

Unit Testing Plan

for Public Transportation System

- Test Plan
- Test Design Specification
- Test Cases Specification

Project Team

Team 3

Date

2014-11-21

Team Information

201311269 김제현

201311275 박상희

201311276 박형민

201311287 엄현식

Table of Contents

- 1 Introduction 5
 - 1.1 Objectives..... 5
 - 1.2 Background 5
 - 1.3 Scope..... 5
 - 1.4 Project plan 5
 - 1.5 Configuration management plan..... 5
 - 1.6 References..... 5
- 2 Test items 6
 - 2.1 Public Transportation System 6
 - 2.2 Recharger System..... 7
 - 2.3 Fee Calculation System..... 8
- 3 Features to be tested 8
 - 3.1 Public Transportation System 8
 - 3.2 Recharger System..... 9
 - 3.3 Fee Calculation System..... 9
- 4 Features not to be tested 10
 - 4.1 Public Transportation System 10
 - 4.2 Recharger System..... 11
 - 4.3 Fee Calculation System..... 11
- 5 Approach..... 11
- 6 Item pass/fail criteria 11

- 7 Unit test design specification..... 12
 - 7.1 Test design specification identifier 12
 - 7.1.1 Public Transportation System..... 12
 - 7.1.2 Recharger System 12
 - 7.1.3 Fee Calculation..... 12
 - 7.2 Features to be tested 12
 - 7.2.1 Process in SRA..... 12
 - 7.2.1.1 Public Transportation System..... 12
 - 7.2.1.2 Recharger System 12
 - 7.2.1.3 Fee Calculation System 12
 - 7.3 Approach refinements..... 12
 - 7.3.1 Brute force Testing 12
 - 7.4 Test identification 12
 - 7.4.1 Public Transportation System..... 12
 - 7.4.2 Recharger System 17
 - 7.4.3 Fee Calculation System 18
 - 7.5 Feature pass/fail criteria 21
- 8 Unit test case specification..... 22
 - 8.1 Test case specification identifier..... 22
 - 8.1.1 Public Transportation System..... 22
 - 8.1.2 Recharger System 26
 - 8.1.3 Fee Calculation System 28
 - 8.2 Test items 31

- 8.2.1 Public Transportation System..... 31
- 8.2.2 Recharger System 31
- 8.2.3 Fee Calculation System 31
- 8.3 Input specifications..... 31
 - 8.3.1 Public Transportation System..... 31
 - 8.3.2 Recharger System 31
 - 8.3.3 Fee Calculation System 31
- 8.4 Output specifications..... 32
 - 8.4.1 Public Transportation System..... 32
 - 8.4.2 Recharger System 32
 - 8.4.3 Fee Calculation System 32
- 9 Testing tasks 32
- 10 Environmental needs 32
- 11 Unit Test deliverables..... 32
- 12 Schedules 32

1 Introduction

1.1 Objectives

이 문서는 T3의 Public Transportation System, Recharger System, Fee Calculation System의 unit test를 수행하기 위해 작성된 계획 문서이며, 본 system이 제대로 작동하는지를 살펴보기 위해 필요한 요소들을 정리해 놓은 문서이다. Test를 수행하기 위해 필요한 활동 및 자원을 정의하고, test approach 및 techniques를 정의한다. 또한 test를 위한 환경적인 요구사항 및 test 도구들을 정의한다.

1.2 Background

Public Transportation System(이하 PTS)은 대중교통 카드 시스템으로, 사용자가 카드 태그를 통해 요금 결제, 환승 및 카드 정보를 출력하는 시스템이다.

Recharger System(이하 GS)은 충전 시스템으로, 사용자가 카드를 태그하고 돈을 넣어 카드를 충전 하는 시스템이다.

Fee Calculation System(이하 FCS)은 정산 시스템으로, 3분마다 수익을 계산하고 수익을 출력하고 전송하는 시스템이다.

Unit test는 시스템을 구성하는 최소 단위 모듈들을 대상으로 하는 test이며, 시스템의 성능을 좌우하는 요소들이 요구사항을 만족하는지를 확인할 수 있는 기본적인 Test approach이다.

1.3 Scope

이 계획 문서는 PTS, GS, FCS의 unit test를 수행하기 위한 모든 것을 포함한다. PTS, GS, FCS의 unit test를 수행하기 위한 자원과 절차, test approach와 technique과 필요로 하는 환경 및 도구 등을 정의한다. PTS, GS, FCS의 unit test는 시스템을 구성하는 최소 단위의 모듈들을 대상으로 하며, 구현된 모듈이 요구사항을 만족하는지를 test 한다.

1.4 Project plan

1.5 Configuration management plan

1.6 References

DS-2014SE-PTS-SRS-1.0

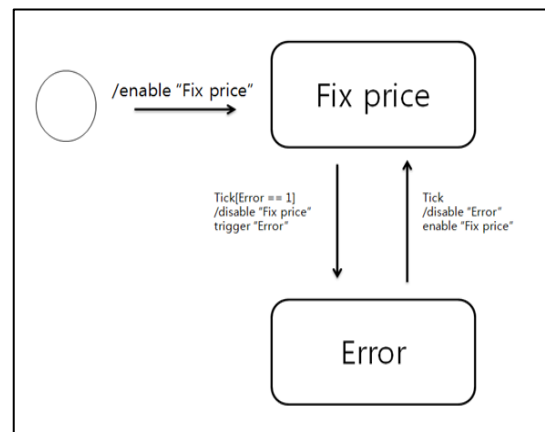
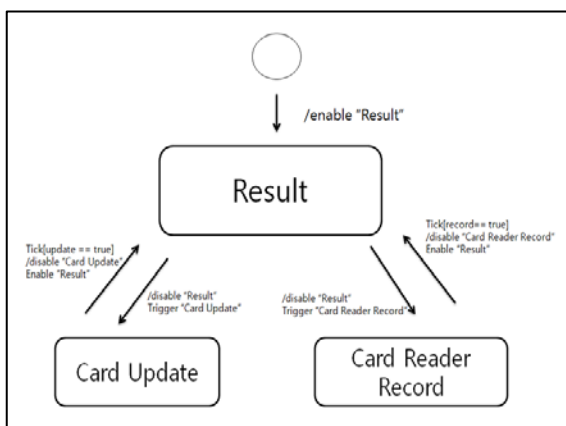
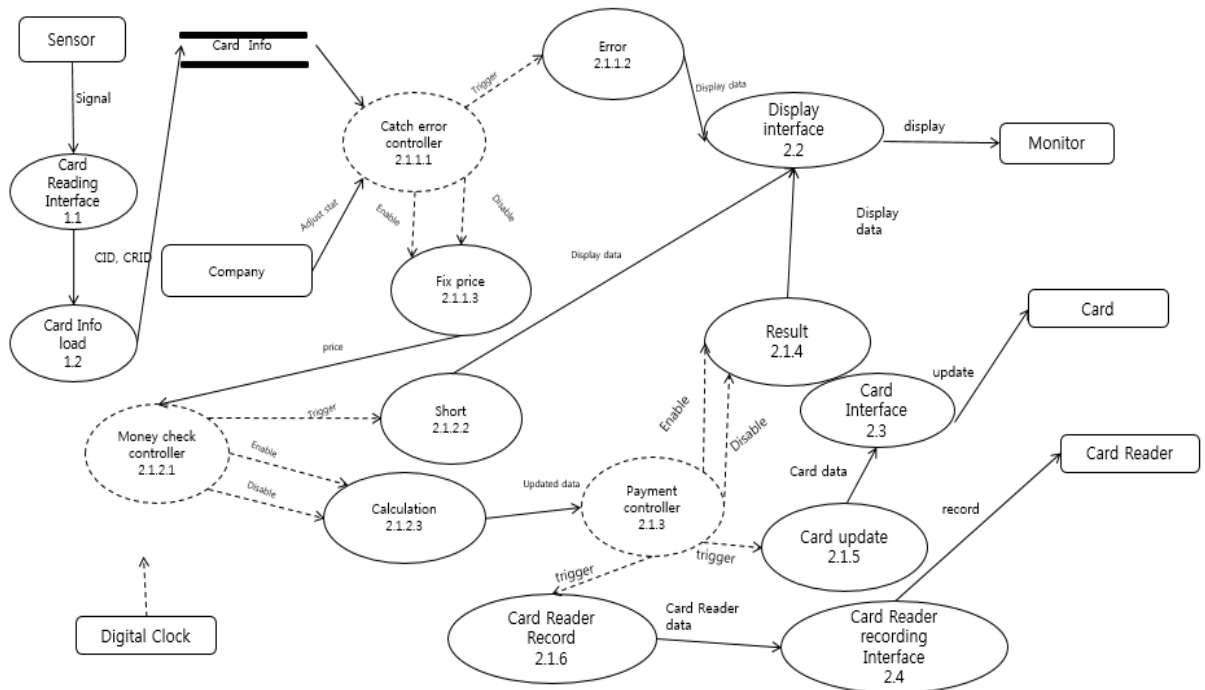
T3-2014-PTS-SRA-1.3

