 && (stock_info->상품1의 수량 >0 || stock_info->상품1의 수량 < 0) Output: (sale_info->상품1의 수량++) && (stock_info->상품1의 수량 --) && total_price(added)
POS_2.3_002	2.3 Change Quantity Information	mode = 2일 때 발생 Input: index = 2 && (stock_info->상품2의 수량 == 0) Output: "상품2 이름 is sold out"
POS_2.3_003	2.3 Change	mode = 2일 때 발생

	Quantity Information	Input: index = 3 && (stock_info->상품3의 수량 == 0) Output: "상품3 이름 is sold out"
POS_2.3_004	2.3 Change Quantity Information	mode = 2일 때 발생 Input: index = 4 && (stock_info->상품4의 수량 == 0) Output: "상품4 이름 is sold out"
POS_2.3_005	2.3 Change Quantity Information	mode = 2일 때 발생 Input: index = 5 && (stock_info->상품5의 수량 == 0) Output: "상품5 이름 is sold out"
POS_2.3_006	2.3 Change Quantity Information	mode = 2일 때 발생 Input: index = 6 && (stock_info->상품6의 수량 == 0) Output: "상품6 이름 is sold out"
POS_2.3_007	2.3 Change Quantity Information	mode = 2일 때 발생 Input: index = 6 && (stock_info->상품6의 수량 == 0) Output: "상품6 이름 is sold out"
POS_2.4_000	2.4 Money Received Information	mode = 3일 때 발생 Input: index = 1 , recv_money, total price Output: snd_money
POS_2.4_001	2.4 Money Received Information	mode = 3일 때 발생 Input: index = 2 , recv_money, total price Output: snd_money
POS_2.4_002	2.4 Money Received Information	mode = 3일 때 발생 Input: index = 3 , recv_money, total price Output: snd_money
POS_2.4_003	2.4 Money Received Information	mode = 3일 때 발생 Input: index = 4 , recv_money, total price Output: snd_money
POS_2.4_004	2.4 Money Received Information	mode = 3일 때 발생 Input: index = 5 , recv_money, total price Output: snd_money
POS_2.4_005	2.4 Money Received Information	mode = 3일 때 발생 Input: index = 6 , recv_money, total price Output: snd_money
POS_2.4_006	2.4 Money	mode = 3일 때 발생

	Received Information	Input: index = 7 , recv_money, total price Output: snd_money
POS_2.5.1_000	2.5.1 Sale List Management	mode = 1이거나 3이나 2 이후에 실행 Input: index = 1 Output: sale_info->상품1의 수량++ sale_info->상품1의 수량- - sale_info->total_price(added)
POS_2.5.1_001	2.5.1 Sale List Management	mode = 1이거나 3이나 2 이후에 실행 Input: index = 2 Output: sale_info->상품2의 수량++ sale_info->상품2의 수량- - sale_info->total_price(added)
POS_2.5.1_002	2.5.1 Sale List Management	mode = 1이거나 3이나 2 이후에 실행 Input: index = 3 Output: sale_info->상품3의 수량++ sale_info->상품3의 수량- - sale_info->total_price(added)
POS_2.5.1_003	2.5.1 Sale List Management	mode = 1이거나 3이나 2 이후에 실행 Input: index = 4 Output: sale_info->상품4의 수량++ sale_info->상품4의 수량- - sale_info->total_price(added)
POS_2.5.1_004	2.5.1 Sale List Management	mode = 1이거나 3이나 2 이후에 실행 Input: index = 5 Output: sale_info->상품5의 수량++ sale_info->상품5의 수량- - sale_info->total_price(added)
POS_2.5.1_005	2.5.1 Sale List Management	mode = 1이거나 3이나 2 이후에 실행 Input: index = 6 Output: sale_info->상품6의 수량++ sale_info->상품6의 수량- - sale_info->total_price(added)
POS_2.5.1_006	2.5.1 Sale List Management	mode = 1이거나 3이나 2 이후에 실행 Input: index = 7 Output: sale_info->상품7의 수량++ sale_info->상품7의 수량- - sale_info->total_price(added)
POS_2.5.1_007	2.5.1 Sale List	mode = 1이거나 3이나 2 이후에 실행

