Requirements Based Test Cases for the

System for Uniform Route-Based

Transportation Simulation (SURTS)

11.09.2020

Group A41
CSCI 5801

Document Revision History

Rev	Date	Author	Change Description
0.2	**10.31.2020**	David Ibarra	Initial Requirements Specification document creation.
0.3	**11.05.2020**	David Ibarra	Test Cases for Reports.
1.0	11.09.2020	David Ibarra et al	Completed Test Cases

This document is originating from Neil Bitzenhofer and DataCard Corporation.

Table of Contents

1. Test Requirements

1.1 Objective

The purpose of the Test Requirements section is to list ALL hardware and software test requirements, whether explicitly determined from any relevant documents or implicitly determined from experience and product knowledge. For most projects, the documents referred to may be the Product Definition Document, Software/Hardware Requirements Specification and perhaps the Software/Hardware Design Specification. A Test Case Matrix is provided that simply lists all the test cases by title or description, and includes a method of tracking when the test case was run and whether it passed or not.

1.2 Definitions and Acronyms

Here, we list the technical terms or acronyms used in the document.

- SRS Software Requirements Specification
- TM Traceability Matrix
- SURTS System for Uniform Route-Based Transportation Simulation 1

1.3 Traceability Matrix

The purpose of a Traceability Matrix is to show which test cases verify which requirements. A possible format for a Traceability Matrix is as follows:

Requirements / TestCase	TestCaseE521	TestCaseE522	TestCaseE523	TestCaseE524	TestCaseE525	TestCaseE526	TestCaseE531	TestCaseE532	TestCaseE533	TestCaseE534	TestCaseE535	TestCaseE536	TestCaseE541	TestCaseE542	TestCaseE543	TestCaseE544	TestCaseE545	TestCaseE546	TestCaseE547	TestCaseE551	TestCaseE552	TestCaseE553	TestCaseE554
Req 1	X	X																					
Req 2							X																
Req 3			X	X																			
Req 4					X	X																	

Req 5							X																
Req 6								X															
Req 7									X														
Req 10										X													
Req 11											X												
Req 12												X											
Req 13													X	X									
Req 14													X		X								
Req 15																X	X						
Req 16																		X	X				
Req 17																				X	X		
Req 18																					X	X	X
Req 19																							X

2. Test Cases

E.5.2.1 Valid call to create Bus

Description: Ensures that a bus is created if given valid inputs

Test Inputs:
1. A bus type *T*
2. A bus route *R*

Expected Results: A bus is created entailing the given inputs, and stored in the system.

Dependencies: Depends on successful route creation, simulation run tests

Initialization: The system should be in a state to accept bus creation.

Test Steps:
1. Verify that the test inputs are within design constraints
2. A bus creation instruction is given.
3. Verify that a bus is created with the type *T* and route *R*

Owner: Jackson McEligot

E.5.2.2 Non-valid call to create Bus

Description: Ensures that the system handles for invalid bus inputs

Test Inputs: None

Expected Results: The system indicates that a bus was not generated due to an impossible system quality

Dependencies: None

Initialization: None

Test Steps:
1. A bus creation instruction is given with the parameter NULL for type *T* and route *R*
2. Verify that no bus is generated, nor shown to the user.
3. Verify that an error indicating that bus type *T* is not valid nor is route *R*

Owner: Jackson McEligot

E.5.2.3 Valid call to create Bus type

Description: Ensures that a bus is created if given valid inputs

Test Inputs:
1. Max capacity *C* where 0 <= *C* <= 10000
2. Length of bus *L* where 0 <= *L* <= 1000
3. Fuel consumption *F* where 0 <= *F* <= 1000
4. Bus type name *N* where length of *N* < 1000 characters

Expected Results: A bus type is created entailing the given inputs, and stored in the system.

