CMSC 447

Software Test Description (STD)

Revision 1

1 Scope 3
1.1 Identification 3
1.2 System overview 3
1.3 Document overview 3
2 Referenced documents 3
3 Test preparations 3
3.1 (Project-unique identifier of a test) 3
3.1.1 Hardware preparation 3
3.1.2 Software preparation 4
3.1.3 Other pre-test preparations 4
4 Test descriptions 4
4.1 (Project-unique identifier of a test) 4
4.1.1 (Project-unique identifier of a test case) 4
5 Requirements traceability 6
6 Notes 7
A. Appendixes 8

1 Scope

1.1 Identification

This Software Test Plan (STP) provides aspects for various tests (designed to confirm requirement satisfiability, as described in the SRS) to be run against code units in version 1.0.0. of the CSCI, called the ROBBR system. No other releases of this system have been made, and no new releases are foreseen as of latest document revision.

1.2 System overview

The purpose of the ROBBR system is to allow a user to determine the best location for a new headquarters for some particular jewel thief syndicate. The system collect publicly available statistics about locations within the United States, and the user provides criteria for filtering and ranking houses within a user-selected state. The system then generates a list of houses, weighed by their adherence to the user specified criteria, and displays it to the user. The ROBBR system is being developed for the customer who will be the sole sponsor, acquirer, and user. The ROBBR system will run on a bootable which may be carried by the customer. The I am Root Software Engineering Team shall be the sole developers of the software, and any reference to the developer will refer to the I am Root Software Engineering Team. For privacy and security reasons, the customer is not identified.

1.3 Document overview

The STP describes how the system is tested against the requirements, specifically their satisfiability conditions. A number of tests are cataloged, along with their intended purpose, the requirement they mean to satisfy, and the code unit they are to be run against. The environment in which the tests are to be run is also detailed.

2 Referenced documents

This section shall list the number, title, revision, and date of all documents referenced in this document. This section shall also identify the source for all documents not available through normal Government stocking activities.

Number	Title	Revision	Date
1	Software Development Plan	1	4/19/2018
2	Software Requirement Specification	1	4/19/2018

3	Software Design Document	1	5/9/2018
4	Software User Manual	1	5/13/2018

3 Test preparations
The flash drive to be tested must have a version of linux as well as a local instance of the main ROBBR driver installed on it. No other test preparations are necessary.

4 Test descriptions
4.1 ROBBR_DATA
4.1.1 ROBBR_DATA_FILE_TEST
4.1.1.1 *Prerequisite conditions*
In order to pass the ROBBR_DATA_FILE_TEST successfully, it is required that state and housing data, which are all stored in CSV's, are already preprocessed by scripts.

4.1.1.2 *Test inputs*
Inputs include a list of zip codes and their corresponding state that will be gathered from census data as well as a home directory from which to start the search algorithm.

4.1.1.3 *Expected test results*
It is expected that the return list will have a True value associated with every zip code in the list. That is to say the file system should be evaluated as complete.

4.1.1.4 *Criteria for evaluating results*
The script will return a list of zip codes and boolean values. All values marked false represent files that are missing from their intended or expected file locations and therefore need to be fixed.

4.1.1.5 *Test procedure*
The test script will go through the list of all american zip codes and check the corresponding state file system for its presence. The script will then add the zip code and a boolean value for whether or not it was found in the proper location to a list. If an exception is thrown during execution then the zip code currently being searched for will be marked as not found.

4.1.1.6 *Assumptions and constraints*
This test only finds missing zip codes. It does not account for zip codes being in the wrong folder as that does not affect whether or not the ROBBR system will run. This test also assumes

that the input list of zip codes and their corresponding states is complete and accurate for the continental U.S..

4.1.2 ROBBR_DATA_USB_TEST

4.1.2.1 *Prerequisite conditions*

ROBBR_DATA_USB_TEST requires access to a computer with usb 3.0 capabilities and the final product flashdrive.

4.1.2.2 Test inputs

No test inputs.

4.1.2.3 *Expected test results*

The host computer is expected to successfully boot off of the flash drive.

4.1.2.4 *Criteria for evaluating results*

The test will be considered a success if the start screen for the local version of linux on the flash drive appears instead of the normal start screen for the host machine operation system.

4.1.2.5 *Test procedure*

The tester will attempt to boot into the flash drive following the instruction given in the Software User Manual(SUM).

