Unit Testing Plan

for Public Transportation System

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

Project Team

Team 4

Date

2014-11-20

노은방 200811428
김상민 200910044
박수민 201111353
한별 201214217

목차

1 Introduction

1.1 Objectives

본 문서는 2014년 2학기 소프트웨어 공학 개론 수업의 T4가 개발한 Public Transportation System(PTS)을 Unit Testing하기 위한 계획문서이다. T4가 정의한 Unit Testing을 수행하기 위하여 Testing Pass/Fail Criteria를 정의하고 이를 수행하기 위한 Test design & test cases를 제작한다.

1.2 Background

2014년 2학기 소프트웨어 공학 개론 수업에서 개발하는 모든 PTS는 SASD기법을 이용하여 개발된다. 기능별로 나눠진 Unit은 SRA, SDS 문서에 모두 정의되어 있다.

1.3 Scope

SRA 및 SDS 문서에 정의된 Unit을 Testing한다.

1.4 Project plan

1.5 Configuration management plan

1.6 References

SRA3

SDS3

2 Test items

T4가 SASD기법을 이용하여 개발한 PTS를 Testing한다. SA와 SD에서 분류한 각 process/module 별로 Testing을 수행한다. <Figure 1 Overall DFD>은 SA를 이용하여 요구사항을 분석한 결과 중 전체 버스 DFD를 나타낸 그림이며, <Figure 2 Overall DFD>은 전체 지하철 DFD를 나타낸 그림이고, <Figure 3 Overall DFD>은 전체 정산 DFD를 나타낸 그림이다. 그리고 <Figure 4 Structural Chart>은 SD의 버스 Basic Structural Chart를 나타낸 그림이며, <Figure 5 Structural Chart>은 SD의 지하철 Basic Structural Chart를 나타낸 그림이고, 그리고 <Figure 6 Structural Chart>은 SD의 정산 Basic Structural Chart를 나타낸 그림이다. 각 그림을 참조하여 Unit을 지정하고, 지정한 Unit을 SRA에 명세 된 내용과 같은 동작을 하는지 확인한다.