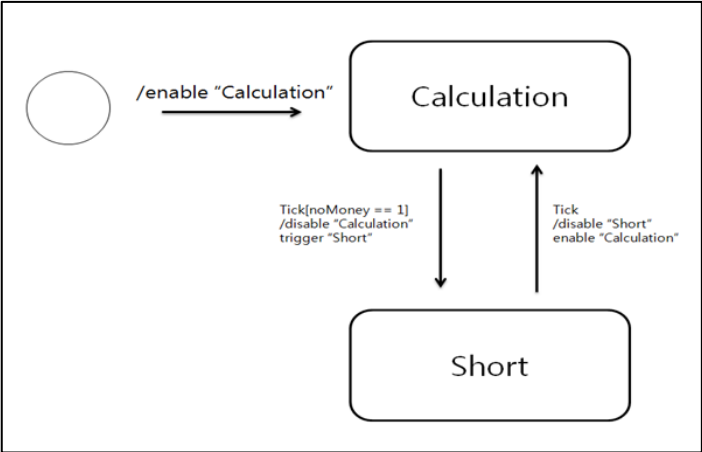
T3-2014-PTS-SDS-2.0

2 Test items

2.1 Public Transportation System

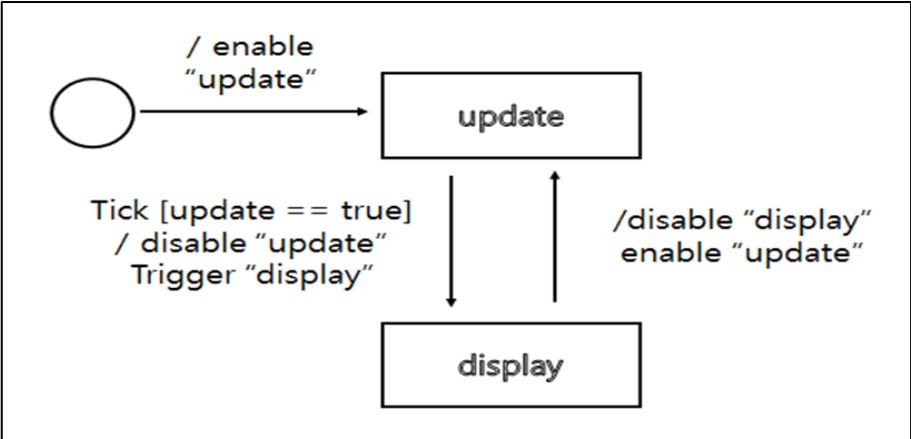
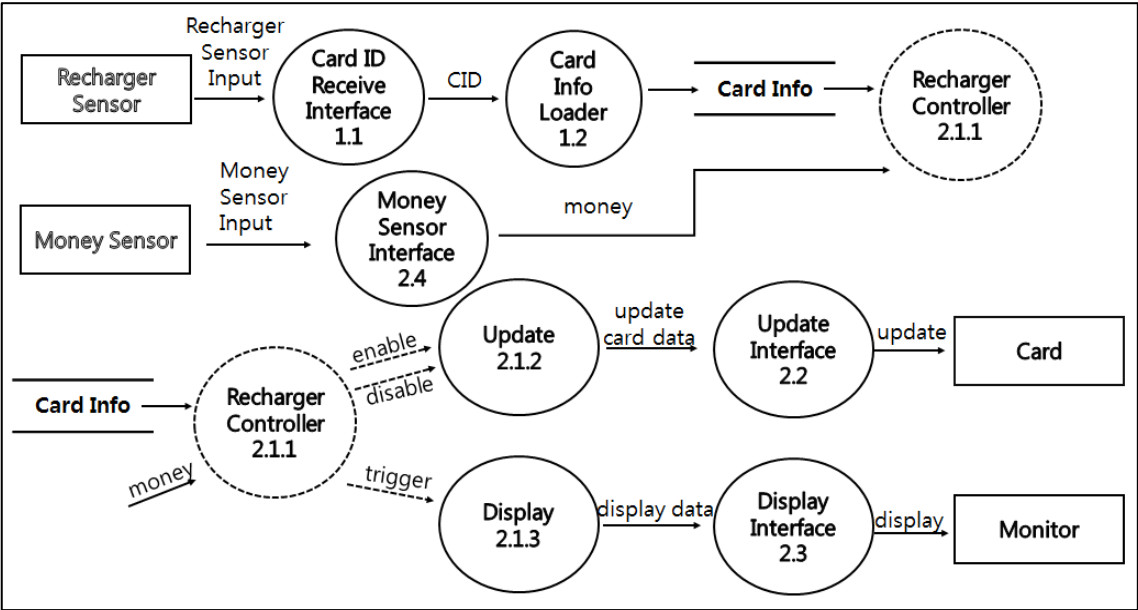
PTS를 구성하는 최소 단위의 모듈들이 unit test의 대상이 된다. 각 모듈의 요구사항을 만족 하는지를 test하며 test item은 다음 자료들로부터 작성되었다.





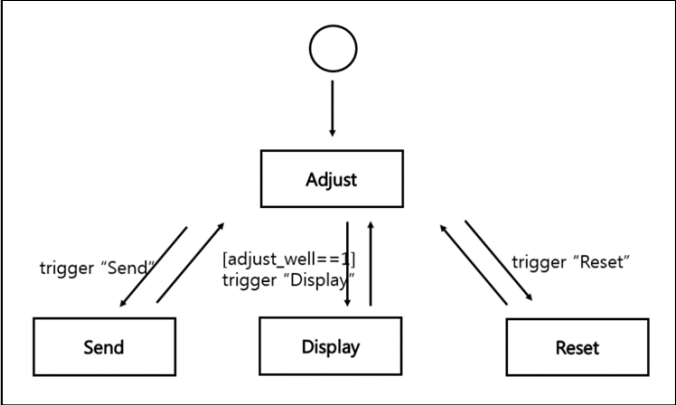
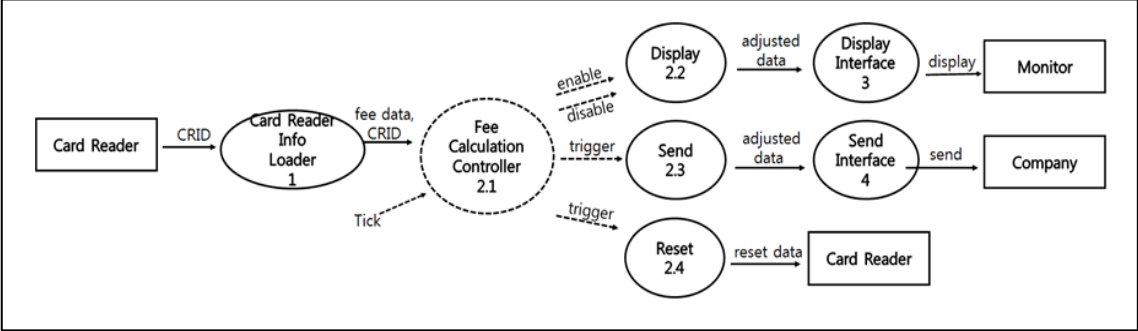
2.2 Recharger System

RS를 구성하는 최소 단위의 모듈들이 unit test의 대상이 된다. 각 모듈의 요구사항을 만족 하는지를 test하며 test item은 다음 자료들로부터 작성되었다.



2.3 Fee Calculation System

Fee Calculation System을 구성하는 최소 단위의 모듈들이 Unit Test의 대상이 된다. 각 모듈의 요구 사항을 만족하는지를 Test하며, Test Item은 다음 자료들로부터 작성되었다



3 Features to be tested

- 1) Process in SRA : 각 프로세스가 가지고 있는 요구사항을 만족하는지를 Test한다.
- 2) Modules in SDS : 각 모듈이 가지고 있는 데이터 인터페이스를 Test한다.