	Management	Input: index < 1|| index > 7 Output: "Barcode Input₩n-> Barcode : "
POS_2.5.2_000	2.5.2 Calculte Sale List	Input: index, sale_info_quantity =0, sale_info_total_price, s_r[0][index], s_r[0][7] Output: sale_display(), database_update(), receipt_list_file(), sale_receipt_print(), s_r[0][index], s_r[0][7]
POS_2.5.2_001	2.5.2 Calculte Sale List	Input: index, sale_info_quantity =1, sale_info_total_price, s_r[0][index], s_r[0][7] Output: sale_display(), database_update(), receipt_list_file(), sale_receipt_print(), s_r[0][index], s_r[0][7]
POS_2.6.1_000	2.6.1 Refund Availablity	mode = 4 일때 실행 Input: check, barcdoe, t_barcode, list_data Output: 0
POS_2.6.1_001	2.6.1 Refund Availablity	mode = 4 일때 실행 Input: check, barcdoe, t_barcode, list_data Output: 0
POS_2.6.1_002	2.6.1 Refund Availablity	mode = 4 일때 실행 Input: check, barcdoe, t_barcode, list_data Output: 1
POS_2.6.1_003	2.6.1 Refund Availablity	mode = 4 일때 실행 Input: check, barcdoe, t_barcode, list_data Output: 2
POS_2.6.1_004	2.6.1 Refund Availablity	mode = 4 일때 실행 Input: check, barcdoe, t_barcode, list_data Output: 3
POS_2.6.1_005	2.6.1 Refund Availablity	mode = 4 일때 실행 Input: check, barcdoe, t_barcode, list_data Output: 4
POS_2.6.2_000	2.6.2 Refund Management	Input: refund_i_quantity : 0 refund_i_name : "snack" stock_k_quantity : 2 stock_k_name: "water" Output: stock_k_quantity: 2 함수 호출
POS_2.6.2_001	2.6.2 Refund Management	Input: refund_i_quantity : 1 refund_i_name : "snack"

		stock_k_quantity: 2 stock_k_name : "snack" Output: stock_k_quantity: 3 함수 호출
POS_2.6.2_002	2.6.2 Refund Management	Input: refund_i_quantity : 1 refund_i_name "snack" stock_k_quantity: 2 stock_k_name "water"' Output: stock_k_quantity: 2 함수 호출
POS_2.7.1_000	2.7.1 Stock Check Availability	mode = 5일 때 실행된다. Input: settle_flag = 0 Output: 0
POS_2.7.1_001	2.7.1	mode = 5일 때 실행된다. Input: settle_flag > 0 || settle_flag < 0 Output: 1
POS_3.1.4_000	3.1.4 Sale Display	Input: index sale_info_recv_money sale_info_snd_money sale_info_s_arr[index].name sale_info_s_arr[index].quantity sale_info_s_arr[index].price sale_info_total_price Output: 출력
POS_3.1.4_001	3.1.4 Sale Display	Input: index sale_info_recv_money sale_info_snd_money sale_info_s_arr[index].name sale_info_s_arr[index].quantity sale_info_s_arr[index].price sale_info_total_price Output: 출력
POS_3.1.5_000	3.1.5 Stock Display	Input: index stock_info_s_arr[index].quantity stock_info_s_arr[index].name stock_info_s_arr[index].price Output: 화면 출력
POS_3.1.5_001	3.1.5 Stock	Input: index