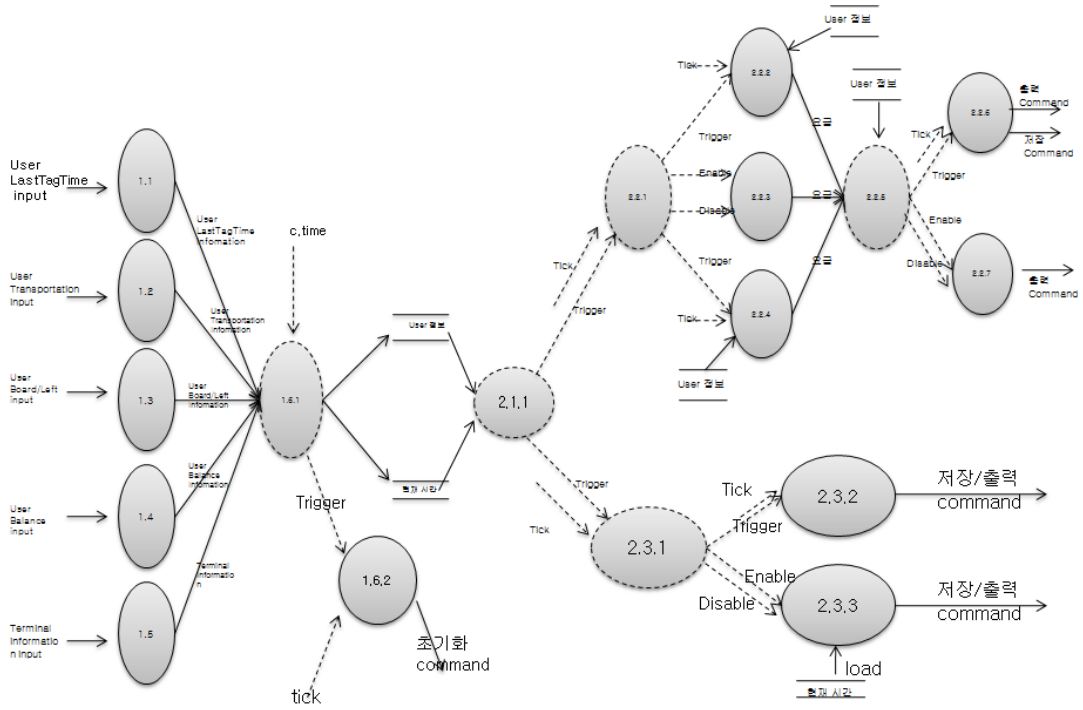

Figure 1 Overall DFD

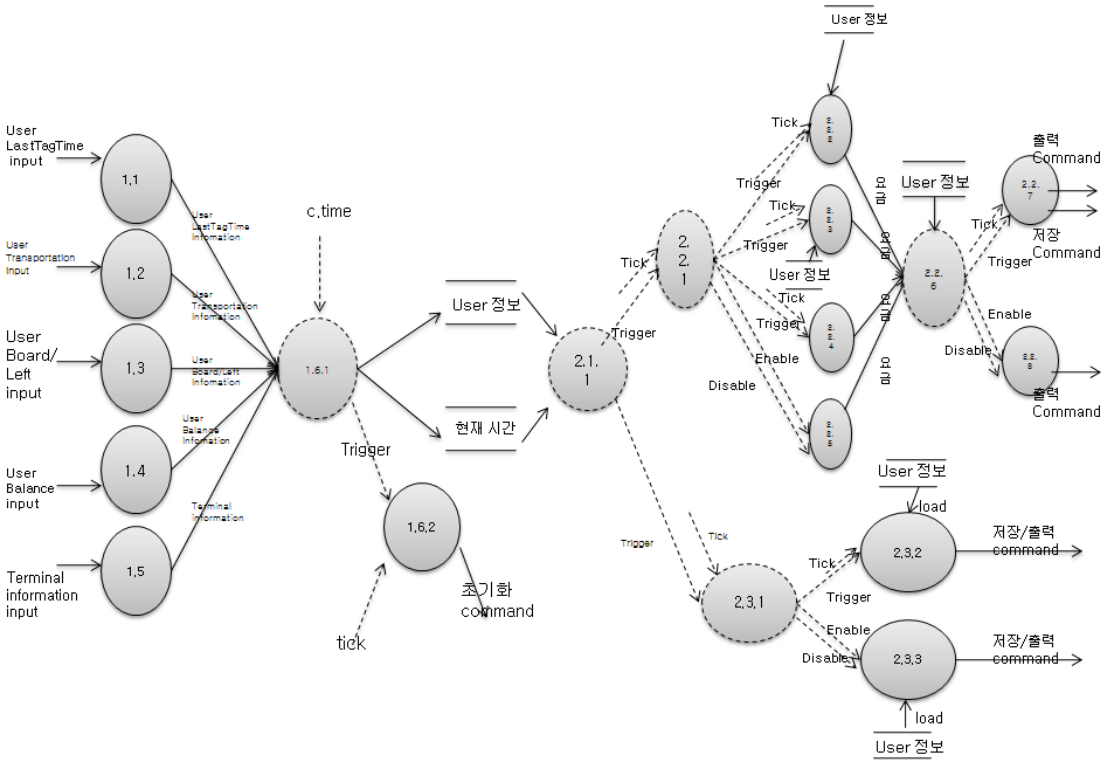

Figure 2 Overall DFD

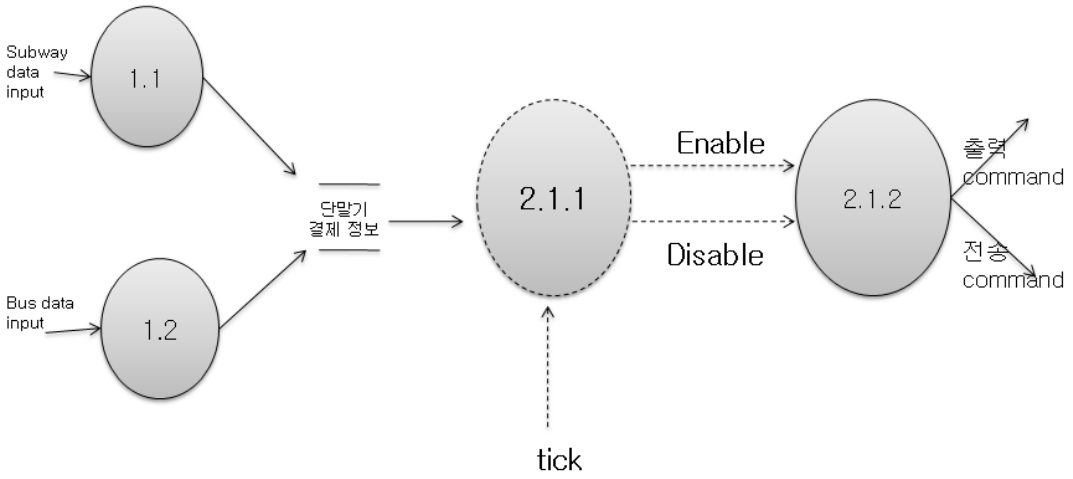

Figure 3 Overall DFD

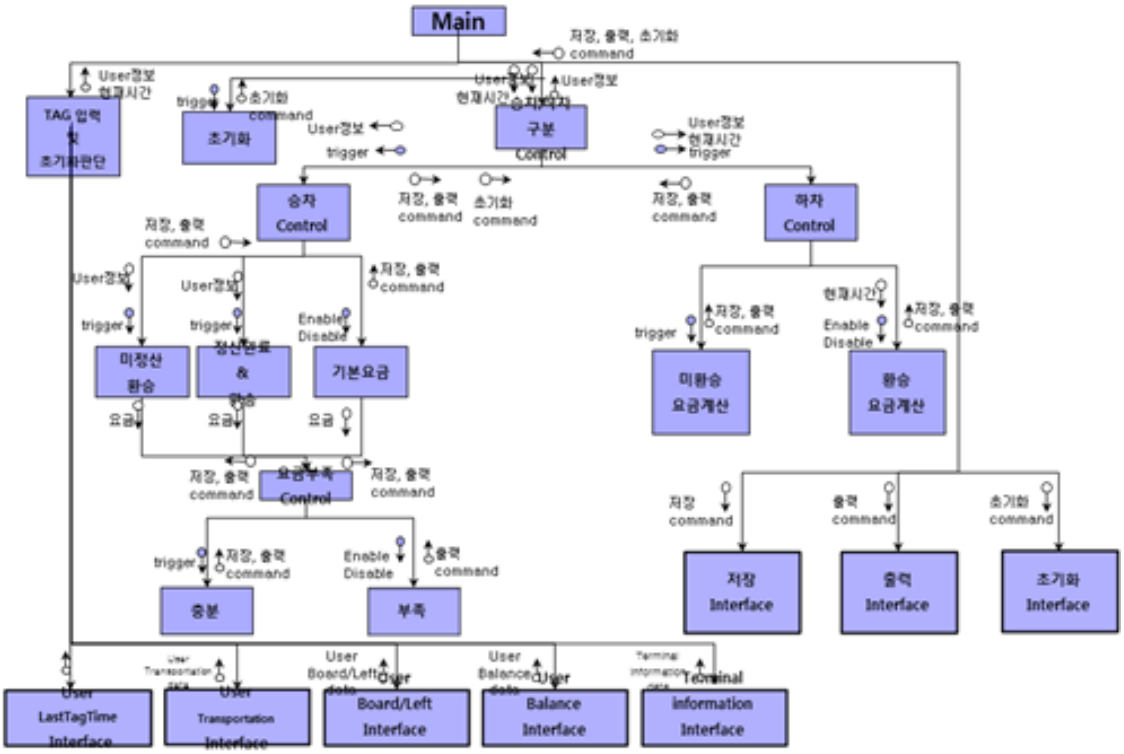

Figure 4 Structural Chart

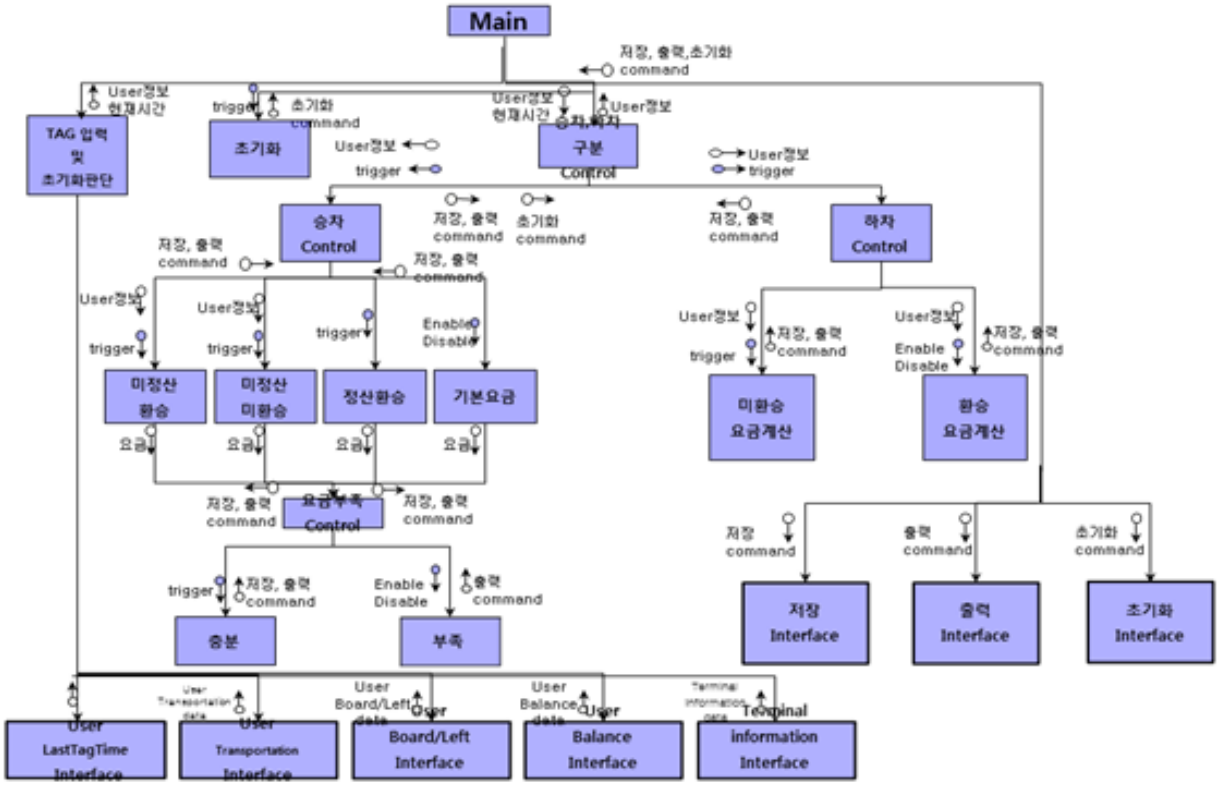

Figure 5 Structural Chart

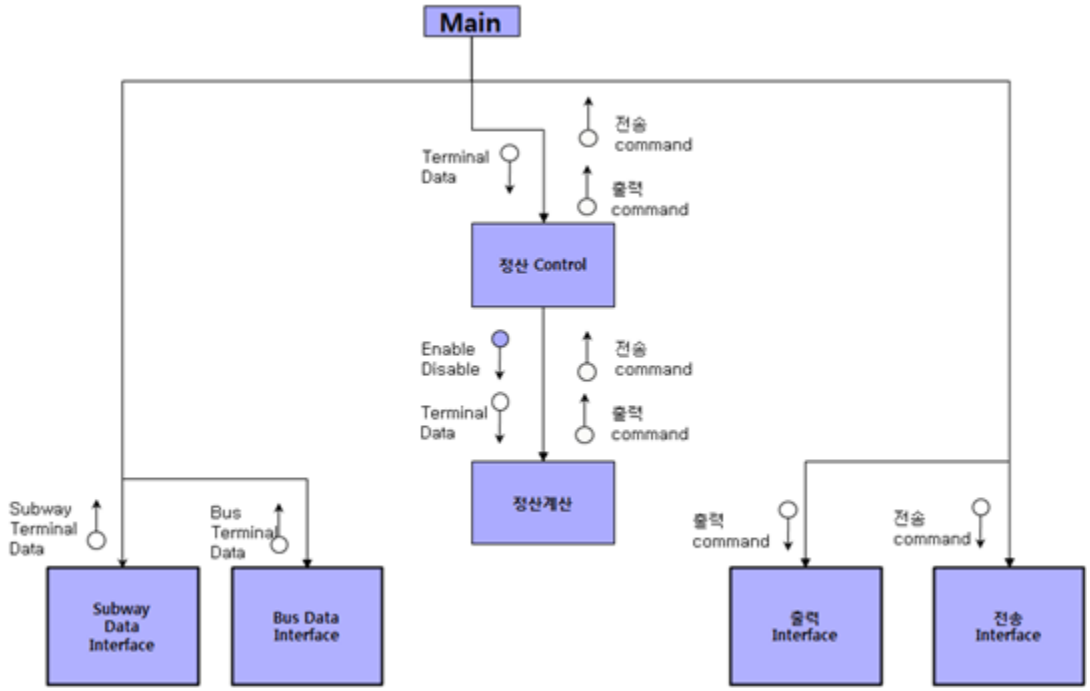

Figure 6 Structural Chart

3 Features to be tested

모듈 별로 입력부와 출력부를 거쳐 테스트한다.

4 Features not to be tested

이번 Unit Test에서는 통합하여 전체적으로 테스트하지 않는다.

5 Approach

Brute Force, SRA의 가장 하위 Level의 Process부터 Testing을 시작하여 상위 Level의 Process 까지 Testing한다

6 Item pass/fail criteria

모든 Unit은 SRA에 명시된 Process Specification과 동일한 결과를 출력해야 한다. 각 Unit별 Pass/Fail Criteria는 <Table2 Test Case Identification>를 참조한다.

7 Unit test design specification

7.1 Test design specification identifier

PTS.UTC.*Number*

7.2 Features to be tested

7.3 Approach refinements

각 Process Specification에 명시된 내용을 기반으로 Test Design 및 Test Cases를 생성해 낸다

7.4 Test identification

7.5 Feature pass/fail criteria

8 Unit test case specification

8.1 Test case specification identifier

8.2 Test items

8.3 Input specifications

8.4 Output specifications

9 Testing tasks

Table 1 Testing tasks & Schedule

Task	Predecessor tasks	Special skills	Effort	Finish date
(1) Unit Test Plan 작성	SRA 작성 SDS 작성 DWS 구현		3	
(2) Test design specification	Task 1	PTS에 대한 이해		
(3) Test case specification	Task 2	PTS에 대한 이해		
(4) Test execution	Task 3	Test code 작성 Test tools에 대한 이해		
(5) Test result report	Task 4		1	
(6) 개발팀에 report 전달	Task 5		1	

10 Environmental needs

11 Unit Test deliverables

Unit Testing Report

12 Schedules

<Table 1 Testing tasks & Schedule> 참조

Table 2 Test Design Identification-버스

Identifier	Feature	Pass/Fail
PTS.UTC.1000	유저가 태그 하였을 시 유저 카드에 저장된 정보를 입력 받아 저장해주는 부분	pass
PTS.UTC.1100	받은 사용자 카드에서 마지막으로 태그 된 시간 정보를 전달한다.	Pass
PTS.UTC.1200	받은 사용자 카드에 저장된 마지막으로 태그 된 교통수단 정보를 전달한다.	Pass