3.1 Public Transportation System

<Table 1.1 테스트할 Process(DFD) 리스트>

ID	Name	Description
1.2	Card Info load	CID를 indexing하여 해당 Card의 잔액, 탑승 단말기, 승/하차, 태그 시간, 환승상태 정보를 불러온다.
2.1.1.1	Catch Error Controller	정산 여부에 관한 정보를 받아온 뒤, 정산이 이루어지지 않았을 경우 경고메시지를 출력하도록 한다. 정산이 이루어졌을 경우 Card Info를 받아와서 정상적인 카드 Tag가 이루어졌는지 판별하고, 아닐 경우 경고메시지를 출력하도록 한다. 정상적일

		경우 가격을 측정하도록 한다.
2.1.1.2	Error	정상적이지 않은 Tag나 정산이 이루어지지 않았을 경우 경고 메시지를 보낸다.
2.1.1.3	Fix price	Card Info를 받아온 뒤, 해당 조건에 맞는 가격을 측정한다.
2.1.2.1	Money Check Controller	측정된 가격을 받아온 뒤, Card Info의 잔액과 비교한 뒤 부족하면 Short, 충분하면 Calculation을 실행한다.
2.1.2.2	Short	가격이 부족할 경우 경고메시지를 출력한다.
2.1.2.3	Calculation	잔액이 충분할 경우 잔액-결제금액 을 한 뒤, 승/하차 상태 및 환승 상태, 역 단말기 정보, 잔액을 갱신한 뒤 Updated data로 보내준다.
2.1.3	Payment Controller	Card 결제가 이루어진 후, 처리된 정보와 현재 시각을 받아와서 결과값을 출력하고, Card에 갱신, 역 단말기에 기록한다.
2.1.4	Result	Card 결제가 이루어지고 난 뒤의 결제 금액, 잔액, 현재 시각(Display data)을 보내준다.
2.1.5	Card update	Card 결제가 이루어지고 난 뒤 바뀐 Card Info를 갱신한다.
2.1.6	Card Reader Record	Card 결제가 이루어지고 난 뒤, 결제 금액을 Card Reader에 기록한다.

3.2 Recharger System

<Table 2.1 테스트할 Process(DFD) 리스트>

ID	Name	Description
1.2	Card Info Loader	CID를 사용해서 Card Info를 불러와 Recharger Control 에 전달한다.
2.1.1	Recharger Controller	입력받은 Card Info, money Data를 종합하여 충전 계산을 한 뒤, 적절한 update와 display를 실행 해준다.
2.1.2	Update	충전된 정보로 교통카드를 갱신한다.
2.1.3	Display	교통카드에 충전된 정보를 Monitor에 보여준다.

3.3 Fee Calculation System

<Table 3.1 테스트할 Process(DFD) 리스트>

ID	Name	Description
1	Card Reader Info Loader	CRID(단말기 고유 ID)를 받아서 해당하는 단말기파일을 열

		어서 데이터들을 불러와서 데이터 형식에 맞게 각각 저장한다.
2.1	Fee Calculation Controller	CRID를 통해서 얻은 데이터와 Tick을 받아서 정산을 한 후 상태를 통해서 적절한 프로세스에 명령을 전달한다.
2.2	Display	Fee Calculation Controller가 정산한 결과를 출력할 때 호출하는 프로세스. 명령을 받으면 adjust_well==0인지 비교(정산이 잘 되었는지)를 하여 정산이 잘 되었으면 adjusted data (fee_bus, fee_metro, time_now)를 출력한다.
2.3	Send	Fee Calculation Controller가 정산한 결과를 지하철회사와 버스회사에 보낼 때 호출하는 프로세스. Trigger 명령을 받으면 adjust_well==0인지 비교를 하여 정산이 잘 되었으면 버스회사와 지하철 회사에 정산이 잘되었다는 값(c_well)과 정산 결과값(fee_bus or fee_metro)을 보낸다.
2.4	Reset	Fee Calculation Controller가 정산한 결과를 초기화 시킬 때 호출하는 프로세스. Trigger 명령을 받으면 adjust_well==0인지 비교를 하여 정산이 잘 되었으면 모든 단말기 파일을 초기화 시킨다.

4 Features not to be tested

Process in SRA : 외부 장치 드라이버, 단순 데이터 전달 프로세스 등은 Test에서 제외한다.

4.1 Public Transportation System

<Table 1.2 테스트하지 않을 Process(DFD) 리스트>

ID	Name	Description
1.1	Card Reading Interface	카드를 Tag했을 때 전달되는 신호를, 컴퓨터가 해석할 수 있는 값을 바꾸어 보내준다.
2.2	Display interface	결정된 금액 혹은 경고 메시지, 현재 시각에 관한 정보를 Display data 를 통해 받아오고, 정리된 정보를 보내준다.
2.3	Card interface	결제 후, 갱신해야 할 Card data를 받아온 뒤, Card에 넘겨준다.
2.4	Card Reader Recording interface	결제 후, 결제 금액을 받아온 뒤, 각 Card Reader에 기록하기 위한 정보를 보내준다.

4.2 Recharger System

<Table 2.2 테스트하지 않을 Process(DFD) 리스트>

ID	Name	Description
1.1	Card ID Receive Interface	Recharger Sensor로부터 받은 아날로그 신호를 디지털 신호로 변환한다
2.2	Update Interface	update card data를 받아서 Card 정보를 Update 시키는 정보를 보내준다
2.3	Display interface	display data를 받아서 Monitor에 출력할 display정보를 보내준다.
2.4	Money Sensor Interface	Money Sensor로부터 받은 아날로그 신호를 디지털 신호로 변환한다.

4.3 Fee Calculation System

<Table 3.2 테스트하지 않을 Process(DFD) 리스트>

ID	Name	Description
3	Display Interface	adjusted data(fee_bus, fee_metro, time_now)를 받아서 정산 결과를 출력해주는 Display 장치에 보내준다.
4	Send Interface	adjusted data(fee_bus, fee_metro, c_well)를 받아서 정산 결과를 버스 회사와 지하철 회사에 보내주는 Send 장치에 보내준다.

5 Approach

Public Transportation System, Recharger System, Fee Calculation System의 Program source code 및 Unit Test를 위한 Test code는 Visual studio 2010 환경에서 이루어지며, Program source code/ Test code의 변경 및 수정 사항은 지속적으로 통합되고 Test된다.

1)Brute force Testing : 각 모듈의 요구사항을 만족하는지를 확인할 수 있는 Test Case를 작성한다. 그 이외의 예외사항에 대해서는 Test하지 않는다.

6 Item pass/fail criteria

Functional Test pass/ Fail criteria : 각 모듈은 요구사항을 모두 만족하여야 한다.

7 Unit test design specification

7.1 Test design specification identifier

7.1.1 Public Transportation System

PTS.UTC.0000.000

7.1.2 Recharger System

RS.UTC.000.000

7.1.3 Fee Calculation

FCS.UTC.000.000

7.2 Features to be tested

7.2.1 Process in SRA

7.2.1.1 Public Transportation System

<Table 1.1 테스트할 Process(DFD) 리스트> 참조

7.2.1.2 Recharger System

<Table 2.1 테스트할 Process(DFD) 리스트> 참조

7.2.1.3 Fee Calculation System

<Table 3.1 테스트할 Process(DFD) 리스트> 참조

7.3 Approach refinements

7.3.1 Brute force Testing

PTS, RS, FCS의 각 모듈이 요구사항을 만족하는지를 확인하기 위하여, 요구사항에 정의된 내용에 기반하여 Test Case를 작성한다. 단, 그 이외의 상황에 대해서는 Test Case를 작성하지 않는다.

7.4 Test identification

7.4.1 Public Transportation System

<Table 1.3 Test Design Identification>

Identifier	Feature	Valid/ Invalid Value
PTS.UTC_1200_000	1.2 Card Info Loader	File!=NULL 상태에서 유효한 card_info의 입력이 들어온다.
PTS.UTC_1200_001	1.2 Card Info Loader	File!=NULL 유효하지 않은 card_info의 입력이 들어온다.
PTS.UTC_1200_002	1.2 Card Info Loader	File==NULL인 입력이 들어온다.
PTS.UTC_2111_000	2.1.1.1 Catch Error Controller	card_info의 값과 정수인 price의 값과 stat==Normal인 입력이 들어온다
PTS.UTC_2111_001	2.1.1.1 Catch Error Controller	card_info의 값과 정수인 price의 값과 stat!=Normal 인 입력이 들어온다
PTS.UTC_2112_000	2.1.1.2 Error	interval_sec <=0 인 입력이 들어온다.
PTS.UTC_2112_001	2.1.1.2 Error	state==1,interval_sec<15,getout==0이고 CRID의 값이 card_info의 CRID의 값과 같은 입력이 들어온다.
PTS.UTC_2112_002	2.1.1.2 Error	state==1,interval_sec>=15,getout==0이고 CRID의 값이 card_info의 CRID의 값과 같은 입력이 들어온다.
PTS.UTC_2112_003	2.1.1.2 Error	state==1,interval_sec<15,getout!=0이고 CRID의 값이 card_info의 CRID의 값과 같은 입력이 들어온다.
PTS.UTC_2112_004	2.1.1.2 Error	state==1,interval_sec<15,getout==0이고 CRID의 값이 card_info의 CRID의 값과 다른 입력이 들어온다.
PTS.UTC_2112_005	2.1.1.2 Error	state==1,interval_sec<15,getout!=0이고 CRID의 값이 card_info의 CRID의 값과 다른 입력이 들어온다.
PTS.UTC_2112_006	2.1.1.2 Error	state==1,interval_sec>=15,getout!=0이고 CRID의 값이 card_info의 CRID의 값과 같은 입력이 들어온다.
PTS.UTC_2112_007	2.1.1.2 Error	state==1,interval_sec>=15,getout==0이고 CRID의 값이 card_info의 CRID의 값과 다른 입력이 들어온다.
PTS.UTC_2112_008	2.1.1.2 Error	state==1,interval_sec>=15,getout!=0이고 CRID의 값이 card_info의 CRID의 값과 다른 입력이 들어온다.
PTS.UTC_2112_009	2.1.1.2 Error	state==0,interval_sec<15,getout==0이고 CRID의 값이 card_info의 CRID의 값과 같은 입력이 들어