	Display	stock_info_s_arr[index].quantity stock_info_s_arr[index].name stock_info_s_arr[index].price Output: 화면 출력
POS_3.1.6_000	3.1.6 Refund Display	Input: index I, year1, month1, day1, hour1, min1 t_price, refund_info[i].name refund_info[i].price, refund_info[i].quantity Output: 출력
POS_3.1.6_001	3.1.6 Refund Display	Input: index I, year1, month1, day1, hour1, min1 t_price, refund_info[i].name refund_info[i].price, refund_info[i].quantity Output: 출력
POS_3.1.7_000	3.1.7 Database Update	Input: index i. sale_info stock_info_s_arr[i].name, stock_info_s_arr[i].price stock_info_s_arr[i].quantity year1, month1, day1, hour1, min1 Output: 출력
POS_3.1.7_001	3.1.7 Database Update	Input: index i. sale_info stock_info_s_arr[i].name, stock_info_s_arr[i].price stock_info_s_arr[i].quantity year1, month1, day1, hour1, min1 Output: 출력
POS_3.1.7_002	3.1.7 Database Update	Input: index i. sale_info stock_info_s_arr[i].name, stock_info_s_arr[i].price stock_info_s_arr[i].quantity year1, month1, day1, hour1, min1 Output: 출력
POS_3.1.8_000	3.1.8 Sale Receipt Print	Input: index i sale_info_s_arr[i].name sale_info_s_arr[i].price sale_info_s_arr[i].quantity sale_info_total_price year1, month1, day1, hour1, min1 Output: 출력
POS_3.1.8_001	3.1.8 Sale Receipt Print	Input: index i sale_info_s_arr[i].name sale_info_s_arr[i].price sale_info_s_arr[i].quantity

		sale_info_total_price year1, month1, day1, hour1, min1 Output: 출력
POS_3.1.9_000	3.1.9 Refund Receipt Print	Input: index i refund_info_quantity refund_info_name refund_info_price year1, month1, day1, hour1, min1 Output: 출력
POS_3.1.9_001	3.1.9 Refund Receipt Print	Input: index i refund_info_quantity refund_info_name refund_info_price year1, month1, day1, hour1, min1 Output: 출력
POS_3.1.10_000	3.1.10 Stock Receipt Print	Input: index i stock_info_s_arr[i].name stock_info_s_arr[i].price stock_info_s_arr[i].quantity year1, month1, day1, hour1, min1 Output: 출력
POS_3.1.11_000	3.1.11 Settlement Report Print	Input: index i day1, month1, year1 sale_info_temp[i].name sale_info_temp[i].price sale_info_temp[i].quantity refund_info_temp[i].name refund_info_temp[i].price refund_info_quantity total_sale_price total_refund_price Output: 출력
POS_3.1.11_001	3.1.11 Settlement Report Print	Input: index i day1, month1, year1 sale_info_temp[i].name sale_info_temp[i].price sale_info_temp[i].quantity refund_info_temp[i].name

		refund_info_temp[i].price refund_info_quantity total_sale_price total_refund_price Output: 출력
POS_3.1.11_002	3.1.11 Settlement Report Print	Input: index i day1, month1, year1 sale_info_temp[i].name sale_info_temp[i].price sale_info_temp[i].quantity refund_info_temp[i].name refund_info_temp[i].price refund_info_quantity total_sale_price total_refund_price Output: 출력
POS_3.1.11_003	3.1.11 Settlement Report Print	Input: index i day1, month1, year1 sale_info_temp[i].name sale_info_temp[i].price sale_info_temp[i].quantity refund_info_temp[i].name refund_info_temp[i].price refund_info_quantity total_sale_price total_refund_price Output: 출력
POS_3.1.12_000	3.1.12 Settlement Of Sales	Input: settle_flag: 2, get_time_flag: 0 check: 0, year: 2017, month: 11, day: 6 hour: 3, min: 47, start: 1, end: 100 Output: check : 0, get_time_flag: 0 update_flag: 0, expe_index: 0
POS_3.1.12_001	3.1.12 Settlement Of Sales	Input: settle_flag: 1, get_time_flag: 1 check: 1, year: 2017, month: 11 day: 6, hour: 3, min: 47 start: 0,end: 100 Output: check : 0, get_time_flag :0 update_flag 0, expe_index 2

7.5 Feature pass/fail criteria

각각의 나올 수 있는 여러 경우의 수를 입력해보고 그 결과가 만족하는지 여러 번 테스트하여 확인한다.

