# Unit Test Report

# for Point of Sale System

- **Test Cases Specification**
- **Test Summary Report**

**Project Team**

**Team 4**

Latest update on:

**2017-11-04**

---

**Team Information**
**산업공학과 201211178 민경훈**
**산업공학과 201211187 배승현**
**컴퓨터공학과 201311283 송형선**
**컴퓨터공학과 201611299 정희승**

# Table of Contents

# 1 Introduction

## 1.1 Objectives

본 문서는 2017학년도 2학기 소프트웨어 공학 개론 수업의 Team 4이 개발한 Point of Sale System을 Unit Testing을 수행한 결과에 대한 Report 문서이다. Test 요소들에 대한 Test Case와 Test 수행 결과에 대한 내용을 담고 있다.

## 1.2 References

[2017SE_B][TP#1]SRA_Team4_rev3

[2017SE_B][TP#2]SDS_Team4_rev2

[2017SE_B][TP#3]UTP_Team4

# 2 Unit test case specification

## 2.1 Test case specification identifier

| **Identifier** | **Feature** | **Pass / Fail Criteria** |
|---|---|---|
| POS.UTC.111 | Sale Process 가 흐르는지 확인 | SALE = TRUE/False |
| POS.UTC.111.00 | 데이터를 전달하는 모듈이므로 테스트 하지 않음 | |
| POS.UTC.121 | Refund Process 가 흐르는지 확인 | REFUND = TRUE/False |
| POS.UTC.121.00 | 데이터를 전달하는 모듈이므로 테스트 하지 않음 | |
| POS.UTC.131 | Inventory Check Process 가 흐르는지 확인 | CHECK = TRUE/False |
| POS.UTC.131.00 | 데이터를 전달하는 모듈이므로 테스트 하지 않음 | |
| POS.UTC.141 | Power Process 가 흐르는지 확인 | POWER_OFF = True/False |
| POS.UTC.141.00 | 데이터를 전달하는 모듈이므로 테스트 하지 않음 | |
| POS.UTC.151 | SalesList Merging 되는지 확인 | Saleslist_server_Merging = 0 or True |
| POS.UTC.151.00 | input:<br>char saleFileTestServer[] = "test_sale_management.txt"; | Saleslist_server_Merging = 0 |

| | | |
|---|---|---|
| | expected : 0 | |
| POS.UTC.151.01 | input:<br>char saleFileTestServer[] = "test_sale_management.txt";<br>char receipt_code[20] = "2017.11.04.09.36";<br>SoldData * sold_data = (SoldData *)malloc(sizeof(SoldData) * total_item);<br>strcpy((sold_data)->item, "과자");<br>(sold_data)->money = 1000;<br>(sold_data)->quantity = 2;<br>((sold_data)->pay) = 2000;<br>expected : 1 | Saleslist_server_Merging = 1 |
| POS.UTC.151.02 | input:<br>char saleFileTestServer[] = "test_sale_management.txt";<br>char receipt_code[20] = "2017.11.04.09.36";<br>SoldData * sold_data = (SoldData *)malloc(sizeof(SoldData) * total_item);<br>strcpy((sold_data)->item, "과자");<br>(sold_data)->money = 1000;<br>(sold_data)->quantity = 2;<br>((sold_data)->pay) = 2000;<br>strcpy((sold_data+1)->item, "물");<br>(sold_data+1)->money = 500;<br>(sold_data+1)->quantity = 2;<br>((sold_data+1)->pay) = 1000;<br><br>expected : 2 | Saleslist_server_Merging = 2 |
| POS.UTC.152 | Inventory Merging 되는지 확인 | Inventory_server_Merging = 0 or 1 |
| POS.UTC.152.00 | input:<br>char productTestServer[] = "test_product.txt"; | Inventory_server_Merging = 0 |