PTS.UTC.1300	받은 사용자 카드에 저장된 마지막으로 태그 된 승.하차 정보를 전달한다.	Pass
PTS.UTC.1400	받은 사용자 카드에 저장된 잔액 정보를 전달한다.	Pass
PTS.UTC.1500	받은 사용자 카드에 저장된 마지막으로 태그 된 단말기 정보를 전달한다.	Pass
PTS.UTC.1600	유저가 태그를 했을 시 유저의 입력을 받은 부분을 모아서 컨트롤 부에서 이용할 수 있는 자료구조로 따로 저장해준다. 또한 태그를 한 시간을 따로 저장해준다.	Pass
PTS.UTC.2000	Tag 정보에서 입력을 받아 정보처리를 한 후 각각 화면에 출력하거나 저장을 한다.	Pass
PTS.UTC.2100	승객이 승차를 하는지 하차를 하는지에 따라서 case로 나눈다.	Pass
PTS.UTC.2200	승객이 승차를 했을 때 지불해야 할 요금계산 후 요금의 충분 부족에 따라 각 저장/출력 명령을 출력한다.	Pass
PTS.UTC.2300	승객이 하차를 했을 때 지불해야 할 요금계산 후 저장/출력 명령을 출력한다.	Pass
PTS.UTC.2400	저장에 대한 명령어를 받아서 각각 지하철과 버스 단말기 정보 저장소 파일에 요금 정보를 포함해서 저장을 하고, 승객의 카드정보에도 잔액 정보를 포함해서 저장을 한다.	pass
PTS.UTC.2500	출력에 대한 명령어를 받아서 요금과 현재 시간에 대한 정보를 단말기(여기선 터미널)에 출력한다.	Pass
PTS.UTC.1610	유저가 태그를 했을 시 유저의 입력을 받은 부분을 모아서 컨트롤 부에서 이용 할 수 있는 자료구조로 따로 저장해준다. 또한 태그 한 시간을 따로 저장해 준다. 그리	pass

	고 3분마다 초기화 판단 및 시행을 해준다.	
PTS.UTC.1620	3분주기마다 시행이 되며 유저카드 이외의 정보를 초기화하는 command를 출력한다.	Pass
PTS.UTC.2110	승객의 승차, 하차를 구분 짓는 컨트롤로 사용자 정보와 현재시간을 입력 받아 승차인지 하차인지 결정 짓는다.	Pass
PTS.UTC.2210	실질적으로 요금의 경우를 나누어 주는 주 컨트롤러로, 사용자의 정보를 입력 받고, 환승 여부, 미정산 여부, 최초 탑승여부의 조합에 따라 경우를 나누어 각 계산해야 되는 요금식을 판단해주고 다음 프로세스에 트리거를 전송해준다.	Pass
PTS.UTC.2220	사용자가 이전에 정산을 하고 환승을 하였을 경우의 금액을 계산하고, 최대금액을 계산한 후 해당 금액들을 다음 컨트롤러로 전달해준다.	Pass
PTS.UTC.2230	사용자가 최초탑승이거나 이전에 버스를 탔거나 환승시간이 넘은 경우, 기본금액을 다음 컨트롤러로 전달해 준다.	Pass
PTS.UTC.2240	사용자가 미정산을 하고 환승을 안한 경우의 금액을 계산하고, 최대금액을 계산한 후 해당 금액들을 다음 컨트롤러로 전달해준다.	Pass
PTS.UTC.2250	전달받은 금액을 유저의 잔액과 비교해서 요금부족 여부를 판단하여 각각 트리거를 전달 해준다.	Pass
PTS.UTC.2260	요금정보와 그에 해당하는 정보를 사용자 카드와 단말기에 각각 저장시키고, 금액과 시간을 출력하는 명령을 전송한다.	Pass
PTS.UTC.2270	최종처리에 금액이 부족한 경우 활성화 되며, 금액 부족 문구를 출력하는 명령을 전송한다.	Pass
PTS.UTC.2310	사용자가 하차태그를 하였을 시 환승을 하였는지 안 하였는지의 여부에 대한 트리거를 다음 프로세스에 전송해준다.	Pass
PTS.UTC.2320	하차시 미환승 하였을 경우 지금까지 이동한 시간정보를 입력 받아서 해당하는 금액을 계산하고 해당 금액과 해당하는 정보를 사용자 카드와 단말기에서 각각 저장시키고, 금액과 시간을 출력한다.	pass

PTS.UTC.2330	하차시 환승 하였을 경우 지금까지 이동한 시간정보를 입력 받아서 해당하는 금액을 계산하고 해당금액과 해당하는 정보를 사용자 카드와 단말기에 각각 저장시키고, 금액과 시간을 출력한다.	pass