		온다.
PTS.UTC_2112_010	2.1.1.2 Error	state==0,interval_sec>=15,getout==0이고 CRID의 값이 card_info의 CRID의 값과 같은 입력이 들어온다.
PTS.UTC_2112_011	2.1.1.2 Error	state==0,interval_sec<15,getout!=0이고 CRID의 값이 card_info의 CRID의 값과 같은 입력이 들어온다.
PTS.UTC_2112_012	2.1.1.2 Error	state==0,interval_sec<15,getout==0이고 CRID의 값이 card_info의 CRID의 값과 다른 입력이 들어온다.
PTS.UTC_2112_013	2.1.1.2 Error	state==0,interval_sec<15,getout!=0이고 CRID의 값이 card_info의 CRID의 값과 다른 입력이 들어온다.
PTS.UTC_2112_014	2.1.1.2 Error	state==0,interval_sec>=15,getout!=0이고 CRID의 값이 card_info의 CRID의 값과 같은 입력이 들어온다.
PTS.UTC_2112_015	2.1.1.2 Error	state==0,interval_sec>=15,getout==0이고 CRID의 값이 card_info의 CRID의 값과 다른 입력이 들어온다.
PTS.UTC_2112_016	2.1.1.2 Error	state==0,interval_sec>=15,getout!=0이고 CRID의 값이 card_info의 CRID의 값과 다른 입력이 들어온다.
PTS.UTC_2113_000	2.1.1.3 FixPrice	interval_sec <= 0인 입력을 들어온다.
PTS.UTC_2113_001	2.1.1.3 FixPrice	CRID < 10인 입력이 들어온다.
PTS.UTC_2113_002	2.1.1.3 FixPrice	CRID >= 10인 입력이 들어온다.
PTS.UTC_2113_003	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 1 && transfer == 1 && tp == 1이다.
PTS.UTC_2113_004	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 1 && transfer == 1 && tp == 0이다.
PTS.UTC_2113_005	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 1 && transfer == 1 && tp != 1 && tp != 0이다.
PTS.UTC_2113_006	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 1 && transfer == 0 && tp == 1이다.
PTS.UTC_2113_007	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 1 && transfer == 0 && tp == 0이다.
PTS.UTC_2113_008	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 1 && transfer == 0 && tp != 1 && tp != 0이다.
PTS.UTC_2113_009	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 1 &&

		transfer != 0 && transfer != 1이다.
PTS.UTC_2113_010	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 0 && transfer == 0 && interval_sec <= 15 && tp == card_info->tp이다.
PTS.UTC_2113_011	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 0 && transfer == 0 && interval_sec <= 15 && tp == 1 && card_info->tp == 0이다.
PTS.UTC_2113_012	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 0 && transfer == 0 && interval_sec <= 15 && tp == 0 && card_info->tp == 1이다.
PTS.UTC_2113_013	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 0 && transfer == 0 && interval_sec <= 15 && tp != 0 && tp != 1 && card_info->tp != 0 && card_info->tp != 1이다.
PTS.UTC_2113_014	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 0 && transfer == 0 && interval_sec > 15
PTS.UTC_2113_015	2.1.1.3 FixPrice	(CRID%10) == 1 상태에서 state == 0 && transfer != 0
PTS.UTC_2113_016	2.1.1.3 FixPrice	(CRID%10) == 0 상태에서 state == 1 && transfer == 1 && tp == 1 이면서 interval_station 가 1 또는 4이다.
PTS.UTC_2113_017	2.1.1.3 FixPrice	(CRID%10) == 0 상태에서 state == 1 && transfer == 1 && tp == 1 이면서 interval_station 가 2 또는 3이다.
PTS.UTC_2113_018	2.1.1.3 FixPrice	(CRID%10) == 0 상태에서 state == 1 && transfer == 1 && tp == 1 이면서 interval_station 가 1~4가 아니다.
PTS.UTC_2113_019	2.1.1.3 FixPrice	(CRID%10) == 0 상태에서 state == 1 && transfer == 1 && tp == 0 이면서 (interval_sec/30) > 5이다.
PTS.UTC_2113_020	2.1.1.3 FixPrice	(CRID%10) == 0 상태에서 state == 1 && transfer == 1 && tp == 0 이면서 (interval_sec/30) <= 5이다.
PTS.UTC_2113_021	2.1.1.3 FixPrice	(CRID%10) == 0 상태에서 state == 1 && transfer == 1 && tp != 0 && tp != 1이다.
PTS.UTC_2113_022	2.1.1.3 FixPrice	PTS.UTC_2113_021 (CRID%10) == 0 상태에서 state == 1 && transfer == 0 && tp == 1 이면서 interval_station가 1 또는 4이다.

PTS.UTC_2113_023	2.1.1.3 FixPrice	(CRID%10) == 0 상태에서 state == 1 && transfer == 0 && tp == 1 이면서 interval_station 가 2 또는 3이다.
PTS.UTC_2113_024	2.1.1.3 FixPrice	(CRID%10) == 0 상태에서 state == 1 && transfer == 0 && tp == 1 이면서 interval_station 가 1~4가 아니다.
PTS.UTC_2113_025	2.1.1.3 FixPrice	(CRID%10) == 0 상태에서 state == 1 && transfer == 0 && tp == 0 이다.
PTS.UTC_2113_026	2.1.1.3 FixPrice	(CRID%10) == 0 상태에서 state == 1 && transfer == 0 && tp != 1 && tp != 0 이다.
PTS.UTC_2113_027	2.1.1.3 FixPrice	(CRID%10) == 0 상태에서 state != 1 이다.
PTS.UTC_2113_028	2.1.1.3 FixPrice	(CRID%10) != 0 && (CRID%10) != 1 이다.
PTS.UTC_2121_000	2.1.2.1 Money Check Controller	stat == Normal이다.
PTS.UTC_2121_001	2.1.2.1 Money Check Controller	Short호출 후 stat == Normal이다.
PTS.UTC_2122_000	2.1.2.2 Short	(cash-price) < 0 이다.
PTS.UTC_2122_000	2.1.2.3 Calculation	CRID < 10 값이 들어온다.
PTS.UTC_2123_001	2.1.2.3 Calculation	CRID >= 10 값이 들어온다.
PTS.UTC_2123_002	2.1.2.3 Calculation	카드에 저장된 transfer값이 0이다.
PTS.UTC_2123_003	2.1.2.3 Calculation	(CRID%10)==0이다.
PTS.UTC_2123_004	2.1.2.3 Calculation	(CRID%10)!=0이다.
PTS.UTC_2130_000	2.1.3 Payment Controller	stat == Normal 값이 들어온다.
PTS.UTC_2140_000	2.1.4 Result	stat == Normal && transfer == 1인 값이 들어온다.
PTS.UTC_2140_001	2.1.4 Result	Normal && transfer != 1인 값이 들어온다.
PTS.UTC_2140_002	2.1.4 Result	stat == HopInProcessing 값이 들어온다.
PTS.UTC_2140_003	2.1.4 Result	stat == GetoffProcessing 값이 들어온다.
PTS.UTC_2140_004	2.1.4 Result	stat == EShort 값이 들어온다.
PTS.UTC_2140_005	2.1.4 Result	stat == NotAdjust 값이 들어온다.
PTS.UTC_2140_006	2.1.4 Result	stat == InvalidInput 값이 들어온다.
PTS.UTC_2150_000	2.1.5 Card update	file != NULL && newFile != NULL && feof(file) == false 상태에서 CID == fileCID 값이 들어온다.
PTS.UTC_2150_001	2.1.5 Card update	file != NULL && newFile != NULL && feof(file) == false 상태에서 CID != fileCID 값이 들어온다.
PTS.UTC_2160_000	2.1.6 Card Reader	file != NULL 이다.