| | | |
|---|---|---|
| | ItemData* itd = (ItemData*)malloc(sizeof(ItemData));<br>expected : 0 | |
| POS.UTC.152.01 | input:<br>char productTestServer[] = "test_product.txt";<br>ItemData* itd = (ItemData*)malloc(sizeof(ItemData) * 8);<br>expected : 1 | Inventory_server_Merging |
| POS.UTC.211 | Main Controller 가 제대로 수행되는지 확인 | 통합적인 부분(전달 프로세스)이므로 Test 제외 |
| POS.UTC.211.00 | 데이터를 전달하는 모듈이므로 테스트 하지 않음 | |
| POS.UTC.212 | Update_Cashier_Screen 가 수행 되어지는지 확인 | Update_Cashier_Screen =0 or 1 |
| POS.UTC.212.00 | input :<br>char ca[1000];<br>sprintf(ca,"");<br>expected : 0 | Update_Cashier_Screen =0 |
| POS.UTC.212.01 | input :<br>char ca[1000];<br>sprintf(ca,"₩n%-15s %-15s %-15s₩n", "상품", "수량", "가격");<br>expected : 1 | Update_Cashier_Screen = 1 |
| POS.UTC.213 | Update_Customer_Screen 가 수행 되어지는지 확인 | Update_Customer_Screen=0 or 1 |
| POS.UTC.213.00 | input:<br>char ct[1000];<br>sprintf(ct,"");<br>expected : 0 | Update_Customer_Screen = 0 |
| POS.UTC.213.01 | input:<br>char ct[1000];<br>sprintf(ct,"₩n 판매 | Update_Customer_Screen = 1 |

| | | |
|---|---|---|
| | 날짜 : %d 년 %d 월 %d 일 %d 시 %d 분₩n", 2017, 1, 1, 1, 1);<br>expected: 1 | |
| POS.UTC.214 | Print_Receipt 가 수행 되어지는지 확인 | Print_Receipt=0 or 1 |
| POS.UTC.214.00 | input:<br>char testReceipt[] = "stock_test.txt";<br>char str[1000];<br>sprintf(str, “");<br>expected: 0 | Print_Receipt = 0 |
| POS.UTC.214.01 | input:<br>char testReceipt[] = "stock_test.txt";<br>char str[1000];<br>sprintf(str, "%-15s %-15s %-15s₩n", "상품", "단가", "수량");<br>expected: 1 | Print_Receipt = 1 |
| POS.UTC.215 | Run 이 작동하는지 확인. 잘못된 데이터시 Run =0 | Run = 0 or 1 |
| POS.UTC.215.00 | Main 과 마찬가지로 데이터를 전달하는 모듈이므로 테스트 하지 않음 | |
| POS.UTC.216 | Reset_Saleslist 가 수행 되어지는지 확인 | Reset_Saleslist = 0 or 1 |
| POS.UTC.216.00 | input:<br>char saleFileTestServer[] = "test_sale_management.txt";<br>char receipt_code[20] = "2017.11.04.01.28";<br>SoldData * sold_data = (SoldData *)malloc(sizeof(SoldData) * (1+1));<br>strcpy((sold_data)->item, "물");<br>(sold_data)->money = 500;<br>(sold_data)->quantity = 2;<br>((sold_data)->pay) = 1000;<br>len = 0<br>expected: 0 | Reset_Saleslist = 0 |

| | | |
|---|---|---|
| POS.UTC.216.01 | input:<br>char saleFileTestServer[] = "test_sale_management.txt";<br>char receipt_code[20] = "2017.11.04.01.28";<br>SoldData * sold_data = (SoldData *)malloc(sizeof(SoldData) * (1+1));<br>strcpy((sold_data)->item, "물");<br>(sold_data)->money = 500;<br>(sold_data)->quantity = 2;<br>((sold_data)->pay) = 1000;<br>len = 1<br>expected: 1 | Reset_Saleslist = 1 |
| POS.UTC.217 | Update_Sales_List 가 수행 되어지는지 확인 | Update_Sales_List = 0 or 1 |
| POS.UTC.217.00 | input:<br>char productTestServer[] = "test_product.txt";<br>len = 0;<br>expected: 0 | Update_Sales_List = 0 |
| POS.UTC.217.01 | input:<br>char productTestServer[] = "test_product.txt";<br>len = 1;<br>expected: 1 | Update_Sales_List = 1 |
| POS.UTC.218 | Update_Inventory 가 수행 되어지는지 확인 | Update_Inventory = 0 or 1 |
| POS.UTC.218.00 | input:<br>char productTestServer[] = "test_product.txt";<br>len = 0;<br>expected: 0 | Update_Inventory = 0 |
| POS.UTC.218.01 | input:<br>char productTestServer[] = "test_product.txt"; | Update_Inventory = 1 |

| | | |
|---|---|---|
| | len = 1;<br>expected: 1 | |
| POS.UTC.219 | Reset_Inventory 가 수행 되어지는지 확인 | Reset_Inventory = 0 or 1 |
| POS.UTC.219.00 | input:<br>char productTestServer[] = "";<br>expected: 0 | Reset_Inventory = 0 |
| POS.UTC.219.01 | input:<br>char productTestServer[] = "reset_product.txt";<br>expected: 1 | Reset_Inventory = 1 |