Dependencies: None

Initialization: The system should already set up

Test Steps:
1. Verify that the test inputs are within design constraints
2. A bus type creation instruction is given.
3. Verify that a bus type is created with the max capacity *C*, length *L*, fuel consumption *F*, and name *N*

Owner: Jackson McEligot

E.5.2.4 Non-valid call to create Bus type

Description: Ensures that the system handles for invalid bus inputs

Test Inputs: None

Expected Results: The system indicates that a bus was not generated due to an impossible system quality

Dependencies: None

Initialization: None

Test Steps:
1. A bus creation instruction is given with the parameter NULL for max capacity *C*, length *L*, fuel consumption *F*, and name *N*
2. Verify that no bus type is created or stored, nor shown to the user.
3. Verify that an error indicating that the input parameters are NULL is shown to the user

Owner: Jackson McEligot

E.5.2.5 Valid call to Delete Buses

Description: Ensures that the system can delete buses correctly.

Test Inputs: An previously created bus in the simulations bus repository.

Expected Results: The system will indicate that the bus was deleted corectly

Dependencies: None

Initialization: Create a bus and add it to the repository

Test Steps:
1. Create a bus
2. Indicate to delete that bus
3. The system should call delete bus and delete it
4. The user should see the bus disappear from the repository

Owner: Connor Boehm

E.5.2.6 In-valid call to Delete Buses

Description: Ensures that the system will not function if there are no buses in the repository.

Test Inputs: An previously created bus in the simulations bus repository.

Expected Results: The system will indicate that it was unable to delete as there is no buses selected taht are in the repository

Dependencies: None

Initialization: None

Test Steps:
1.
2. Indicate to delete that bus
3. The system should call delete bus and fail
4. The user should see some sort of error message

Owner: Connor Boehm

E.5.3.1 Create Passengers

Description: The system should allow for passengers to be created.

Test Inputs: User selected passenger parameters.

Expected Results: Passengers created and added to the simulation.

Dependencies: N/A

Initialization: The user decides to create a passenger and is presented with a passenger creation menu.

Test Steps:
1. Prompt user with passenger creation fields.
2. Verify that all required fields are filled.
3. Verify that all values entered are within an allowable range.
4. Notify the user that the passenger has been created.
5. Add the passenger to the current simulation.

Owner: Austin Wessel

E.5.3.2 Historical Passenger Data

Description: The system should allow for the upload of historically collected passenger data.

Test Inputs: Files containing historical passenger data.

Expected Results: Historical passenger data added to the system and passenger patterns created from it.

Dependencies: N/A

Initialization: The user uploads a file containing historical passenger data.

Test Steps:
1. Import the file into the systems scope.
2. Verify that the file is compatible with our system.
3. Verify that all values in file are within the acceptable range.
4. Notify the user that the file has been successfully imported.
5. Create a new passenger pattern using the information within the imported file.

Owner: Austin Wessel

E.5.3.3 Create Passenger Pattern

Description: The system should allow for the creation of a new passenger pattern.

Test Inputs: User specified passenger pattern parameters.

Expected Results: A new passenger pattern will be created and can be added to future simulations.

Dependencies: N/A

Initialization: The user chooses to create a new passenger pattern.

Test Steps:
1. Prompt the user with passenger pattern fields.
2. Verify that all required fields have a value.
3. Verify that all values are within the acceptable range.
4. Notify the user that the passenger pattern has been created.
5. Save the passenger pattern and make it available to future simulations.

Owner: Austin Wessel

E.5.3.4 Delete Passenger Pattern

Description: The system should allow for the deletion of existing passenger patterns.

Test Inputs: User specified passenger pattern.

Expected Results: The passenger pattern will be deleted and the memory will be returned to the system.

Dependencies: N/A

Initialization: The user chooses to delete an existing passenger pattern.

Test Steps:

1. Alert the user that the deleted passenger pattern will no longer be available for simulations.
2. Prompt the user to verify that they understand.
3. Verify that the specified passenger pattern is currently active and eligible for deletion.
4. Delete the passenger pattern and recover the corresponding system memory.
5. Notify the user that the passenger pattern has been successfully deleted.

Owner: Austin Wessel

E.5.3.5 Create/Configure Routes

Description: The system should allow for users to create new routes and modify existing ones.

Test Inputs: User specified route selection(existing) and route parameters.

Expected Results: The newly created/configured route will be available to run in future simulations.