8 Unit test case specification

8.1 Test case specification identifier

<Table4: test case identification>

Identifier	Input Specification	Output Specification
POS_1.3_000	mode : 1 n: 0	settle_flag: 0 function: Sale Mode
POS_1.3_001	mode : 2 n : 0	settle_flag: 0 function: Quantity Change Mode
POS_1.3_002	mode: 3 n : 0	settle_flag: 1 function: Money Received Mode
POS_1.3_003	mode: 4 n: 1	settle_flag: 0 function: Refund Mode
POS_1.3_004	mode: 4 n: 2	settle_flag: 1 function: NULL
POS_1.3_005	mode: 5 n: 0	settle_flag: 0 function: “Can not check Stock”
POS_1.3_006	mode: 5 n:1	settle_flag = 1 function: Stock Check Mode
POS_1.3_007	mode: 6 n: 0	settle_flag = 0 function: Reselect Mode
POS_1.4_000	temp: 1	1 (Power_on)
POS_1.4_001	temp: 2	0 (Power_off)
POS_2.1_000	barcode: 001	index: 1
POS_2.1_001	barcode: 010	index: 2
POS_2.1_002	barcode: 011	index: 3
POS_2.1_003	barcode: 100	index: 4
POS_2.1_004	barcode: 101	index: 5
POS_2.1_005	barcode: 110	index: 6
POS_2.1_006	barcode: 111	index: 7

POS_2.3_000	index: 1 sale_info_quantity: 0 sale_info_total_price: 0 sale_info_price: 1000 stock_info_quantity: 0	sale_info_quantity: 0 sale_info_total_price: 0 stock_info_quantity: 0 "Sold out"
POS_2.3_001	index: 1 sale_info_quantity: 1 sale_info_total_price: 500 sale_info_price: 1000 stock_info_quantity: 100	sale_info_quantity: 2 sale_info_total_price: 1500 stock_info_quantity: 99 "Sale_display"
POS_2.3_002	index: 2 sale_info_quantity: 3 sale_info_total_price: 1000 sale_info_price: 1500 stock_info_quantity: 4	sale_info_quantity: 4 sale_info_total_price: 2500 stock_info_quantity: 3 "Sale_display"
POS_2.3_003	index: 3 sale_info_quantity: 0 sale_info_total_price: 0 sale_info_price: 3000 stock_info_quantity: 0	sale_info_quantity: 0 sale_info_total_price: 0 stock_info_quantity: 0 "Sold out"
POS_2.3_004	index: 4 sale_info_quantity: 5 sale_info_total_price: 2000 sale_info_price: 500 stock_info_quantity: 50	sale_info_quantity: 6 sale_info_total_price: 2500 stock_info_quantity: 49 "Sale_display"
POS_2.3_005	index: 5 sale_info_quantity: 0 sale_info_total_price: 0 sale_info_price: 800 stock_info_quantity: 0	sale_info_quantity: 0 sale_info_total_price: 0 stock_info_quantity: 0 "Sold out"
POS_2.3_006	index: 6 sale_info_quantity: 10 sale_info_total_price: 5000 sale_info_price: 1200 stock_info_quantity: 3	sale_info_quantity: 11 sale_info_total_price: 6200 stock_info_quantity: 2 "Sale_display"
POS_2.3_007	index: 7 sale_info_quantity: 0 sale_info_total_price: 0	sale_info_quantity: 0 sale_info_total_price: 0 stock_info_quantity: 0