Table 3 Test design Identification-지하철

Identifier	Feature	Pass/Fail
PTS.UTC.1000	유저가 태그 하였을 시 유저 카드에 저장된 정보를 입력 받아 저장해주는 부분	pass
PTS.UTC.1100	받은 사용자 카드에서 마지막으로 태그 된 시간 정보를 전달한다.	pass
PTS.UTC.1200	받은 사용자 카드에 저장된 마지막으로 태그 된 교통수단 정보를 전달한다.	pass
PTS.UTC.1300	받은 사용자 카드에 저장된 마지막으로 태그 된 승.하차 정보를 전달한다.	pass
PTS.UTC.1400	받은 사용자 카드에 저장된 잔액 정보를 전달한다.	pass
PTS.UTC.1500	받은 사용자 카드에 저장된 마지막으로 태그 된 단말기 정보를 전달한다.	pass
PTS.UTC.1600	유저가 태그를 했을 시 유저의 입력을 받은 부분을 모아서 컨트롤 부에서 이용할 수 있는 자료구조로 따로 저장해준다. 또한 태그를 한 시간을 따로 저장해준다.	pass
PTS.UTC.2000	Tag 정보에서 입력을 받아 정보처리를 한 후 각각 화면에 출력하거나 저장을 한다.	pass
PTS.UTC.2100	승객이 승차를 하는지 하차를 하는지에 따라서 case로 나눈다.	pass
PTS.UTC.2200	승객이 승차를 했을 때 지불해야 할 요금계산 후 요금의 충분 부족에 따라 각 저장/출력 명령을 출력한다..	pass
PTS.UTC.2300	승객이 하차를 했을 때 지불해야 할 요금계산 후 저장/출력 명령을 출력한다.	pass
PTS.UTC.2400	저장에 대한 명령어를 받아서 각각 지하철과 버스 단말기 정보 저장소 파일에 요금정보를 포함해서 저장을 하고, 승객의 카드정보에도 잔액 정보를 포함해서 저장을 한다.	pass
PTS.UTC.2500	출력에 대한 명령어를 받아서 요금과 현재 시간에 대한 정보를 단말기(여기선 터미	pass

	널)에 출력한다.	
PTS.UTC.1610	유저가 태그를 했을 시 유저의 입력을 받은 부분을 모아서 컨트롤 부에서 이용 할 수 있는 자료구조로 따로 저장해준다. 또한 태그 한 시간을 따로 저장해 준다. 그리고 3분마다 초기화 판단 및 시행을 해준다.	pass
PTS.UTC.1620	3분주기마다 시행이 되며 유저카드 이외의 정보를 초기화하는 command를 출력한다.	pass
PTS.UTC.2110	승객의 승차, 하차를 구분 짓는 컨트롤로 사용자 정보와 현재시간을 입력 받아 승차인지 하차인지 결정 짓는다.	pass
PTS.UTC.2210	실질적으로 요금의 경우를 나누어 주는 주 컨트롤러로, 사용자의 정보를 입력 받고, 환승 여부, 미정산 여부, 최초 탑승여부의 조합에 따라 경우를 나누어 각 계산해야 되는 요금식을 판단해주고 다음 프로세스에 트리거를 전송해준다.	pass
PTS.UTC.2220	사용자가 이전에 정산을 하고 환승을 하였을 경우의 금액을 계산하고, 최대금액을 계산한 후 해당 금액들을 다음 컨트롤러로 전달해준다.	pass
PTS.UTC.2230	사용자가 미정산을 하고 환승을 안 한 경우의 금액을 계산하고, 최대금액을 게산한 후 해당 금액들을 다음 컨트롤러로 전달해준다.	pass
PTS.UTC.2240	사용자가 정산을 하고 환승을 안한 경우의 금액을 계산하고, 최대금액을 계산한 후 해당 금액들을 다음 컨트롤러로 전달해준다.	pass
PTS.UTC.2250	사용자가 최초탑승이거나 이전에 버스를 탔거나 환승시간이 넘은 경우, 기본금액을 다음 컨트롤러로 전달해준다.	pass
PTS.UTC.2260	전달받은 금액을 유저의 잔액과 비교해서 요금부족 여부를 판단하여 각각 트리거를 전달해준다.	pass
PTS.UTC.2270	요금정보와 그에 해당하는 정보를 사용자 카드와 단말기에 각각 저장시키고, 금액과 시간을 출력하는 명령을 전송한다.	pass
PTS.UTC.2280	금액이 부족한 경우 활성화 되며, 금액 부족 문구를 출력하는 명령을 전송한다.	pass
PTS.UTC.2310	사용자가 하차태그를 하였을 시 환승을 하였는지 안 하였는지의 여부에 대한 트리거	pass

	를 다음 프로세스에 전송해준다.	
PTS.UTC.2320	하차시 미환승 하였을 경우 지금까지 이동한 시간정보를 입력 받아서 해당하는 금액을 계산하고 해당 금액과 해당하는 정보를 사용자 카드와 단말기에서 각각 저장시키고, 금액과 시간을 출력한다.	pass
PTS.UTC.2330	하차시 환승 하였을 경우 지금까지 이동한 시간정보를 입력 받아서 해당하는 금액을 계산하고 해당금액과 해당하는 정보를 사용자 카드와 단말기에 각각 저장시키고, 금액과 시간을 출력한다.	pass

Table 4 Test design Identification-정산

Identifier	Feature	Pass/Fail
PTS.UTC.1000	각 버스와 지하철 단말기에서부터 정보를 입력받아 저장을 해주는 부분이다.	pass
PTS.UTC.2000	단말기 결제정보에서 정산에 필요한 정보를 불러와 정산처리를 한 뒤 정산된 금액을 각 회사에 보내고, 화면에 출력한다.	pass
PTS.UTC.1100	지하철의 데이터를 입력 받아서 정보를 전송해 준다.	pass
PTS.UTC.1200	버스의 데이터를 입력 받아서 정보를 전송해 준다.	pass
PTS.UTC.2100	저장해둔 단말기 결재정보에서 필요한 정보를 불러오고, 정산금액에 대한 연산을 실행한 후, 각 버스.지하철 회사에 전송 명령을 주고, 화면에 출력하는 명령을 준다.	pass
PTS.UTC.2200	출력 명령어를 입력받아서 정산금액을 화면에 직접적으로 출력을 해준다.	pass
PTS.UTC.2300	전송 명령을 받아서 각 버스와 지하철 회사에 정산금액에 대한 정보를 전송해준다 (구현시에는 각 저장)	pass
PTS.UTC.2110	각 환승별로 정산금액을 계산해주고, 해당 정보를 저장하며, 3분마다 전송부를 활성화 시킨다.	pass

PTS.UTC.2120	정산된 금액에 대해서 각각 지하철과 버스회사로 전송 해주고, 정산된 금액을 출력 해준다	pass

Table 5 Test case Identification-버스

Identifier	Input	Pass/Fail Criteria
PTS.UTC.1000	유저가 태그 하였을 시 유저 카드에 저장된 정보를 입력 받아 저장해주는 부분	
PTS.UTC.1000.00	규모가 큰 Unit.	
PTS.UTC.1100	받은 사용자 카드에서 마지막으로 태그 된 시간 정보를 전달한다.	
PTS.UTC1100.00	Interfaces는 Testing 제외	
PTS.UTC.1200	받은 사용자 카드에 저장된 마지막으로 태그 된 교통수단 정보를 전달한다.	
PTS.UTC1200.00	Interfaces는 Testing 제외	
PTS.UTC.1300	받은 사용자 카드에 저장된 마지막으로 태그 된 승.하차 정보를 전달한다.	
PTS.UTC1300.00	Interfaces는 Testing 제외	
PTS.UTC.1400	받은 사용자 카드에 저장된 잔액 정보를 전달한다.	
PTS.UTC1400.00	Interfaces는 Testing 제외	
PTS.UTC.1500	받은 사용자 카드에 저장된 마지막으로 태그 된 단말기 정보를 전달한다.	
PTS.UTC1500.00	Interfaces는 Testing 제외	
PTS.UTC.1600	유저가 태그를 했을 시 유저의 입력을 받은 부분을 모아서 컨트롤 부에서 이용할 수 있는 자료구조로 따로 저장해준다. 또한 태그를 한 시간을 따로 저장	