4.1.2.6 *Assumptions and constraints*

It is assumed that the host machine has a start screen different enough from that of the flash drive's linux so that the tester can easily tell when the test was successful.

4.2 ROBBR_GUI

4.2.1 ROBBR_GUI_INPUT_TEST

4.2.1.1 *Prerequisite conditions*

The ROBBR_GUI_INPUT_TEST will require that the file system is properly set up so that state and housing data are correctly stored. The tester will also require multiple subsets of data (all of which vary in how they are sampled to the tester's discretion) so that they may efficiently test the GUI's ability to take in input from the user. The test will be performed after the ROBBR_DATA tests to ensure that the GUI takes in the intended, accurate data.

4.2.1.2 *Test inputs*

ROBBR_GUI_INPUT_TEST will be carried out in multiple tests to the tester's discretion. The procedure will be same for all tests but the subsets of data will be different. The subsets will be pulled from existing, preprocessed data; data will be real.

4.2.1.3 *Expected test results*

ROBBR_GUI_INPUT_TEST is testing for two criterias: the GUI is able to utilize the values pulled from preprocessed data and that the GUI is able to successfully take in multiple values from the user. The expected result from ROBBR_GUI_INPUT_TEST is that the GUI will display the correct values and successfully store the inputs from the tester so that the results can be processed using those inputs. It follows that there will be no intermediate results and only final results.

4.2.1.4 *Criteria for evaluating results*

The results will be evaluated by checking the GUI and ensuring that the various inputs entered in by the user are correctly stored. The values displayed on the GUI will also be correct. Since taking in input from the user is necessary for ROBBR to function, there will be no margin for error. Otherwise, the ROBBR_GUI_INPUT_TEST will be considered to have failed.

4.2.1.5 *Test procedure*

1. The first step in ROBBR_GUI_INPUT_TEST is to ensure that the values displayed on the sliders and menu match with the dataset being utilized at the time. This is because the sliders represent the range of values used by the dataset. The GUI has multiple sliders and dropdown menus that utilize information from existing, preprocessed data. Thus, it is assumed that the GUI will store multiple values. Should the values not match, the tester should correct this issue.
2. The tester should then interact with the GUI to completion. This will send the multiple values to the backend. Should there be an error, the system should alert the user of a problem. If this step is successfully completed, the tester should ensure the system was able to take in the values entered through the output.

4.2.1.6 *Assumptions and constraints*

This test assumes that the datasets are correct and properly formatted so that the GUI can safely take them in without crashing. This is to ensure that the test can proceed with testing user input.

4.2.2 ROBBR_GUI_OUTPUT_TEST

4.2.2.1 *Prerequisite conditions*

The ROBBR_GUI_OUTPUT_TEST will require that the file system is properly set up so that state and housing data are correctly stored. The tester will also require multiple subsets of data (all of which vary in how they are sampled to the tester's discretion) so that they may efficiently test the GUI's ability to take in input from the user. The test will be performed following the success of ROBBR_GUI_INPUT_TEST.

4.2.2.2 *Test inputs*

ROBBR_GUI_OUTPUT_TEST takes all datasets utilized during ROBBR_GUI_INPUT_TEST along with the values entered in by the tester.

4.2.2.3 *Expected test results*

Test results will be displayed as the output. It follows that there will only be final results.

4.2.2.4 *Criteria for evaluating results*

The results will be evaluated by manually checking the output for accurateness. Accurateness will be determined by cross-referencing with other tools and websites. Should the results be correct, the test will be deemed as a success. Since displaying the output is a core functionality of ROBBR, there will be no margin for error. Otherwise, ROBBR_GUI_OUTPUT_TEST will have failed.

4.2.2.5 *Test procedure*

1. The first step in this test follows the successful task of entering in input through the ROBBR GUI. The tester will need to verify that the output is successfully displayed on the GUI. If not, the tester will need to fix this. The GUI will have displayed a map with pointers indicating possible houses to move into. Below the map will be a list detailing those houses.

2. Should the output be displayed successfully, the tester will verify that the output is accurate. Accurateness can be determined by checking with tools such as Google Maps to see if the houses listed exist and are for sale.

4.2.2.6 *Assumptions and constraints*
The test will be performed after the ROBBR_DATA tests to ensure that the GUI takes in the intended, accurate data.