	sale_info_price: 2000 stock_info_quantity: 0	"Sold out"
POS_2.4_000	index: 1 recv_money: 4000 sale_info_total_price: 3000	snd_money: 1000 "Calculate_sale"
POS_2.4_001	index: 2 recv_money: 4000 sale_info_total_price :5000	snd_money: 0 "Received Money deffeceincy"
POS_2.4_002	index: 3 recv_money: 4000 sale_info_total_price: 2500	snd_money: 1500 "Calculate_sale"
POS_2.4_003	index: 4 recv_money: 4000 sale_info_total_price: 6000	snd_money: 0 "Received Money deffeceincy"
POS_2.4_004	index: 5 recv_money: 9000 sale_info_total_price: 5000	snd_money: 4000 "Calculate_sale"
POS_2.4_005	index: 6 recv_money: 2000 sale_info_total_price: 8000	snd_money: 0 "Received Money defficeincy"
POS_2.4_006	index: 7 recv_money: 4500 sale_info_total_price: 3300	snd_money: 1200 "Calculate_sale"
POS_2.5.1_000	index: 1 sale_info_quantity: 1 sale_info_total_price: 500 sale_info_price: 1000 stock_info_quantity: 100	sale_quantity: 2 sale_total_price: 1500 stock_quantity: 99 "Sale_display"
POS_2.5.1_001	index: 2 sale_info_quantity: 3 sale_info_total_price : 1000 sale_info_price: 1500 stock_info_quantity: 4	sale_quantity: 4 sale_total_price: 2500 stock_quantity: 3 "Sale_display"
POS_2.5.1_002	index: 3 sale_info_quantity: 0 sale_info_total_price : 0 sale_info_price: 3000 stock_info_quantity: 12	sale_quantity: 1 sale_total_price: 3000 stock_quantity: 11 "Sale_display"

POS_2.5.1_003	index: 4 sale_info_quantity: 5 sale_info_total_price: 2000 sale_info_price: 500 stock_info_quantity: 50	sale_quantity: 6 sale_total_price: 2500 stock_quantity: 49 "Sale_display"
POS_2.5.1_004	index: 5 sale_info_quantity: 4 sale_info_total_price: 0 sale_info_price: 800 stock_info_quantity: 20	sale_quantity: 5 sale_total_price: 800 stock_quantity: 19 "Sale_display"
POS_2.5.1_005	index: 6 sale_info_quantity: 10 sale_info_total_price: 5000 sale_info_price: 1200 stock_info_quantity: 3	sale_quantity: 11 sale_total_price: 6200 stock_quantity: 2 "Sale_display"
POS_2.5.1_006	index: 7 sale_info_quantity: 8 sale_info_total_price: 0 sale_info_price: 2000 stock_info_quantity: 10	sale_quantity: 9 sale_total_price: 2000 stock_quantity: 9 "Sale_display"
POS_2.5.1_007	index: 0 sale_info_quantity: 9 sale_info_total_price: 0 sale_info_price: 4000 stock_info_quantity: 5	sale_quantity: 9 sale_total_price: 0 stock_quantity: 5 "Invalid Index"
POS_2.5.2_000	index: 1 sale_info_quantity: 0 sale_info_total_price: 1000 s_r_0_i: 3 s_r_0_7_i: 1500	s_r_0: 3 s_r_0_7: 2500 "sale_display" "database_update" "receipt_list_file" "sale_receipt_print"
POS_2.5.2_001	index: 1 sale_info_quantity: 1 sale_info_total_price: 1000 s_r_0_i: 2 s_r_0_7_i: 2000	s_r_0: 3 s_r_0_7: 3000 "sale_display" "database_update" "receipt_list_file" "sale_receipt_print"
POS_2.6.1_000	check: 0	0