	해준다.	
PTS.UTC1600.00	LastTagTime transportation in_out balance terminalinfo	userInfo에 저장, currentInfo에 저장.
PTS.UTC.2000	Tag 정보에서 입력을 받아 정보처리를 한 후 각각 화면에 출력하거나 저장을 한다.	
PTS.UTC.2000.00	규모가 큰 Unit. 단순한 Spec으로 인해 Testing 불가	
PTS.UTC.2100	승객이 승차를 하는지 하차를 하는지에 따라서 case로 나눈다.	
PTS.UTC.2100.00	규모가 큰 Unit. 단순한 Spec으로 인해 Testing 불가	
PTS.UTC.2200	승객이 승차를 했을 때 지불해야 할 요금계산 후 요금의 충분 부족에 따라 각 저장/출력 명령을 출력한다.	
PTS.UTC.2200.00	State ==”case 승차”	저장 command ==fputa 출력 command==printf
PTS.UTC.2300	승객이 하차를 했을 때 지불해야 할 요금계산 후 저장/출력 명령을 출력한다.	
PTS.UTC.2300.00	State == ”Case 하차”	저장 command ==fputa 출력 command ==printf
PTS.UTC.2400	저장에 대한 명령어를 받아서 각각 지하철과 버스 단말기 정보 저장소 파일에 요금정보를 포함해서 저장을 하고, 승객의 카드정보에도 잔액 정보를 포함해서 저장을 한다.	
PTS.UTC.2400.00	Interface는 Testing 제외	

PTS.UTC.2500	출력에 대한 명령어를 받아서 요금과 현재 시간에 대한 정보를 단말기(여기선 터미널)에 출력한다.	
PTS.UTC.2500.00	Interface는 Testing 제외	
PTS.UTC.1610	유저가 태그를 했을 시 유저의 입력을 받은 부분을 모아서 컨트롤 부에서 이용 할 수 있는 자료구조로 따로 저장해준다. 또한 태그 한 시간을 따로 저장해 준다. 그리고 3분마다 초기화 판단 및 시행을 해준다.	
PTS.UTC.1610.00	LastTagTime == ??? Transportation ==??? in_out == ??? balance == ??? terminalinfo == ??? c.time != 03:00	struct userInfo struct currentInfo secTime
PTS.UTC.1610.01	LastTagTime == ??? Transportation == ??? in_out == ??? balance == ??? terminalinfo == ??? c.time == 03:00	Trigger"초기화"
PTS.UTC.1620	3분주기마다 시행이 되며 유저카드 이외의 정보를 초기화하는 command를	

	출력한다.	
PTS.UTC.1620.00	Trigger	"초기화!" 출력
PTS.UTC.2110	승객의 승차, 하차를 구분 짓는 컨트롤로 사용자 정보와 현재시간을 입력 받아 승차인지 하차인지 결정 짓는다.	
PTS.UTC.2110.00	Currentinfo. In_out == 0	Trigger"승차Control"
PTS.UTC.2110.01	Currentinfo. In_out == 1	Trigger"하차Control"
PTS.UTC.2210	실질적으로 요금의 경우를 나누어 주는 주 컨트롤러로, 사용자의 정보를 입력받고, 환승 여부, 미정산 여부, 최초 탑승여부의 조합에 따라 경우를 나누어 각 계산해야 되는 요금식을 판단해주고 다음 프로세스에 트리거를 전송해준다.	
PTS.UTC.2210.00	Trigger "승차Control" Currentinfo. In_out == 0 Userinfo.In_Out == 1 Userinfo.transportation ==1 Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime < 15	Trigger"정산완료&환승"
PTS.UTC.2210.01	Trigger "승차Control" Currentinfo. In_out == 0 Userinfo.In_Out == 0 Userinfo.transportation ==1 Userinfo.terminalinfo[0] =='a'	Trigger"미정산&환승"
PTS.UTC.2210.02	Trigger "승차Control" Currentinfo. In_out == 0 Userinfo.In_Out == 0 Userinfo.transportation ==0	Trigger"미정산&환승"

	Userinfo.terminalinfo[0] != 'a'	
PTS.UTC.2210.03	Trigger "승차Control" Currentinfo. In_out == 0 Userinfo.In_Out == 0 Userinfo.transportation ==1 Userinfo.terminalinfo[0] != 'a'	Trigger"미정산&환승"
PTS.UTC.2210.04	Trigger "승차Control" Currentinfo. In_out == 0 Userinfo.In_Out == 1 Userinfo.transportation ==0	Enable"기본 요금"
PTS.UTC.2210.05	Trigger "승차Control" Currentinfo. In_out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] == 'a'	Enable"기본 요금"
PTS.UTC.2210.06	Trigger "승차Control" Currentinfo. In_out == 0 Userinfo.In_Out == 1 Userinfo.transportation ==1 Userinfo.terminalinfo[0] != 'a' Currentinfo.lasttagtime – userinfo.lasttagtime > 15	Enable"기본 요금"
PTS.UTC.2220	사용자가 이전에 정산을 하고 환승을 하였을 경우의 금액을 계산하고, 최대금액을 계산한 후 해당 금액들을 다음 컨트롤러로 전달해준다.	
PTS.UTC.2220.00	Trigger"정산완료&환승" Currentinfo. In_out == 0 Userinfo.In_Out == 1	Currentinfo.balance == 0

	Userinfo.transportation ==1 Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime < 15	
PTS.UTC.2230	사용자가 최초탑승이거나 이전에 버스를 탔거나 환승시간이 넘은 경우, 기본 금액을 다음 컨트롤러로 전달해 준다.	
PTS.UTC.2230.01	Enable"기본 요금" Currentinfo. In_out == 0 Userinfo.In_Out == 1 Userinfo.transportation ==0	Currentinfo.balance == 1050
PTS.UTC.2230.02	Enable "기본 요금" Currentinfo. In_out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] == 'a'	Currentinfo.balance == 1050
PTS.UTC.2230.03	Enable"기본 요금" Currentinfo. In_out == 0 Userinfo.In_Out == 1 Userinfo.transportation ==1 Userinfo.terminalinfo[0] != 'a' Currentinfo.lasttagtime – userinfo.lasttagtime > 15	Currentinfo.balance == 1050
PTS.UTC.2240	사용자가 미정산을 하고 환승을 안한 경우의 금액을 계산하고, 최대금액을 계산한 후 해당 금액들을 다음 컨트롤러로 전달해준다.	
PTS.UTC.2240.00	Trigger"미정산&환승" Currentinfo. In_out == 0 Userinfo.In_Out == 0 Userinfo.transportation ==1	Currentinfo.balance == 1650

	Userinfo.terminalinfo[0] ==’a’	
PTS.UTC.2240.01	Trigger”미정산&환승” Currentinfo. In_out == 0 Userinfo.In_Out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] != ’a’	Currentinfo.balance == 1650
PTS.UTC.2240.02	Trigger”미정산&환승” Currentinfo. In_out == 0 Userinfo.In_Out == 0 Userinfo.transportation ==1 Userinfo.terminalinfo[0] != ’a’	Currentinfo.balance == 1250
PTS.UTC.2250	전달받은 금액을 유저의 잔액과 비교해서 요금부족 여부를 판단하여 각각 트리거를 전달 해준다.	
PTS.UTC.2250.00	Currentinfo.balance == 0 Userinfo.balance == ???	Userinfo.balance>=500 Trigger”충분”
PTS.UTC.2250.01	Currentinfo.balance == 0 Userinfo.balance == ???	Userinfo.balance<500 Enable”부족”
PTS.UTC.2250.02	Currentinfo.balance == 1050 Userinfo.balance == ???	Userinfo.balance>=1050 Trigger”충분”
PTS.UTC.2250.03	Currentinfo.balance == 1050 Userinfo.balance == ???	Userinfo.balance<1050 Enable”부족”
PTS.UTC.2250.04	Currentinfo.balance == 1250 Userinfo.balance == ???	Userinfo.balance>=1250 Trigger”충분”
PTS.UTC.2250.05	Currentinfo.balance == 1250 Userinfo.balance == ???	Userinfo.balance<1250 Enable”부족”