2.2 Test items

**Table 1 Test Design Identification** 참조

2.3 Input specifications

**Table 1 Test Design Identification** 참조

2.4 Output specifications

**Table 1 Test Design Identification** 참조

3 Environmental needs

4 Unit test summary report

4.1 Test summary report identifier

| **Identifier** | **Feature** | **Pass / Fail Criteria** | **Test Result** |
|---|---|---|---|
| POS.UTC.111 | Sale Process 가 흐르는지 확인 | SALE = TRUE/False | |
| POS.UTC.111.00 | 데이터를 전달하는 모듈이므로 테스트 하지 않음 | | |
| POS.UTC.12 | Refund Process 가 흐르는지 확인 | REFUND = | |

| | | | |
|---|---|---|---|
| 1 | | TRUE/False | |
| POS.UTC.121.00 | 데이터를 전달하는 모듈이므로 테스트 하지 않음 | | |
| POS.UTC.131 | Inventory Check Process 가 흐르는지 확인 | CHECK = TRUE/False | |
| POS.UTC.131.00 | 데이터를 전달하는 모듈이므로 테스트 하지 않음 | | |
| POS.UTC.141 | Power Process 가 흐르는지 확인 | POWER_OFF = True/False | |
| POS.UTC.141.00 | 데이터를 전달하는 모듈이므로 테스트 하지 않음 | | |
| POS.UTC.151 | SalesList Merging 되는지 확인 | Saleslist_server_Merging = 0 or True | |
| POS.UTC.151.00 | input:<br>char saleFileTestServer[] = "test_sale_management.txt";<br>expected : 0 | Saleslist_server_Merging = 0 | Passed |
| POS.UTC.151.01 | input:<br>char saleFileTestServer[] = "test_sale_management.txt";<br>char receipt_code[20] = "2017.11.04.09.36";<br>SoldData * sold_data = (SoldData *)malloc(sizeof(SoldData) * total_item);<br>strcpy((sold_data)->item, "과자");<br>(sold_data)->money = 1000;<br>(sold_data)->quantity = 2;<br>((sold_data)->pay) = 2000;<br>expected : 1 | Saleslist_server_Merging = 1 | Passed |
| POS.UTC.151.02 | input:<br>char saleFileTestServer[] = "test_sale_management.txt";<br>char receipt_code[20] = "2017.11.04.09.36"; | Saleslist_server_Merging = 2 | Passed |

| | | | |
|---|---|---|---|
| | SoldData * sold_data = (SoldData *)malloc(sizeof(SoldData) * total_item);<br>strcpy((sold_data)->item, "과자");<br>(sold_data)->money = 1000;<br>(sold_data)->quantity = 2;<br>((sold_data)->pay) = 2000;<br>strcpy((sold_data+1)->item, "물");<br>(sold_data+1)->money = 500;<br>(sold_data+1)->quantity = 2;<br>((sold_data+1)->pay) = 1000;<br><br>expected : 2 | | |
| POS.UTC.152 | Inventory Merging 되는지 확인 | Inventory_server_Merging = 0 or 1 | |
| POS.UTC.152.00 | input:<br>char productTestServer[] = "test_product.txt";<br>ItemData* itd = (ItemData*)malloc(sizeof(ItemData));<br>expected : 0 | Inventory_server_Merging = 0 | Passed |
| POS.UTC.152.01 | input:<br>char productTestServer[] = "test_product.txt";<br>ItemData* itd = (ItemData*)malloc(sizeof(ItemData) * 8);<br>expected : 1 | Inventory_server_Merging | Passed |
| POS.UTC.211 | Main Controller 가 제대로 수행되는지 확인 | 통합적인 부분(전달 프로세스)이므로 Test 제외 | |
| POS.UTC.211.00 | 데이터를 전달하는 모듈이므로 테스트 하지 않음 | | |
| POS.UTC.21 | Update_Cashier_Screen 가 수행 | Update_Cashier_Scre | |