Dependencies: N/A

Initialization: The user chooses to create a new or modify an existing route.

Test Steps:

1. Provide the user with the fields required for route creation.
2. Verify that all required fields contain a value.
3. Verify that all values are within the approved range.
4. Save the new/modified route for use in future simulations.
5. Notify the user that the route was either successfully created or successfully modified.

Owner: Austin Wessel

E.5.3.6 Delete Routes

Description: The system should allow for the deletion of existing routes.

Test Inputs: User specified route.

Expected Results: The route will be deleted and the memory will be returned to the system.

Dependencies: N/A

Initialization: The user chooses to delete an existing route..

Test Steps:
1. Alert the user that the deleted route will no longer be available for simulations.
2. Prompt the user to verify that they understand.
3. Verify that the specified route is currently active and eligible for deletion.
4. Delete the route and recover the corresponding system memory.
5. Notify the user that the route has been successfully deleted.

Owner: Austin Wessel

E.5.4.1 Valid call to create and store a log

Description: Ensures that a log is created if given valid inputs

Test Inputs:
1. A bus with the following valid attributes:
    a. Name: *N*
    b. Route: *R*
    c. Location: *L*
    d. Max capacity: *C*
    e. Passenger count: *P*
2. A positive, reachable simulation time *K*

Expected Results: A log is created entailing the given inputs, and stored in the system.

Dependencies: Depends on successful bus creation, bus route creation, simulation run tests

Initialization: The system should be running a simulation and be at a point where the inputs to the test are true to the simulation.

Test Steps:
1. Verify that the test inputs are true to the system status.
2. A log creation instruction is given.
3. Verify that a log is generated with the following output:
    ~~~Previous simulation log entries~~~
    K,N,C,P,R,L
4. Verify that a log is shown to the user.
5. Verify that the generated log is stored in the system's defined log memory location.

Owner: Jackson McEligot

E.5.4.2 Non-valid call to create log

Description: Ensures that the system handles for invalid log inputs

Test Inputs:
1. A bus with the following attributes:
    a. Name: *N*
    b. Route: *R*
    c. Location: *L*
    d. Max capacity: 300
    e. Passenger count: 301
2. A positive, reachable simulation time *K*

Expected Results: The system indicates that a log was not generated due to an impossible system quality

Dependencies: None

Initialization: None

Test Steps:
1. A log creation instruction is given with the test inputs.
2. Verify that no log is generated, nor shown to the user.
3. Verify that an error indicating that bus *N*'s passenger count exceeds its max capacity is shown to the user

Owner: Jackson McEligot

E.5.4.3 Non-valid call to store a log in the system

Description: Ensures that the system handles for invalid log storage system calls