PTS.UTC.2250.06	Currentinfo.balance == 1650 Userinfo.balance == ???	Userinfo.balance>=1650 Trigger"충분"
PTS.UTC.2250.07	Currentinfo.balance == 1650 Userinfo.balance == ???	Userinfo.balance<1650 Enable"부족"
PTS.UTC.2250.08	Currentinfo.balance == 1750 Userinfo.balance == ???	Userinfo.balance>=1750 Trigger"충분"
PTS.UTC.2250.09	Currentinfo.balance == 1750 Userinfo.balance == ???	Userinfo.balance<1750 Enable"부족"
PTS.UTC.2260	요금정보와 그에 해당하는 정보를 사용자 카드와 단말기에 각각 저장시키고, 금액과 시간을 출력하는 명령을 전송한다.	
PTS.UTC.2260.00	Trigger,"충분"	저장 Command == fputa 출력 Command == printf
PTS.UTC.2270	최종처리에 금액이 부족한 경우 활성화 되며, 금액 부족 문구를 출력하는 명령을 전송한다.	
PTS.UTC2270.00	Enable. "부족"	출력 Command == pritntf
PTS.UTC.2310	사용자가 하차태그를 하였을 시 환승을 하였는지 안 하였는지의 여부에 대한 트리거를 다음 프로세스에 전송해준다.	
PTS.UTC.2310.00	Trigger "하차 Control" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] =='a'	Trigger"미환승 요금계산"
PTS.UTC.2310.01	Trigger "하차Control" Currentinfo. In_out == 1 Userinfo.In_Out == 0	Trigger"환승 요금 계산"

	Userinfo.transportation ==0 Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime < 30	
PTS.UTC.2310.02	Trigger "하차Control" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime <60	Trigger"환승 요금 계산"
PTS.UTC.2310.03	Trigger "하차Control" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime < 90	Trigger"환승 요금 계산"
PTS.UTC.2310.04	Trigger "하차Control" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime < 120	Trigger"환승 요금 계산"
PTS.UTC.2310.05	Trigger "하차Control" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==0	Trigger"환승 요금 계산"

	Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime < 150	
PTS.UTC.2310.06	Trigger "승차Control" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime < 180	Trigger"환승 요금 계산"
PTS.UTC.2320	하차시 미환승 하였을 경우 지금까지 이동한 시간정보를 입력 받아서 해당하는 금액을 계산하고 해당 금액과 해당하는 정보를 사용자 카드와 단말기에서 각각 저장시키고, 금액과 시간을 출력한다.	
PTS.UTC2320.00	Trigger,"미환승 요금계산" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] =='a' c.time	Currentinfo.balance == 0 저장 Command == fputa 출력 Command == printf
PTS.UTC.2330	24차시환 승 하였을 경우 지금까지 이동한 시간정보를 입력 받아서 해당하는 금액을 계산하고 해당금액과 해당하는 정보를 사용자 카드와 단말기에 각각 저장시키고, 금액과 시간을 출력한다.	
PTS.UTC.2330.00	Trigger"환승 요금 계산" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] !='a'	Currentinfo.balance == 0 저장 Command == fputa 출력 Command == printf

	Currentinfo.lasttagtime – userinfo.lasttagtime < 30	
PTS.UTC.2330.01	Trigger"환승 요금 계산" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime <60 c.time	Currentinfo.balance ==100 저장 Command == fputa 출력 Command == printf
PTS.UTC.2330.02	Trigger"환승 요금 계산" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime < 90 c.time	Currentinfo.balance == 200 저장 Command == fputa 출력 Command == printf
PTS.UTC.2330.03	Trigger"환승 요금 계산" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime < 120 c.time	Currentinfo.balance == 300 저장 Command == fputa 출력 Command == printf
PTS.UTC.2330.04	Trigger"환승 요금 계산" Currentinfo. In_out == 1 Userinfo.In_Out == 0	Currentinfo.balance == 400 저장 Command == fputa 출력 Command == printf

	Userinfo.transportation ==0 Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime < 150 c.time	
PTS.UTC.2330.05	Trigger"환승 요금 계산" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==0 Userinfo.terminalinfo[0] !='a' Currentinfo.lasttagtime – userinfo.lasttagtime < 180 c.time	Currentinfo.balance == 500 저장 Command == fputa 출력 Command == printf

Table 6 Test case Identification-지하철

Identifier	Input	Pass/Fail Criteria
PTS.UTC.1000	유저가 태그 하였을 시 유저 카드에 저장된 정보를 입력 받아 저장해주는 부분	
PTS.UTC.1000.00	규모가 큰 Unit.	

PTS.UTC.1100	받은 사용자 카드에서 마지막으로 태그 된 시간 정보를 전달한다.	
PTS.UTC1100.00	Interfaces는 Testing 제외	
PTS.UTC.1200	받은 사용자 카드에 저장된 마지막으로 태그 된 교통수단 정보를 전달한다.	
PTS.UTC1200.00	Interfaces는 Testing 제외	
PTS.UTC.1300	받은 사용자 카드에 저장된 마지막으로 태그 된 승.하차 정보를 전달한다.	
PTS.UTC1300.00	Interfaces는 Testing 제외	
PTS.UTC.1400	받은 사용자 카드에 저장된 잔액 정보를 전달한다.	
PTS.UTC1400.00	Interfaces는 Testing 제외	
PTS.UTC.1500	받은 사용자 카드에 저장된 마지막으로 태그 된 단말기 정보를 전달한다.	
PTS.UTC1500.00	Interfaces는 Testing 제외	
PTS.UTC.1600	유저가 태그를 했을 시 유저의 입력을 받은 부분을 모아서 컨트롤 부에서 이용할 수 있는 자료구조로 따로 저장해준다. 또한 태그를 한 시간을 따로 저장해준다.	
PTS.UTC1600.00	LastTagTime transportation in_out balance terminalinfo	userInfo에 저장, currentInfo에 저장.
PTS.UTC.2000	Tag 정보에서 입력을 받아 정보처리를 한 후 각각 화면에 출력하거나 저장을 한다.	
PTS.UTC.2000.00	규모가 큰 Unit. 단순한 Spec으로 인해 Testing 불가	
PTS.UTC.2100	승객이 승차를 하는지 하차를 하는지에 따라서 case로 나눈다.	

PTS.UTC.2100.00	규모가 큰 Unit. 단순한 Spec으로 인해 Testing 불가	
PTS.UTC.2200	승객이 승차를 했을 때 지불해야 할 요금계산 후 요금의 충분 부족에 따라 각 저장/출력 명령을 출력한다.	
PTS.UTC.2200.00	State ==“case 승차”	저장 command ==fputa 출력 command==printf
PTS.UTC.2300	승객이 하차를 했을 때 지불해야 할 요금계산 후 저장/출력 명령을 출력한다.	
PTS.UTC.2300.00	State == “Case 하차”	저장 command ==fputa 출력 command ==printf
PTS.UTC.2400	저장에 대한 명령어를 받아서 각각 지하철과 버스 단말기 정보 저장소 파일에 요금정보를 포함해서 저장을 하고, 승객의 카드정보에도 잔액 정보를 포함해서 저장을 한다.	
PTS.UTC.2400.00	Interface는 Testing 제외	
PTS.UTC.2500	출력에 대한 명령어를 받아서 요금과 현재 시간에 대한 정보를 단말기(여기선 터미널)에 출력한다.	
PTS.UTC.2500.00	Interface는 Testing 제외	
PTS.UTC.1610	유저가 태그를 했을 시 유저의 입력을 받은 부분을 모아서 컨트롤 부에서 이용 할 수 있는 자료구조로 따로 저장해준다. 또한 태그 한 시간을 따로 저장해 준다. 그리고 3분마다 초기화 판단 및 시행을 해준다.	
PTS.UTC.1610.00	LastTagTime == ??? Transportation ==??? in_out == ??? balance == ???	struct userInfo struct currentInfo secTime

	terminalinfo == ??? c.time != 03:00	
PTS.UTC.1610.01	LastTagTime == ??? Transportation == ??? in_out == ??? balance == ??? terminalinfo == ??? c.time == 03:00	Trigger"초기화"
PTS.UTC.1620	3분주기마다 시행이 되며 유저카드 이외의 정보를 초기화하는 command를 출력한다.	
PTS.UTC.1620.00	Trigger	"초기화!" 출력
PTS.UTC.2110	승객의 승차, 하차를 구분 짓는 컨트롤로 사용자 정보와 현재시간을 입력 받아 승차인지 하차인지 결정 짓는다.	
PTS.UTC.2110.00	Currentinfo. In_out == 0	Trigger"승차Control"
PTS.UTC.2110.01	Currentinfo. In_out == 1	Trigger"하차Control"
PTS.UTC.2210	실질적으로 요금의 경우를 나누어 주는 주 컨트롤러로, 사용자의 정보를 입력받고, 환승 여부, 미정산 여부, 최초 탑승여부의 조합에 따라 경우를 나누어 각 계산해야 되는 요금식을 판단해주고 다음 프로세스에 트리거를 전송해준다.	
PTS.UTC.2210.00	Trigger,"승차Control" Userinfo.In_Out == 0	Trigger"미정산환승"