| | | | |
|---|---|---|---|
| 2 | 되어지는지 확인 | en=0 or 1 | |
| POS.UTC.212.00 | input :<br>char ca[1000];<br>sprintf(ca,"");<br>expected : 0 | Update_Cashier_Screen =0 | Passed |
| POS.UTC.212.01 | input :<br>char ca[1000];<br>sprintf(ca,"₩n%-15s %-15s %-15s₩n", "상품", "수량", "가격");<br>expected : 1 | Update_Cashier_Screen = 1 | Passed |
| POS.UTC.213 | Update_Customer_Screen 가 수행 되어지는지 확인 | Update_Customer_Screen=0 or 1 | |
| POS.UTC.213.00 | input:<br>char ct[1000];<br>sprintf(ct,"");<br>expected : 0 | Update_Customer_Screen = 0 | Passed |
| POS.UTC.213.01 | input:<br>char ct[1000];<br>sprintf(ct,"₩n 판매<br>날짜 : %d 년 %d 월 %d 일 %d 시 %d 분₩n", 2017, 1, 1, 1, 1);<br>expected: 1 | Update_Customer_Screen = 1 | Passed |
| POS.UTC.214 | Print_Receipt 가 수행 되어지는지 확인 | Print_Receipt=0 or 1 | |
| POS.UTC.214.00 | input:<br>char testReceipt[] =<br>"stock_test.txt";<br>char str[1000];<br>sprintf(str, "");<br>expected: 0 | Print_Receipt = 0 | Passed |
| POS.UTC.214.01 | input:<br>char testReceipt[] =<br>"stock_test.txt";<br>char str[1000];<br>sprintf(str, "%-15s %-15s %- | Print_Receipt = 1 | Passed |

| | | | |
|---|---|---|---|
| | 15s₩n", "상품", "단가", "수량");<br>expected: 1 | | |
| POS.UTC.215 | Run 이 작동하는지 확인. 잘못된 데이터시 Run =0 | Run = 0 or 1 | |
| POS.UTC.215.00 | Main 과 마찬가지로 데이터를 전달하는 모듈이므로 테스트 하지 않음 | | |
| POS.UTC.216 | Reset_Saleslist 가 수행 되어지는지 확인 | Reset_Saleslist = 0 or 1 | |
| POS.UTC.216.00 | input:<br>char saleFileTestServer[] = "test_sale_management.txt";<br>char receipt_code[20] = "2017.11.04.01.28";<br>SoldData * sold_data = (SoldData *)malloc(sizeof(SoldData) * (1+1));<br>strcpy((sold_data)->item, "물");<br>(sold_data)->money = 500;<br>(sold_data)->quantity = 2;<br>((sold_data)->pay) = 1000;<br>len = 0<br>expected: 0 | Reset_Saleslist = 0 | Passed |
| POS.UTC.216.01 | input:<br>char saleFileTestServer[] = "test_sale_management.txt";<br>char receipt_code[20] = "2017.11.04.01.28";<br>SoldData * sold_data = (SoldData *)malloc(sizeof(SoldData) * (1+1));<br>strcpy((sold_data)->item, "물");<br>(sold_data)->money = 500;<br>(sold_data)->quantity = 2;<br>((sold_data)->pay) = 1000;<br>len = 1<br>expected: 1 | Reset_Saleslist = 1 | Passed |
| POS.UTC.21 | Update_Sales_List 가 수행 | Update_Sales_List = | |

| | | | |
|---|---|---|---|
| 7 | 되어지는지 확인 | 0 or 1 | |
| POS.UTC.217.00 | input:<br>char productTestServer[] = "test_product.txt";<br>len = 0;<br>expected: 0 | Update_Sales_List = 0 | Passed |
| POS.UTC.217.01 | input:<br>char productTestServer[] = "test_product.txt";<br>len = 1;<br>expected: 1 | Update_Sales_List = 1 | Passed |
| POS.UTC.218 | Update_Inventory 가 수행 되어지는지 확인 | Update_Inventory = 0 or 1 | |
| POS.UTC.218.00 | input:<br>char productTestServer[] = "test_product.txt";<br>len = 0;<br>expected: 0 | Update_Inventory = 0 | Passed |
| POS.UTC.218.01 | input:<br>char productTestServer[] = "test_product.txt";<br>len = 1;<br>expected: 1 | Update_Inventory = 1 | Passed |
| POS.UTC.219 | Reset_Inventory 가 수행 되어지는지 확인 | Reset_Inventory = 0 or 1 | |
| POS.UTC.219.00 | input:<br>char productTestServer[] = "";<br>expected: 0 | Reset_Inventory = 0 | Passed |
| POS.UTC.219.01 | input:<br>char productTestServer[] = "reset_product.txt";<br>expected: 1 | Reset_Inventory = 1 | Passed |

## 4.2 Evaluation

Total Test Case : 19

Passed : 19

Failed : 0