	Record	
--	--------	--

7.4.2 Recharger System

<Table 2.3 Test Design Identification>

Identifier	Feature	Valid/Invalid Value
PTS.UTC_120_000	1.2 Card Info Loader	Card.txt파일을 열어 입력받은 CID값이 있는 줄을 찾고, 찾으면 카드의 정보를 덮어씌운다.
PTS.UTC_120_001	1.2 Card Info Loader	Card.txt파일을 열어 입력받은 CID값이 있는 줄을 찾고, 찾지 못하면 카드의 정보를 초기화해 준다.
PTS.UTC_120_002	1.2 Card Info Loader	Card.txt파일이 지정한 상대경로에 존재하지 않는다면, 파일열기 실패를 출력하면서 프로그램이 종료된다.
PTS.UTC_211_000	2.1.1 Recharger Controller	적합한 CID값을 입력받아서, card_info값을 갱신한 후에는, money값을 입력 받고, 잔액을 충전한 후에, 충전된 정보로 교통카드를 갱신하고, 충전시각과 함께 교통카드에 충전된 정보를 Monitor에 보여준다.
PTS.UTC_211_001	2.1.1 Recharger Controller	적합한 CID값을 입력받지 못해서, card_info값을 0으로 모두 초기화한 후일지라도, money값을 입력 받고, 잔액을 충전한다. 충전된 정보로 교통카드를 갱신하려 하지만, Card.txt파일에서 일치하는 CID정보가 없으므로 실질적으로 갱신은 이루어지지 않는다. 이후, 충전시각과 함께 교통카드에 충전된 정보를 Monitor에 보여준다.
PTS.UTC_212_000	2.1.2. Update	입력받은 충전할 금액과 카드의 잔액을 더한 카드의 정보를 입력 받아서, Card.txt파일과 newCard.txt파일을 열은 후, Card.txt파일에서, 입력받은 CID값이 있는 줄이면 충전한 카드정보를 newCard.txt에 기록하고, 다른 CID값이 있는 줄이면 그대로 newCard.txt에 기록한다. 이후, Card.txt 파일은 제거하고, newCard.txt의 이름을 Card.txt로 변경한다.
PTS.UTC_212_001	2.1.2. Update	적합한 CID값을 입력받지 못해서, card_info값을 0으로 모두 초기화한 카드의 정보를 입력받을 경우, Card.txt파일에서, CID값이 0인 줄이 없으

		므로, Card.txt파일의 값이 그대로 newCard.txt에 기록되면서 실질적으로 바뀌는 부분은 없다. 이후, Card.txt 파일은 제거하고, newCard.txt의 이름을 Card.txt로 변경한다.
PTS.UTC.212.002	2.1.2. Update	Card.txt파일이 지정한 상대경로에 존재하지 않는다면, 파일열기 실패를 출력하면서 프로그램이 종료된다.
PTS.UTC.213.000	2.1.3. Display	적합한 CID값을 입력받아서, card_info값을 갱신한 후에는, 충전된 카드의 정보와 충전한 금액을 입력 받았을 때, 충전한 시각을 구한 후, 충전한 시각, 충전 전의 금액, 충전 한 금액, 충전 후의 금액을 보여준다.
PTS.UTC.213.001	2.1.3. Display	적합한 CID값을 입력받지 못해서, card_info값을 0으로 모두 초기화한 카드가 충전된 정보와 충전한 금액을 입력 받았을 때, 충전한 시각을 구한 후, 충전 한 시각, 충전 전의 금액(=0원), 충전 한 금액, 충전 후의 금액(=충전 한 금액)을 보여준다.

7.4.3 Fee Calculation System

<Table 3.3 Test Design Identification>

Identifier	Feature	Valid/Invalid value
FCS.UTC.010.000	1 Card Reader Info Loader	Tick()==0일 때, 각 단말기 파일 != NULL이면 모든 단말기 파일에 대하여 CardReader_info!=NULL 입력이 들어온다.
FCS.UTC.010.001	1 Card Reader Info Loader	Tick()==0일 때, 각 단말기 파일 == NULL이면 모든 단말기 파일에 대하여 CardReader_info!=NULL 입력이 들어온다.
FCS.UTC.010.002	1 Card Reader Info Loader	Tick()==1일 때, 각 단말기 파일 != NULL이면 모든 단말기 파일에 대하여 CardReader_info!=NULL 입력이 들어온다.
FCS.UTC.010.003	1 Card Reader Info Loader	Tick()==1일 때, 각 단말기 파일 == NULL이면 모든 단말기 파일에 대하여 CardReader_info==NULL 입력이 들어온다.
FCS.UTC.010.004	1 Card Reader Info Loader	CardReader_info[i].CID==1000일 때, CardReader_info[i]를 CID_1000.txt에 저장한다.
FCS.UTC.010.005	1 Card Reader Info Loader	CardReader_info[i].CID==1001일 때, CardReader_info[i]를 CID_1001.txt에 저장한다.
FCS.UTC.010.006	1 Card Reader Info Loader	CardReader_info[i].CID==1002일 때, CardReader_info[i]를 CID_1002.txt에 저장한다.

FCS.UTC.010.007	1 Card Reader Info Loader	CardReader_info[i].CID==1003일 때, CardReader_info[i]를 CID_1003.txt에 저장한다.
FCS.UTC.010.008	1 Card Reader Info Loader	CardReader_info[i].CID==1004일 때, CardReader_info[i]를 CID_1004.txt에 저장한다.
FCS.UTC.010.009	1 Card Reader Info Loader	CardReader_info[i].CID==1005일 때, CardReader_info[i]를 CID_1005.txt에 저장한다.
FCS.UTC.010.010	1 Card Reader Info Loader	CardReader_info[i].CID==1006일 때, CardReader_info[i]를 CID_1006.txt에 저장한다.
FCS.UTC.010.011	1 Card Reader Info Loader	CardReader_info[i].CID==NULL일 때, CardReader_info[i]를 CID_*.txt에 저장한다.
FCS.UTC.021.000	2.1 Fee Calculation Controller	AdjustStart상태 일 때, c_well==1을 입력한다.
FCS.UTC.021.001	2.1 Fee Calculation Controller	AdjustStart상태 일 때, c_well==0을 입력한다.
FCS.UTC.021.002	2.1 Fee Calculation Controller	GuestOut상태 일 때, state==1이면 getout==1을 입력한다.
FCS.UTC.021.003	2.1 Fee Calculation Controller	GuestOut상태 일 때, state==1이면 getout==0을 입력한다.
FCS.UTC.021.004	2.1 Fee Calculation Controller	GuestOut상태 일 때, state==0이면 getout==1을 입력한다.
FCS.UTC.021.005	2.1 Fee Calculation Controller	CID_Sort상태 일 때, CID_*.txt != NULL이면 CID_*.txt 데이터를 CardReader_info 배열에 저장한다.
FCS.UTC.021.006	2.1 Fee Calculation Controller	CID_Sort상태 일 때, CID_*.txt == NULL이면 CID_*.txt 데이터를 CardReader_info 배열에 저장한다.
FCS.UTC.021.007	2.1 Fee Calculation Controller	CID_Sort상태 일 때, CardReader_info배열이 !=NULL이면 tagTime이 작은 순으로 real_CardReader_info 배열에 저장한다.
FCS.UTC.021.008	2.1 Fee Calculation Controller	CID_Sort상태 일 때, CardReader_info배열이 ==NULL이면 tagTime이 작은 순으로 real_CardReader_info 배열에 저장한다.
FCS.UTC.021.009	2.1 Fee Calculation Controller	UnAdjust상태 일 때, real_CarfReader_info!=NULL, line!=NULL, &bus_fee!=NULL, &metro_fee!=NULL이면 Adjust를 실행한다.
FCS.UTC.021.010	2.1 Fee Calculation Controller	UnAdjust상태 일 때, real_CarfReader_info==NULL line==NULL &bus_fee==NULL &metro_fee==NULL이면 Adjust를 실행한다.
FCS.UTC.021.011	2.1 Fee Calculation Controller	Adjust상태 일 때, i>0, real_CardReader_info[i].price>1050면 real_Card_info[i].price += real_Card_info[i].price-1050, real_Card_info[i].price-=1050을 실행한다.
FCS.UTC.021.012	2.1 Fee Calculation Controller	Adjust상태 일 때, i>0, real_CardReader_info[i].price>1050면 real_Card_info[i].price += real_Card_info[i].price-1050, real_Card_info[i].price-=1050을 실행하지 않는다.
FCS.UTC.021.013	2.1 Fee Calculation Controller	Adjust상태 일 때, (배열의 처음) (배열의 마지막)