	Currentinfo. In_out == 0 Userinfo.terminalinfo[0] == 'a' Userinfo.transportation ==1 Currentinfo.terminal[0] != 'a'	
PTS.UTC.2210.01	Trigger,"승차Control" Userinfo.In_Out == 0 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] != 'a' Userinfo.transportation ==0 Currentinfo.terminal[0] != 'a'	Trigger"미정산환승"
PTS.UTC.2210.02	Trigger,"승차Control" Userinfo.In_Out == 0 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] != 'a' Userinfo.transportation ==1 Currentinfo.terminal[0] != 'a'	Trigger"미정산미환승"
PTS.UTC.2210.03	Trigger,"승차Control" Userinfo.In_Out == 0 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] == 'a' Userinfo.transportation ==0 Currentinfo.terminal[0] != 'a'	Trigger"기본요금"
PTS.UTC.2210.04	Trigger,"승차Control" Userinfo.In_Out == 1 Currentinfo. In_out == 0	Trigger"기본요금"

	Userinfo.terminalinfo[0] != 'a' Userinfo.transportation == 1 Currentinfo.terminal[0] != 'a'	
PTS.UTC.2210.05	Trigger,"승차Control" Userinfo.In_Out == 1 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] == 'a' Userinfo.transportation == 1 Currentinfo.terminal[0] != 'a'	Trigger"기본요금"
PTS.UTC.2210.06	Trigger,"승차Control" Userinfo.In_Out == 1 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] == 'a' Userinfo.transportation == 0 Currentinfo.terminal[0] != 'a' Currentinfo.lasttagtime – userinfo.lasttagtime > 15	Trigger"기본요금"
PTS.UTC.2210.07	Trigger,"승차Control" Userinfo.In_Out == 1 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] == 'a' Userinfo.transportation == 0 Currentinfo.terminal[0] != 'a' Currentinfo.lasttagtime – userinfo.lasttagtime <= 15	Trigger"정산환승"
PTS.UTC.2220	사용자가 이전에 정산을 하고 환승을 하였을 경우의 금액을 계산하고, 최대금액을 계산한 후 해당 금액들을 다음 컨트롤러로 전달해준다.	

PTS.UTC.2220.00	Trigger,"미정산환승" Userinfo.In_Out == 0 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] != 'a' Userinfo.transportation ==1 Currentinfo.terminal[0] != 'a'	Currentinfo.balance == 1750
PTS.UTC.2220.01	Trigger,"미정상환승" Userinfo.In_Out == 0 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] != 'a' Userinfo.transportation ==0 Currentinfo.terminal[0] != 'a'	Currentinfo.balance == 1650
PTS.UTC.2230	사용자가 미성잔을 하고 환승을 안 한 경우의 금액을 계산하고, 최대금액을 게산한 후 해당 금액들을 다음 컨트롤러로 전달해준다.	
PTS.UTC.2230.00	Trigger,"미정산미환승" Userinfo.In_Out == 0 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] != 'a' Userinfo.transportation ==1 Currentinfo.terminal[0] != 'a'	Currentinfo.balance == 1250
PTS.UTC.2240	사용자가 정산을 하고 환승을 안한 경우의 금액을 계산하고, 최대금액을 계산한 후 해당 금액들을 다음 컨트롤러로 전달해준다.	
PTS.UTC.2240.00	Trigger"정산환승" Userinfo.In_Out == 1 Currentinfo. In_out == 0	Currentinfo.balance == 0

	Userinfo.terminalinfo[0] == 'a' Userinfo.transportation == 0 Currentinfo.terminal[0] != 'a' Currentinfo.lasttagtime – userinfo.lasttagtime <= 15	
PTS.UTC.2250	사용자가 최초탑승이거나 이전에 버스를 탔거나 환승시간이 넘은 경우, 기본 금액을 다음 컨트롤러로 전달해준다.	
PTS.UTC.2250.00	Enable"기본요금" Userinfo.In_Out == 0 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] == 'a' Userinfo.transportation ==0 Currentinfo.terminal[0] != 'a'	Currentinfo.balance == 1050
PTS.UTC.2250.01	Enable "기본요금" Userinfo.In_Out == 1 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] != 'a' Userinfo.transportation == 1 Currentinfo.terminal[0] != 'a'	Currentinfo.balance == 1050
PTS.UTC.2250.02	Enable"기본요금" Userinfo.In_Out == 1 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] == 'a' Userinfo.transportation == 1 Currentinfo.terminal[0] != 'a'	Currentinfo.balance == 1050
PTS.UTC.2250.03	Enable"기본요금"	Currentinfo.balance == 1050

	Userinfo.In_Out == 1 Currentinfo. In_out == 0 Userinfo.terminalinfo[0] == 'a' Userinfo.transportation == 0 Currentinfo.terminal[0] != 'a' Currentinfo.lasttagtime – userinfo.lasttagtime > 15	
PTS.UTC.2260	전달받은 금액을 유저의 잔액과 비교해서 요금부족 여부를 판단하여 각각 트리거를 전달해준다.	
PTS.UTC.2260.00	Currentinfo.balance == 0 Userinfo.balance == ???	Money = Currentinfo.balance + 600 Userinfo.balance<Money Enable,"부족",
PTS.UTC.2260.01	Currentinfo.balance == 0 Userinfo.balance == ???	Money = Currentinfo.balance + 600 Userinfo.balance>=Money Trigger,"충분",
PTS.UTC.2260.02	Currentinfo.balance != 0 Userinfo.balance == ???	Money = Currentinfo.balance + 200 Userinfo.balance>=Money Trigger,"충분",
PTS.UTC.2260.03	Currentinfo.balance == 1050 Userinfo.balance == ???	Money = Currentinfo.balance + 200 Userinfo.balance<Money Enable,"부족",
PTS.UTC.2270	요금정보와 그에 해당하는 정보를 사용자 카드와 단말기에 각각 저장시키고, 금액과 시간을 출력하는 명령을 전송한다.	
PTS.UTC.2270.00	Trigger,"충분"	저장 Command == fputa 출력 Command == printf
PTS.UTC.2280	금액이 부족한 경우 활성화 되며, 금액 부족 문구를 출력하는 명령을 전송한	