Test Inputs: A log with the format:
~~~Previous log entries~~~
K,N,C,P,R,L,z
Where *K*, *N*, *C*, *P*, *R*, and *L* are valid log attributes as described in test case 5.4.1, and *z* is a garbage entry not accepted by the system.

Expected Results: No new log is stored in the system, and an error is displayed to the user.

Dependencies: None

Initialization: Successful generation of system log as in test 5.4.1

Test Steps:
1. An instruction is given to the system to store the provided test input log.
2. Verify that the system does not store the log in the system defined memory location.
3. Verify that an error is displayed to the user indicating that no log was stored and the cause of this error was *z*.

Owner: Jackson McEligot

E.5.4.4 Valid call to inspect a log

Description: Ensures that a created log can be viewed

Test Inputs: A previously created log *L,* given by log id *i*.

Expected Results: A log is displayed to the user.

Dependencies: Depends on successful log creation and storage

Initialization: None

Test Steps:
1. An instruction is given to the system to inspect input log *L* from id *i*.
2. Verify that the system indicates it received the instruction.
3. Verify that the system displays *L* to the user.

Owner: Jackson McEligot

E.5.4.5 Non-valid call to inspect a log

Description: Ensures that the system handles for invalid log inspect inputs

Test Inputs: None

Expected Results: The system indicates that a log was not displayed due to a nonexistent log

Dependencies: None

Initialization: None

Test Steps:
1. An instruction is given to the system to inspect a garbage log *J*, from garbage id *g*.
2. Verify that the system indicates it received the instruction.
3. Verify that no log is displayed to the user.
4. Verify that an error is displayed that indicates *J*, *g* do not exist within the system.

Owner: Jackson McEligot

E.5.4.6 Valid call to delete a log

Description: Ensures that a log is deleted if given valid inputs

Test Inputs: A created log *L* with input id *i*.

Expected Results: Input log *L* is deleted from the system.

Dependencies: Depends on successful log creation and storage.

Initialization: The system should have run and generated/stored at minimum one log.

Test Steps:
1. An instruction is given to the system to delete log *L* based on input id *i*.
2. Verify that the system indicates it received the instruction.
3. Verify that the indicated log is removed from the system.
4. Verify that the system informs the user of a successful deletion.

Owner: Jackson McEligot

E.5.4.7 Non-valid call to delete a log

Description: Ensures that no log is deleted if invalid inputs are given in deletion instruction.

Test Inputs: None

Expected Results: No log is deleted from the system, and an error message is displayed indicating such.

Dependencies: None

Initialization: None

Test Steps:
1. An instruction is given to the system to delete garbage log *L* based on garbage id *i*.
2. Verify that the system indicates it received the instruction.
3. Verify that no log is removed from the system.
4. Verify that an error message is displayed indicating that *L*, based off of *i*, does not exist within the system and could not be deleted.

Owner: Jackson McEligot

E.5.5.1 Generate Reports

Description: After a simulation has ended, the system should generate a summary report.

Test Inputs: A simulation that recently was run and completed successfully on the system.

Expected Results: A completed report on the simulation, stored on the system.

Dependencies: This test case depends on the successful execution and completion of the bus simulation.

Initialization: A simulation should have run and completed successfully on the system.

Test Steps:
1. Run a simulation.
2. Verify that after the simulation has finished, and if there is space on the system the report has been generated and is available on the system.
3. Verify that the user is notified with a message that a report has been generated on the system.
4. Verify that the user is able to view the report on the system after the simulation has terminated.
5. Verify that the simulation includes bus utilization information.
6. Verify that the simulation includes wait times for passengers.

Owner: David Ibarra

E.5.5.2 Save Reports

Description: After a report has been generated, a user may want to save the report outside of the system.

Test Inputs: A previously generated report that is on the system.

Expected Results: Completed report on the simulation, stored outside of the SURTS on the file system. Return an error if the save locations are invalid or there is no space available outside of the system.

Dependencies: Thus test case depends on test case E.5.5.1 as a report must be generated successfully.

Initialization: Run a simulation, having a report generated.

Test Steps:
1. Verify that the user is able to select a report to save.
2. Verify that the user is able to select where to save the report.
3. If there is space available on the system, verify that the report has been saved on the system in the selected location.

Owner: David Ibarra

E.5.5.3 Invalid Save of Reports

Description: If the user tries to save the reports to an invalid location, or there is no space available outside of the system, generate an error.

Test Inputs: A previously generated report that is on the system.

Expected Results: Return an error when the save locations are invalid or there is no space available outside of the system.

Dependencies: Thus test case depends on test case E.5.5.1 as a report must be generated successfully.

Initialization: Run a simulation, having a report generated.

Test Steps:
1. Verify that the user is able to select a report to save.
2. Verify that the user is able to select where to save the report.
3. Verify that invalid save paths within the system give an error message, as well as when you try to save the file when no memory is available.

Owner: David Ibarra

E.5.5.4 Delete Reports

Description: Remove a report from the system.

Test Inputs:	A report file that has been created, that is persistent on the system.
Expected Results:	File is removed from the system.
Dependencies:	Thus test case depends on test case E.5.5.1 as a report must be generated successfully.
Initialization:	Have run a simulation, as well as a report that is generated and stored on the system.
Test Steps:	1. Verify that a user can select a previously created report to delete. 2. Verify that the user is able to delete the report from the system, after which the report is no longer available on the system.
Owner:	David Ibarra