		(state==1, transfer==0)이면 다음 state==1, transfer==0인 전까지를 하거나 state==1, transfer==0인 곳이 없으면 배열의 처음이나 마지막이면 처음이나 마지막을 정산 계산을 할 구간을 정해서 index에 배열 인덱스를 넣는다.
FCS.UTC.021.014	2.1 Fee Calculation Controller	Adjust상태 일 때, (배열의 처음) (배열의 마지막) (state==1, transfer==0)이면 다음 state!=1, transfer!=0인 전까지 정산 계산을 할 구간을 정해서 index에 배열 인덱스를 넣는다.
FCS.UTC.021.015	2.1 Fee Calculation Controller	Adjust상태 일 때, index!=NULL이면 real_fee+=real_CardReader_info[i].price를 수행한다.
FCS.UTC.021.016	2.1 Fee Calculation Controller	Adjust상태 일 때, index==NULL이면 real_fee+=real_CardReader_info[i].price를 수행한다.
FCS.UTC.021.017	2.1 Fee Calculation Controller	Adjust상태 일 때, real_CardReader_info[i-1].tp != real_CardReader_info[i].tp이면 total_fee+=real_fee를 수행한다.
FCS.UTC.021.018	2.1 Fee Calculation Controller	Adjust상태 일 때, real_CardReader_info[i-1].tp == real_CardReader_info[i].tp이면 total_fee+=real_fee를 수행한다.
FCS.UTC.021.019	2.1 Fee Calculation Controller	Adjust상태 일 때, index!=NULL이면 temp_fee+=real_CardReader_info[i].price를 수행한다.
FCS.UTC.021.020	2.1 Fee Calculation Controller	Adjust상태 일 때, index==NULL이면 temp_fee+=real_CardReader_info[i].price를 수행한다.
FCS.UTC.021.021	2.1 Fee Calculation Controller	Adjust상태 일 때, ((real_CardReader_info[j-1].tp != real_CardReader_info[j].tp) i==j), real_CardReader_info[j].tp==0이면 *bus_fee+=temp_fee/total_fee*real_fee를 수행한다.
FCS.UTC.021.022	2.1 Fee Calculation Controller	Adjust상태 일 때, ((real_CardReader_info[j-1].tp == real_CardReader_info[j].tp) i!=j), real_CardReader_info[j].tp==0이면 *bus_fee+=temp_fee/total_fee*real_fee를 수행한다.
FCS.UTC.021.023	2.1 Fee Calculation Controller	Adjust상태 일 때, ((real_CardReader_info[j-1].tp != real_CardReader_info[j].tp) i==j), real_CardReader_info[j].tp==1이면 *metro_fee+=temp_fee/total_fee*real_fee를 수행한다.
FCS.UTC.021.024	2.1 Fee Calculation Controller	Adjust상태 일 때, ((real_CardReader_info[j-1].tp == real_CardReader_info[j].tp) i!=j), real_CardReader_info[j].tp==1이면 *metro_fee+=temp_fee/total_fee*real_fee를 수행한다.

FCS.UTC.021.025	2.1 Fee Calculation Controller	Index 구간이 끝나면, index=i-1, total_fee=0, real_fee=0, temp_fee=0를 수행한다.
FCS.UTC.021.026	2.1 Fee Calculation Controller	Index 구간이 끝나지 않아도, index=i-1, total_fee=0, real_fee=0, temp_fee=0를 수행한다.
FCS.UTC.021.027	2.1 Fee Calculation Controller	Adjus상태가 끝나고, bus_fee!=NULL, metro_fee!=NULL이면 c_well=0을 수행한다.
FCS.UTC.021.028	2.1 Fee Calculation Controller	Adjus상태가 끝나고, bus_fee==NULL, metro_fee==NULL이면 c_well=0을 수행한다.
FCS.UTC.021.029	2.1 Fee Calculation Controller	c_well==0면 enable Display.
FCS.UTC.021.030	2.1 Fee Calculation Controller	c_well!=0면 enable Display.
FCS.UTC.021.031	2.1 Fee Calculation Controller	c_well==0이면 Trigger Send
FCS.UTC.021.032	2.1 Fee Calculation Controller	c_well!=0이면 Trigger Send
FCS.UTC.021.033	2.1 Fee Calculation Controller	c_well==0이면 Trigger Reset
FCS.UTC.021.034	2.1 Fee Calculation Controller	c_well!=0이면 Trigger Reset
FCS.UTC.022.000	2.2 Display	Enable이 들어오고 c_well==0이면 bus_fee, metro_fee, t_now를 print한다
FCS.UTC.022.001	2.2 Display	Enable이 들어오고 c_well!=0이면 bus_fee, metro_fee, t_now를 print한다
FCS.UTC.022.002	2.2 Display	Enable이 들어오고 c_well!=0이면 bus_fee, metro_fee, t_now를 print안한다
FCS.UTC.023.000	2.3 Send	Trigger가 들어오고 c_well==0이면 c_well, bus_fee, metro_fee를 지하철 회사, 버스 회사에 보낸다.
FCS.UTC.023.001	2.3 Send	Trigger가 들어오고 c_well==0이면 c_well, bus_fee, metro_fee를 지하철 회사, 버스 회사에 보낸다.
FCS.UTC.023.002	2.3 Send	Trigger가 들어오고 c_well!=0이면 c_well, bus_fee, metro_fee를 지하철 회사, 버스 회사에 안보낸다.
FCS.UTC.024.000	2.4 Reset	Trigger가 들어오고 c_well == 0이면 모든 단말기파일을 초기화한다.
FCS.UTC.024.001	2.4 Reset	Trigger가 들어오고 c_well != 0이면 모든 단말기파일을 초기화한다.
FCS.UTC.024.002	2.4 Reset	Trigger가 들어오고 c_well != 0이면 모든 단말기파일을 초기화하지 않는다.

7.5 Feature pass/fail criteria

PTS, RS, FCS의 각 모듈(프로세스)은 SRA에 정의되어 있는 요구사항 (입력/ 출력, 동작)을 모두 만족해야 한다. 각 모듈(프로세스)의 입력/ 출력 및 동작은 SRA의 Process

description 항목 및 State Transition Diagram을 참조한다.

8 Unit test case specification

8.1 Test case specification identifier

8.1.1 Public Transportation System

<Table 1.4 Test Case Identification>

Test Case Identifier	Input Specification	Output Specification
PTS.UTC_1200_000	File!=NULL/ card_Info->CID==temp_card_info->CID	Card_info를 card_Reader_1.txt에 쓴다.
PTS.UTC_1200_001	File!=NULL/ card_Info>CID!=temp_card_info->CID	stat = InvalidInput; card_info의 값을 0으로 초기화시킨다.
PTS.UTC_1200_002	File==Null	printf("파일이 없습니다. 파일 열기 실패\n")
PTS.UTC_2111_000	stat==Normal	Error검사를 실행하고 가격을 책정한다.
PTS.UTC_2111_001	stat!=Normal	
PTS.UTC_2112_000	interval_sec <=0	interval_sec += 60;
PTS.UTC_2112_001	state==1/interval_sec<15/g etout==0/CRID==card_info->CRID	interval_sec += 60; stat = HopInProgress;
PTS.UTC_2112_002	state==1/interval_sec>=15 /getout==0/CRID==card_info->CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2112_003	state==1/interval_sec<15/g etout!=0/CRID==card_info->CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2112_004	state==1/interval_sec<15/g etout==0/CRID!=card_info->CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2112_005	state==1/interval_sec<15/g etout!=0/CRID!=card_info->CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2112_006	state==1/interval_sec>=15 /getout!=0/CRID==card_inf	Stat의 값을 변경시키지 않는다.

	o->CRID	
PTS.UTC_2112_007	state==1/interval_sec>=15 /getout==0/CRID!=card_inf o->CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2112_008	state==1/interval_sec>=15 /getout!=0/CRID!=card_inf o->CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2112_009	state==0/interval_sec<15/g etout==0/CRID==card_info ->CRID	stat = GetOffProcessing;
PTS.UTC_2112_010	state==0/interval_sec>=15 /getout==0/CRID==card_in fo->CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2112_011	state==0/interval_sec<15/g etout!=0/CRID==card_info- >CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2112_012	state==0/interval_sec<15/g etout==0/CRID!=card_info- >CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2112_013	state==0/interval_sec<15/g etout!=0/CRID!=card_info- >CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2112_014	state==0/interval_sec>=15 /getout!=0/CRID==card_inf o->CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2112_015	state==0/interval_sec>=15 /getout==0/CRID!=card_inf o->CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2112_016	state==0/interval_sec>=15 /getout!=0/CRID!=card_inf o->CRID	Stat의 값을 변경시키지 않는다.
PTS.UTC_2113_000	interval_sec <= 0	interval_sec += 60 * 60
PTS.UTC_2113_001	CRID < 10	tp = 0;
PTS.UTC_2113_002	CRID >= 10	tp = 1;
PTS.UTC_2113_003	(CRID%10) == 1 / state == 1 / transfer == 1 / tp == 1	price = 1650
PTS.UTC_2113_004	(CRID%10) == 1 / state == 1 / transfer == 1 / tp == 0	price = 1550