	barcode: 001 t_barcode: 001 list_data: 0	
POS_2.6.1_001	check: 1 barcode: 001 t_barcode: 001 list_data: 0	0
POS_2.6.1_002	check: 0 barcode: 001 t_barcode: 001 list_data: 1	1
POS_2.6.1_003	check: 0 barcode: 001 t_barcode: 010 list_data: 1	2
POS_2.6.1_004	check: 1 barcode: 001 t_barcode: 001 list_data: 1	3
POS_2.6.1_005	check: 1 barcode: 001 t_barcode: 010 list_data: 1	4
POS_2.6.2_000	refund_i_quantity : 0 refund_i_name : “snack” stock_k_quantity : 2 stock_k_name: “water”	stock_k_quantity: 2 함수 호출
POS_2.6.2_001	refund_i_quantity : 1 refund_i_name : “snack” stock_k_quantity: 2 stock_k_name : “snack”	stock_k_quantity: 3 함수 호출
POS_2.6.2_002	refund_i_quantity : 1 refund_i_name “snack” stock_k_quantity: 2 stock_k_name ”water”’	stock_k_quantity: 2 함수 호출
POS_2.7.1_000	settle_flag: 0	0
POS_2.7.1_001	settle_flag: 1	1
POS_3.1.4_000	index : 1	화면 출력

	sale_info_recv_money: 1000 sale_info_snd_money: 2000 sale_info_s_arr[index].name: "snack" sale_info_s_arr[index].quantity: 0 sale_info_s_arr[index].price: 1000 sale_info_total_price: 5000	
POS_3.1.4_001	index : 1 sale_info_recv_money: 1000 sale_info_snd_money: 2000 sale_info_s_arr[index].name: "snack" sale_info_s_arr[index].quantity: 1 sale_info_s_arr[index].price : 1000 sale_info_total_price: 5000	화면 출력
POS_3.1.5_000	index: 1 stock_info_s_arr[index].quantity: 0 stock_info_s_arr[index].name: "snack" stock_info_s_arr[index].price: 1000	화면 출력
POS_3.1.5_001	index: 1 stock_info_s_arr[index].quantity: 1 stock_info_s_arr[index].name:"snack" stock_info_s_arr[index].price: 1000	화면 출력
POS_3.1.6_000	index I : 1 year1: 2017 month1: 11 day1: 6 hour1: 12 min1: 7 t_price: 4500 refund_info[i].name: "snack" refund_info[i].price: 3000 refund_info[i].quantity: 0	화면 출력
POS_3.1.6_001	index I : 1 year1 : 2017 month1: 11 day1: 6 hour1: 12 min1: 7	화면 출력

	t_price: 4500 refund_info[i].name: "snack" refund_info[i].price: 3000 refund_info[i].quantity: 1	
POS_3.1.7_000	index I :1 sale_info: 0 sale_info_s_arr[i].quantity: 2 sale_info_s_arr[i].name: "snack" sale_info_s_arr[i].price: 1000 sale_info_s_arr[i].total_price: 3500 stock_info_s_arr[i].name: "snack" stock_info_s_arr[i].price: 1000 stock_info_s_arr[i].quantity: 0 year1: 2017 month1: 11 day1: 6 hour1: 12 min1: 7	화면 출력
POS_3.1.7_001	index I : 1 sale_info: 1 sale_info_s_arr[i].quantity: 2 sale_info_s_arr[i].name: "snack" sale_info_s_arr[i].price: 1000 sale_info_s_arr[i].total_price : 3500 stock_info_s_arr[i].name: "snack" stock_info_s_arr[i].price: 1000 stock_info_s_arr[i].quantity: 0 year1: 2017 month1: 11 day1: 6 hour1: 12 min1: 7	화면 출력
POS_3.1.7_002	index I =1 sale_info: 1 sale_info_s_arr[i].quantity: 2 sale_info_s_arr[i].name: "snack" sale_info_s_arr[i].price: 1000 sale_info_s_arr[i].total_price: 3500 stock_info_s_arr[i].name: "snack"	화면 출력