	다.	
PTS.UTC2270.01	Enable. "부족"	출력 Command == pritntf
PTS.UTC.2310	사용자가 하차태그를 하였을 시 환승을 하였는지 안 하였는지의 여부에 대한 트리거를 다음 프로세스에 전송해준다.	
PTS.UTC.2310.00	Trigger "하차 Control" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==1 Userinfo.terminalinfo[0] !='a' Currentinfo.terminalinfo[0] != 'a'	Trigger"미환승 요금계산"
PTS.UTC.2310.01	Trigger "하차 Control" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==1 Userinfo.terminalinfo[0] =='a' Currentinfo.terminalinfo[0] != 'a'	Enable"환승 요금계산"
PTS.UTC.2320	하차시 미환승 하였을 경우 지금까지 이동한 시간정보를 입력 받아서 해당하는 금액을 계산하고 해당 금액과 해당하는 정보를 사용자 카드와 단말기에서 각각 저장시키고, 금액과 시간을 출력한다.	
PTS.UTC.2320.00	Trigger"미환승 요금계산" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==1 Userinfo.terminalinfo[0] !='a' Currentinfo.terminalinfo[0] != 'a'	userinfo.terminalinfo[0] == 'b' && currentinfo.terminalinfo[0] == 'b') || (userinfo.terminalinfo[0] == 'c' && currentinfo.terminalinfo[0] == 'c') || (userinfo.terminalinfo[0] == 'd' && currentinfo.terminalinfo[0] == 'd')

	c.time	|| (userinfo.terminalinfo[0] == 'e' && currentinfo.terminalinfo[0] == 'e') || (userinfo.terminalinfo[0] == 'f' && currentinfo.terminalinfo[0] == 'f' Currentinfo.balance == 0 저장 Command == fputa 출력 Command == printf
PTS.UTC.2320.01	Trigger"미환승 요금계산" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==1 Userinfo.terminalinfo[0] !='a' Currentinfo.terminalinfo[0] != 'a' c.time	userinfo.terminalinfo[0] == 'b' && (currentinfo.terminalinfo[0] == 'f' || currentinfo.terminalinfo[0] == 'c')) || (userinfo.terminalinfo[0] == 'c' && (currentinfo.terminalinfo[0] == 'b' || currentinfo.terminalinfo[0] == 'd')) || (userinfo.terminalinfo[0] == 'd' && (currentinfo.terminalinfo[0] == 'c' || currentinfo.terminalinfo[0] == 'e')) || (userinfo.terminalinfo[0] == 'e' && (currentinfo.terminalinfo[0] == 'd' || currentinfo.terminalinfo[0] == 'f')) || (userinfo.terminalinfo[0] == 'f' && (currentinfo.terminalinfo[0] == 'e' || currentinfo.terminalinfo[0] == 'b' Currentinfo.balance == 0 저장 Command == fputa

		출력 Command == printf
PTS.UTC.2320.02	Trigger"미환승 요금계산" Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==1 Userinfo.terminalinfo[0] !='a' Currentinfo.terminalinfo[0] != 'a' c.time	userinfo.terminalinfo[0] == 'b' && (currentinfo.terminalinfo[0] == 'd' || currentinfo.terminalinfo[0] == 'e')) || (userinfo.terminalinfo[0] == 'c' && (currentinfo.terminalinfo[0] == 'e' || currentinfo.terminalinfo[0] == 'f')) || (userinfo.terminalinfo[0] == 'd' && (currentinfo.terminalinfo[0] == 'f' || currentinfo.terminalinfo[0] == 'b')) || (userinfo.terminalinfo[0] == 'e' && (currentinfo.terminalinfo[0] == 'b' || currentinfo.terminalinfo[0] == 'c')) || (userinfo.terminalinfo[0] == 'f' && (currentinfo.terminalinfo[0] == 'c' || currentinfo.terminalinfo[0] == 'd') Currentinfo.balance == 200 저장 Command == fputa 출력 Command == printf
PTS.UTC.2330	하차시 환승 하였을 경우 지금까지 이동한 시간정보를 입력 받아서 해당하는 금액을 계산하고 해당금액과 해당하는 정보를 사용자 카드와 단말기에 각각 저장시키고, 금액과 시간을 출력한다.	
PTS.UTC.2330.00	Enable,"환승 요금계산" Currentinfo. In_out == 1	(c == 'b' && currentinfo.terminalinfo[0] == 'b') || (c == 'c' && currentinfo.terminalinfo[0] == 'c')

	Userinfo.In_Out == 0 Userinfo.transportation ==1 Userinfo.terminalinfo[0] ==’a’ Currentinfo.terminalinfo[0] != ’a’ c.time	|| (c == 'd' && currentinfo.terminalinfo[0] == 'd') || (c == 'e' && currentinfo.terminalinfo[0] == 'e') || (c == 'f' && currentinfo.terminalinfo[0] == 'f' Currentinfo.balance == 0 저장 Command == fputa 출력 Command == printf
PTS.UTC.2330.01	Enable,”환승 요금계산” Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==1 Userinfo.terminalinfo[0] ==’a’ Currentinfo.terminalinfo[0] != ’a’ c.time	(c == 'b' && (currentinfo.terminalinfo[0] == 'c' || currentinfo.terminalinfo[0] == 'f')) || (c == 'c' && (currentinfo.terminalinfo[0] == 'd' || currentinfo.terminalinfo[0] == 'b')) || (c == 'd' && (currentinfo.terminalinfo[0] == 'e' || currentinfo.terminalinfo[0] == 'c')) || (c == 'e' && (currentinfo.terminalinfo[0] == 'f' || currentinfo.terminalinfo[0] == 'd')) || (c == 'f' && (currentinfo.terminalinfo[0] == 'b' || currentinfo.terminalinfo[0] == 'e' Currentinfo.balance == 300 저장 Command == fputa 출력 Command == printf
PTS.UTC.2330.02	Enable,”환승 요금계산” Currentinfo. In_out == 1 Userinfo.In_Out == 0 Userinfo.transportation ==1	c == 'b' && (currentinfo.terminalinfo[0] == 'd' || currentinfo.terminalinfo[0] == 'e')) || (c == 'c' && (currentinfo.terminalinfo[0] == 'e' || currentinfo.terminalinfo[0] == 'f'))

	Userinfo.terminalinfo[0] ==’a’ Currentinfo.terminalinfo[0] != ’a’ c.time	|| (c == 'd' && (currentinfo.terminalinfo[0] == 'f' || currentinfo.terminalinfo[0] == 'b')) || (c == 'e' && (currentinfo.terminalinfo[0] == 'b' || currentinfo.terminalinfo[0] == 'c')) || (c == 'f' && (currentinfo.terminalinfo[0] == 'c' || currentinfo.terminalinfo[0] == 'd' Currentinfo.balance == 600 저장 Command == fputa 출력 Command == printf

Table 7 Test case Identification-정산

Identifier	Input	Pass/Fail Criteria	비고
PTS.UTC.1000	각 버스와 지하철 단말기에서부터 정보를 입력받아 저장을 해주는 부분이다.		
PTS.UTC.1000	LastTagTime transportation in_out balance terminalinfo	termInfo에 저장,	

PTS.UTC.2000	단말기 결제정보에서 정산에 필요한 정보를 불러와 정산처리를 한 뒤 정산된 금액을 각 회사에 보내고, 화면에 출력한다.		

PTS.UTC.2000.00	Load(단말기 결제 정보)
	getInfo(struct result* rr);
	sortBal(struct termInfo* temp);
		Transmit
		print	

PTS.UTC.1100	지하철의 데이터를 입력 받아서 정보를 전송해 준다.		
PTS.UTC1100.00	LastTagTime transportation in_out balance terminalinfo	termInfo에 저장,	
PTS.UTC.1200	버스의 데이터를 입력 받아서 정보를 전송해 준다.		
PTS.UTC1200.00	LastTagTime transportation in_out balance terminalinfo	termInfo에 저장,	
PTS.UTC.2100	저장해둔 단말기 결재정보에서 필요한 정보를 불러오고, 정산금액에 대한 연산을 실행한 후, 각 버스.지하철 회사에 전승 명령을 주고, 화면에 출력하는 명령을 준다.		

PTS.UTC2100.00	Load(단말기 결제 정보), tick
	getInfo(struct result* rr);
	sortBal(struct termInfo* temp);
		Case 출력
		Case 전송	

PTS.UTC.2200	출력 명령어를 입력받아서 정산금액을 화면에 직접적으로 출력을 해준다.		
PTS.UTC2200.00	Case 출력	print	
PTS.UTC.2300	전송 명령을 받아서 각 버스와 지하철 회사에 정산금액에 대한 정보를 전송해준다(구현시에는 각 저장)		
PTS.UTC2300.00	Case 전송	Subway corporation transmit, bus corporation transmit	
PTS.UTC.2110	각 환승 방향별로 정산금액을 계산해주고, 해당 정보를 저장하며, 3분마다 전송부를 활성화 시킨다.		

PTS.UTC2110.00	Load tick
	getInfo(struct result* rr);
	sortBal(struct termInfo* temp);
		Enable/Disable"정산실행"	

PTS.UTC.2120	정산된 금액에 대해서 각각 지하철과 버스회사로 전송해주고, 정산된 금액을 출력해준다		
PTS.UTC2120.00	Enable, Disable adjustment(struct result* rt);	Save, Print	