PTS.UTC_2113_005	(CRID%10) == 1 / state == 1 / transfer == 1 / tp != 1 / tp != 0	Error 출력
PTS.UTC_2113_006	(CRID%10) == 1 / state == 1 / transfer == 0 / tp == 1	price = 1250
PTS.UTC_2113_007	(CRID%10) == 1 / state == 1 / transfer == 0 / tp == 0	price = 1050
PTS.UTC_2113_008	(CRID%10) == 1 / state == 1 / transfer == 0 / tp != 1 / tp != 0	Error 출력
PTS.UTC_2113_009	(CRID%10) == 1 / state == 1 / transfer != 0 / transfer != 1	Error 출력
PTS.UTC_2113_010	(CRID%10) == 1 / state == 0 / transfer == 0 / interval_sec <= 15 / tp == card_info->tp	price = 1050
PTS.UTC_2113_011	(CRID%10) == 1 / state == 0 / transfer == 0 / interval_sec <= 15 / tp == 1 / card_info->tp == 0	price = 600
PTS.UTC_2113_012	(CRID%10) == 1 / state == 0 / transfer == 0 / interval_sec <= 15 && tp == 0 && card_info->tp == 1	price = 500
PTS.UTC_2113_013	(CRID%10) == 1 / state == 0 / transfer == 0 / interval_sec <= 15 / tp != 0 / tp != 1 / card_info->tp != 0 / card_info->tp != 1	Error 출력
PTS.UTC_2113_014	(CRID%10) == 1 / state == 0 / transfer == 0 / interval_sec > 15	price = 1050
PTS.UTC_2113_015	(CRID%10) == 1 / state == 0 / transfer != 0	Error출력
PTS.UTC_2113_016	(CRID%10) == 0 / state ==	price = 300

	1 / transfer == 1 / tp == 1 / interval_station == 1 interval_station == 4	
PTS.UTC_2113_017	(CRID%10) == 0 / state == 1 / transfer == 1 / tp == 1 / interval_station == 2 interval_station == 3	price = 600
PTS.UTC_2113_018	(CRID%10) == 0 / state == 1 / transfer == 1 / tp == 1 / interval_station < 1 interval_station > 4	price = 0
PTS.UTC_2113_019	(CRID%10) == 0 / state == 1 / transfer == 1 / tp == 0 / (interval_sec/30) > 5	price = 500
PTS.UTC_2113_020	(CRID%10) == 0 / state == 1 / transfer == 1 / tp == 0 / (interval_sec/30) <= 5	price = interval_sec / 30 * 100
PTS.UTC_2113_021	(CRID%10) == 0 / state == 1 / transfer == 1 / tp != 0 / tp != 1	Error 출력
PTS.UTC_2113_022	(CRID%10) == 0 / state == 1 / transfer == 0 / tp == 1 / interval_station == 1 interval_station == 4	price = 0
PTS.UTC_2113_023	(CRID%10) == 0 / state == 1 / transfer == 0 / tp == 1 / interval_station == 2 interval_station == 3	price = 200
PTS.UTC_2113_024	(CRID%10) == 0 / state == 1 / transfer == 0 / tp == 1 / interval_station < 1 interval_station > 4	price = 0
PTS.UTC_2113_025	(CRID%10) == 0 / state == 1 / transfer == 0 / tp == 0	price = 0
PTS.UTC_2113_026	(CRID%10) == 0 / state == 1 / transfer == 0 / tp != 1 / tp != 0	Error 출력
PTS.UTC_2113_027	(CRID%10) == 0 / state !=	Error 출력

	1	
PTS.UTC_2113_028	(CRID%10)!=0/ (CRID%10) != 1	Error 출력
PTS.UTC_2121_000	stat == Normal	Short()호출
PTS.UTC_2121_001	Short() / stat == Normal	Calculation()호출
PTS.UTC_2122_000	(card_info->cash-price) < 0	stat = EShort
PTS.UTC_2122_000	CRID < 10	tp = 0
PTS.UTC_2123_001	CRID >= 10	tp = 1
PTS.UTC_2123_002	card_info->transfer == 0	card_info->cash -= price
PTS.UTC_2123_003	(CRID%10) == 0	card_info->state = 0
PTS.UTC_2123_004	(CRID%10) != 0	card_info->state = 1
PTS.UTC_2130_000	stat == Normal	CardUpdate(), CardReaderRecord() 호출
PTS.UTC_2140_000	stat == Normal / transfer == 1	0, cash 출력
PTS.UTC_2140_001	stat == Normal / transfer != 1	price, cash 출력
PTS.UTC_2140_002	stat == HopnInProcessing	Error메시지 출력
PTS.UTC_2140_003	stat == GetoffProcessing	Error메시지 출력
PTS.UTC_2140_004	stat == EShort	Error메시지 출력
PTS.UTC_2140_005	stat == NotAddjust	Error메시지 출력
PTS.UTC_2140_006	stat == InvalidInput	Error메시지 출력
PTS.UTC_2150_000	file != NULL / newFile != NULL / feof(file) == false / CID == fileCID	newFile에 update된 card_info를 기록한다.
PTS.UTC_2150_001	file != NULL / newFile != NULL / feof(file) == false / CID == fileCID	기존의 카드값을 유지한다.
PTS.UTC_2160_000	file != NULL	card_info를 card_Reader_1.txt에 쓴다.

8.1.2 Recharger System

<Table 2.4 Test Case Identification>

Test Case Identifier	Input specification	Output specification
PTS.UTC_120_000	CID=1000 /	card_info={CID=1000, CRID=11,

	if(file != NULL)	tagTime={2014, 11, 20, 19, 52, 18}, tp=1, state=1, cash=30900, transfer=0, getout=1}
PTS.UTC_120_001	CID=100 / if(file != NULL)	card_info={CID=0, CRID=0, tagTime={?, ?, ?, ?, ?}, tp=0, state=0, cash=0, transfer=0, getout=0}
PTS.UTC_120_002	if(file == NULL)	"파일이 없습니다. 파일 열기 실패" exit(1);exit(1);
PTS.UTC_211_000	CID=1000 / money=20000	card_info.cash+=money / trigger "Update"/ trigger "Display"
PTS.UTC_211_001	CID=100 / money=20000	card_info.cash+=money / trigger "Update"/ trigger "Display"
PTS.UTC_212_000	card_info={CID=1000, CRID=11, tagTime={2014, 11, 20, 19, 52, 18}, tp=1, state=1, cash=50900, transfer=0, getout=1}	fprintf(newFile, "...", card_info->CID, ... card_info->getout);이 한번 수행
PTS.UTC_212_001	card_info={CID=0, CRID=0, tagTime={?, ?, ?, ?, ?}, tp=0, state=0, cash=0, transfer=0, getout=0}	fprintf(newFile, "...", card_info->CID, ... card_info->getout);이 한번도 수행되지 못함.
PTS.UTC_212_002	if(file == NULL)	"파일이 없습니다. 파일 열기 실패" exit(1);
PTS.UTC_213_000	card_info={CID=1000, CRID=11, tagTime={2014, 11, 20, 19, 52, 18}, tp=1, state=1, cash=50900, transfer=0, getout=1} / money=20000	"충전 한 시각 : 2014년 11월 21일 03시 16 분 44초" "충전 전의 금액 : 30900원" "충전 한 금액 : 20000원" "충전 후의 금액 : 50900원"
PTS.UTC_213_001	card_info={CID=0, CRID=0, tagTime={?, ?, ?, ?, ?}, tp=0, state=0, cash=0, transfer=0, getout=0} / money=20000	"충전 한 시각 : 2014년 11월 21일 03시 16 분 44초" "충전 전의 금액 : 0원" "충전 한 금액 : 20000원" "충전 후의 금액 : 20000원"