	stock_info_s_arr[i].price: 1000 stock_info_s_arr[i].quantity: 1 year1: 2017 month1: 11 day1: 6 hour1: 12 min1: 7	
POS_3.1.8_000	index I :1 sale_info_s_arr[i].name: "snack" sale_info_s_arr[i].price: 1000 sale_info_s_arr[i].quantity: 0 sale_info_total_price: 5000 year1 : 2017 month1: 11 day1: 6 hour1: 12 min1: 15	화면 출력
POS_3.1.8_001	index I : 1 sale_info_s_arr[i].name: "snack" sale_info_s_arr[i].price: 1000 sale_info_s_arr[i].quantity: 1 sale_info_total_price: 5000 year1: 2017 month1: 11 day1: 6 hour1: 12 min1: 15	화면 출력
POS_3.1.9_000	index I :1 refund_info_quantity: 0 refund_info_name: "snack" refund_info_price: 1000 year1: 2017 month1: 11 day1: 6 hour1: 12 min1: 17	화면 출력
POS_3.1.9_001	index I :1 refund_info_quantity: 1	화면 출력

	refund_info_name: "snack" refund_info_price: 1000 year1: 2017 month1: 11 day1: 6 hour1: 12 min1:17	
POS_3.1.10_000	index I : 1 stock_info_s_arr[i].name: "snack" stock_info_s_arr[i].price: 1000 stock_info_s_arr[i].quantity: 4 year1 : 2017 month1: 11 day1: 6 hour1: 12 min1: 19	화면 출력
POS_3.1.11_000	index I : 1 day1 : 6 month1: 11 year1: 2017 sale_info_temp[i].name: "snack" sale_info_temp[i].price: 1000 sale_info_temp[i].quantity: 0 refund_info_temp[i].name: "snack" refund_info_temp[i].price: 1000 refund_info_quantity: 0 total_sale_price: 2000 total_refund_price: 1000	화면 출력
POS_3.1.11_001	index I : 1 day1: 6 month1: 11 year1: 2017 sale_info_temp[i].name: "snack" sale_info_temp[i].price: 1000 sale_info_temp[i].quantity: 0 refund_info_temp[i].name: "snack" refund_info_temp[i].price: 1000 refund_info_quantity: 1 total_sale_price: 2000	화면 출력

	total_refund_price: 1000	
POS_3.1.11_002	index I : 1 day1: 6 month1: 11 year1: 2017 sale_info_temp[i].name: "snack" sale_info_temp[i].price: 1000 sale_info_temp[i].quantity: 1 refund_info_temp[i].name: "snack" refund_info_temp[i].price: 1000 refund_info_quantity: 0 total_sale_price: 2000 total_refund_price: 1000	화면 출력
POS_3.1.11_003	index I = 1 day1: 6 month1 : 11 year1: 2017 sale_info_temp[i].name: "snack" sale_info_temp[i].price: 1000 sale_info_temp[i].quantity: 1 refund_info_temp[i].name: "snack" refund_info_temp[i].price: 1000 refund_info_quantity: 1 total_sale_price:2000 total_refund_price:1000	화면 출력
POS_3.1.12_000	settle_flag: 2 get_time_flag: 0 check: 0 year: 2017 month: 11 day: 6 hour: 3 min: 47 start: 1 end: 100	check : 0 get_time_flag: 0 update_flag: 0 return int: 0
POS_3.1.12_001	settle_flag: 1 get_time_flag: 1 check: 1	check = 0; get_time_flag :0 update_flag 0

	year: 2017 month: 11 day: 6 hour: 3 min: 47 start: 0 end: 100	return int 2

8.2 Test items

<Table3: Test design identification> 참조

8.3 Input specifications

<Table4: Test case identification> input 참조

8.4 Output specifications

<Table4: Test case identification> output 참조

9 Testing tasks

1) SA/SD 작성

2) Process를 모듈 단위로 나누기

3) 모듈별 input/output 정하기

4) 소스 코드 작성

5) Unit test 실시

6) Test report

10 Environmental needs

POS System의 Unit Test를 위한 환경적 요구사항은 다음과 같다.

(1) Hardware & Platform

gcc compiler/linker

(2) Continuous Testing & Integrated Platform Environment

Cygwin

(3) Framework

CUnit

11 Unit Test deliverables

1) Unit test plan

2) Unit test design specification

3) Unit test case specification

4) Unit test report

12 Schedules

9 Testing task 참조.