8.1.3 Fee Calculation System

<Table 3.4 Test Case Identification>

Test Case Identifier	Input Specification	Output Specification
FCS.UTC.010.000	Tick()==0/ *.txt(단말기)!=NULL	fscanf(, &CardReader_info[i].CID,,,,,)
FCS.UTC.010.001	Tick()==0/ *.txt(단말기)==NULL	
FCS.UTC.010.002	Tick()==1/ *.txt(단말기)!=NULL	
FCS.UTC.010.003	Tick()==1/ *.txt(단말기)==NULL	
FCS.UTC.010.004	CardReader_info[i].CID==1000	fprint(file_CID_1000 , ,CardReader_info[i].CID,,,,,)
FCS.UTC.010.005	CardReader_info[i].CID==1001	fprint(file_CID_1001, ,CardReader_info[i].CID,,,,,)
FCS.UTC.010.006	CardReader_info[i].CID==1002	fprint(file_CID_1002, ,CardReader_info[i].CID,,,,,)
FCS.UTC.010.007	CardReader_info[i].CID==1003	fprint(file_CID_1003, ,CardReader_info[i].CID,,,,,)
FCS.UTC.010.008	CardReader_info[i].CID==1004	fprint(file_CID_1004, ,CardReader_info[i].CID,,,,,)
FCS.UTC.010.009	CardReader_info[i].CID==1005	fprint(file_CID_1005, ,CardReader_info[i].CID,,,,,)
FCS.UTC.010.010	CardReader_info[i].CID==1006	fprint(file_CID_1006, ,CardReader_info[i].CID,,,,,)
FCS.UTC.010.011	CardReader_info[i].CID>1006 CardReader_info[i].CID<1000	
FCS.UTC.021.000	AdjustStart()	c_well=1
FCS.UTC.021.001	!AdjustStart()	c_well=1
FCS.UTC.021.002	CardReader_info[i].state==1	CardReader_info[i].getout=1
FCS.UTC.021.003	CardReader_info[i].state==1	CardReader_info[i].getout=0
FCS.UTC.021.004	CardReader_info[i].state==0	CardReader_info[i].getout=1
FCS.UTC.021.005	CID_Sort()/CID_*.txt !=NULL	fscanf(CID_*.txt, ,temp_CardReader_info[i].CID,,,,)
FCS.UTC.021.006	CID_Sort()/CID_*.txt ==NULL	
FCS.UTC.021.007	CID_Sort()/temp_CardReader_info[i]!=NULL &temp_CardReader_info[i].tagTime작은 순 으로	real_CardReader_info[i]=temp_CardReader_info[j]
FCS.UTC.021.008	CID_Sort()/temp_CardReader_info[i]==NULL &temp_CardReader_info[i].tagTime작은 순 으로	
FCS.UTC.021.009	real_CarfReader_info!=NULL, line!=NULL, &bus_fee!=NULL, &metro_fee!=NULL	Adjust();
FCS.UTC.021.010	real_CarfReader_info==NULL line==NULL &bus_fee==NULL &metro_fee==NULL	
FCS.UTC.021.011	i>0, real_CardReader_info[i].price>1050	real_Card_info[i].price += real_Card_info[i].price- 1050,

		real_Card_info[i].price-=1050.
FCS.UTC.021.012	<code>i<=0 real_CardReader_info[i].price<=1050</code>	<code>real_Card_info[i].price += real_Card_info[i].price-1050,</code> <code>real_Card_info[i].price-=1050.</code>
FCS.UTC.021.013	<code>Index=size-1;</code> <code>for(i=size-1;i>=0;i--)</code> <code>if((real_card_info[i].state==1)&&</code> <code>(real_card_info[i].transfer==0))</code>	<code>index=i-1;</code>
FCS.UTC.021.014	<code>Index=size-1;</code> <code>for(i=size-1;i>=0;i--)</code> <code>if((real_card_info[i].state!=1) </code> <code>(real_card_info[i].transfer!=0))</code>	.
FCS.UTC.021.016	<code>Index=size-1;</code> <code>for(i=size-1;i>=0;i--)</code> <code>{</code> <code>if((real_card_info[i].state==1)&&</code> <code>(real_card_info[i].transfer==0))</code> <code>{</code> <code>for(j=index;j>=i;j--){</code>	<code>real_fee+=real_card_info[j].price</code>
FCS.UTC.021.017	<code>Index=NULL</code>	<code>real_fee+= real_CardReader_info[i].price</code>
FCS.UTC.021.018	<code>real_CardReader_info[i-1].tp !=</code> <code>real_CardReader_info[i].tp</code>	<code>total_fee+=real_fee</code>
FCS.UTC.021.019	<code>real_CardReader_info[i-1].tp ==</code> <code>real_CardReader_info[i].tp</code>	<code>total_fee+=real_fee</code>
FCS.UTC.021.020	<code>Index=size-1;</code> <code>for(i=size-1;i>=0;i--)</code> <code>{</code> <code>if((real_card_info[i].state==1)&&</code> <code>(real_card_info[i].transfer==0))</code> <code>{</code> <code>for(j=index;j>=i;j--){</code>	<code>temp_fee+=real_CardReader_info[i].price</code>
FCS.UTC.021.021	<code>index=NULL</code>	<code>temp_fee+=real_CardReader_info[i].price</code>
FCS.UTC.021.022	<code>if(j==i (real_card_info[j-1].tp</code> <code>!=real_card_info[j].tp)){</code> <code>if(real_card_info[j].tp==0)</code>	<code>*bus_fee+=temp_fee/total_fee*real_fee</code>
FCS.UTC.021.023	<code>if(j!=i (real_card_info[j-1].tp</code> <code>==real_card_info[j].tp)){</code> <code>if(real_card_info[j].tp==0)</code>	<code>*bus_fee+=temp_fee/total_fee*real_fee</code>

FCS.UTC.021.024	if(j==i (real_card_info[j-1].tp !=real_card_info[j].tp)){ if(real_card_info[j].tp==1)	Adjust상태 일 때, ((real_CardReader_info[i-1].tp !=real_CardReader_info[i].tp) i==0), real_CardReader_info[i].tp==1이면 *metro_fee+=temp_fee/total_fee*real_fee를 수행한다.
FCS.UTC.021.025	if(j!=i (real_card_info[j-1].tp ==real_card_info[j].tp)){ if(real_card_info[j].tp==1)	*metro_fee+=temp_fee/total_fee*real_fee
FCS.UTC.021.026	Index=size-1; for(i=size-1;i>=0;i--) { if((real_card_info[i].state==1)&&(real_card_info[i].transfer==0)) { for(j=index; j>=i; j--){ } }	index=i-1, total_fee=0, real_fee=0, temp_fee=0
FCS.UTC.021.027	If문 끝난 다음이 아닐때	index=i-1, total_fee=0, real_fee=0, temp_fee=0
FCS.UTC.021.028	bus_fee!=NULL, metro_fee!=NULL	c_well=0
FCS.UTC.021.029	bus_fee==NULL, metro_fee==NULL	c_well=0
FCS.UTC.021.029	c_well==0	enable Display.
FCS.UTC.021.030	c_well!=0	enable Display.
FCS.UTC.021.031	c_well==0	Trigger Send
FCS.UTC.021.032	c_well!=0	Trigger Send
FCS.UTC.021.033	c_well==0	Trigger Reset
FCS.UTC.021.034	c_well!=0	Trigger Reset
FCS.UTC.022.000	Enable/c_well==0	printf("버스 정산 금액: %d\n",bus_fee); printf("지하철 정산 금액: %d\n",metro_fee);
FCS.UTC.022.001	Enable/c_well==0	printf("버스 정산 금액: %d\n",bus_fee); printf("지하철 정산 금액: %d\n",metro_fee);
FCS.UTC.022.002	c_well!=0	
FCS.UTC.023.000	Trigger/c_well==0	file_bus=fopen("send_bus.txt","w"); fprintf(file_bus,"%d\n",c_well); fprintf(file_bus,"정산금액: %d\n",bus_fee); file_metro=fopen("send_metro.txt","w"); fprintf(file_metro,"%d\n",c_well); fprintf(file_metro,"총 정산금액: %d",metro_fee);

FCS.UTC.023.001	Trigger/c_well==0	file_bus=fopen("send_bus.txt","w"); fprintf(file_bus,"%d\n",c_well); fprintf(file_bus,"정산금액: %d\n",bus_fee); file_metro=fopen("send_metro.txt","w"); fprintf(file_metro,"%d\n",c_well); fprintf(file_metro,"총 정산금액: %d",metro_fee);
FCS.UTC.023.002	c_well!=0	
FCS.UTC.024.000	Trigger/c_well==0	remove(fname); rename("temp.txt",fname);
FCS.UTC.024.001	Trigger/c_well==0	remove(fname); rename("temp.txt",fname);
FCS.UTC.024.002	c_well!=0	

8.2 Test items

8.2.1 Public Transportation System

<Table 1.3 Test Design Identification> 참조

8.2.2 Recharger System

<Table 2.3 Test Design Identification> 참조

8.2.3 Fee Calculation System

<Table 3.3 Test Design Identification> 참조

8.3 Input specifications

8.3.1 Public Transportation System

<Table 1.4 Test Case Identification> 참조

8.3.2 Recharger System

<Table 2.4 Test Case Identification> 참조

8.3.3 Fee Calculation System

<Table 3.4 Test Case Identification> 참조

8.4 Output specifications

8.4.1 Public Transportation System

<Table 1.4 Test Case Identification>참조

8.4.2 Recharger System

<Table 2.4 Test Case Identification>참조

8.4.3 Fee Calculation System

<Table 3.4 Test Case Identification>참조

9 Testing tasks

<Table 5 Testing tasks & Schedule>

Task	Predecessor tasks	Special skills	Effort	Finish Date
Unit Test Plan 작성	T3.2014.PTS.SRS 작성 T3.2014.RGS.SRS 작성 T3.2014.FCS.SRS 작성 PTS,RS,FCS 구현		3	
Test Design Specification	Task1	PTS,RS,FCS에 대한 이해	5	
Test Case Specification	Task2	PTS,RS,FCS에 대한 이해	5	
Test Execution	Task3	Test Code 작성	4	
Test result report	Task4		2	

10 Environmental needs

C Language Compiler/Linker Ex)Visual Studio 2010

11 Unit Test deliverables

12 Schedules

<Table 5 Testing tasks & Schedule>참조