4.3 ROBBR_PROCESSING

4.3.1 ROBBR_PROCESSING_TEST

4.3.1.1 *Prerequisite conditions*
The data file system is setup properly as mentioned in the ROBBR_DATA_FILE_TEST (4.1.1).

4.3.1.2 *Test inputs*
This test takes as input a set of input values that when combined create a very small list of houses and the output values we expect to be generated by these inputs. A list of expected values for each of the three filtered CSVs as well as the final expected results.

4.3.1.3 *Expected test results*
Expected and actual output house lists and filter lists should match perfectly.

4.3.1.4 *Criteria for evaluating results*
If the expected and actual output lists are exactly the same, then the test was successful. If the end house list is different from expected then each filter list can be checked to see where specifically the error is.

4.3.1.5 *Test procedure*
1.) The test will run predetermined input values through the main ROBBR driver.
2.) The main ROBBR driver naturally creates four csvs, one for each filter the data is sent through and one for the final fully filtered results.
3.) These CSVs can be checked against the expected ones input at the beginning of the test.

4.3.1.6 *Assumptions and constraints*

This test assumes that inputs can be found that create a suitably small output list.

5 Requirements traceability

CSCI Requirement Addressed	**Test Name**
(1) Select Region	ROBBR_GUI_INPUT_TEST
(2) Switch to Filter Criteria Input Mode	ROBBR_GUI_INPUT_TEST
(1) Switch to Select Region Mode	ROBBR_GUI_INPUT_TEST
(2) Input Criteria Selection Sliders	ROBBR_GUI_INPUT_TEST
(2.1) Minimum Household Income Slider	ROBBR_GUI_INPUT_TEST
(2.2) Maximum Household Income Slider	ROBBR_GUI_INPUT_TEST
(2.3) Minimum Distance Slider	ROBBR_GUI_INPUT_TEST
(2.4) Maximum Distance Slider	ROBBR_GUI_INPUT_TEST
(2.5) Average Crime Slider	ROBBR_GUI_INPUT_TEST
(2.6) House Quantity Slider	ROBBR_GUI_INPUT_TEST
(3.0) Criteria Weighting Fields	ROBBR_GUI_INPUT_TEST
(4.0) Switch to Results Output Mode	ROBBR_GUI_INPUT_TEST ROBBR_GUI_OUTPUT_TEST
(1) Switch to Filter Criteria Input Mode	ROBBR_GUI_INPUT_TEST
(2) Generate Results	ROBBR_PROCESSING_TEST
(2.1) Generate Results Quickly	ROBBR_PROCESSING_TEST
(3) Display Results	ROBBR_GUI_OUTPUT_TEST
(3.1) Display Zip Codes as List	ROBBR_GUI_OUTPUT_TEST
(3.2) Display Houses as List	ROBBR_GUI_OUTPUT_TEST
(3.3) Display a Map	ROBBR_GUI_OUTPUT_TEST

(3.3.1) Display Houses on Map	ROBBR_GUI_OUTPUT_TEST
(3.3.2) Display Region on Map	ROBBR_GUI_OUTPUT_TEST
(3.3.3) Display Center of Mass on Map	ROBBR_GUI_OUTPUT_TEST
(1) Results Generation Algorithm Zip Code Statistics	ROBBR_PROCESSING_TEST
(2) Results Generation Algorithm Zip Code House Information	ROBBR_PROCESSING_TEST
(1) Provide Graphical User Interface	ROBBR_GUI_OUTPUT_TEST ROBBR_GUI_INPUT_TEST
(1) Statistics Format Within File Structure	ROBBR_DATA_FILE_TEST
(2) House Format Within File Structure	ROBBR_DATA_FILE_TEST
(1) Flash Drive System	ROBBR_DATA_USB_TEST
(2) No Logging	---
(1) Computer Guest Operating System Boot Capability	ROBBR_DATA_USB_TEST
(2) Computer USB 3.0 Port Availability for Guest OS Flash Drive	---
(3) Computer Internet Connection	---

DESCRIPTION/PURPOSE

The Software Test Description (STD) describes the test preparations, test cases, and test procedures to be used to perform qualification testing of a Computer Software Configuration Item (CSCI) or a software system or subsystem.

The STD enables the acquirer to assess the adequacy of the qualification testing to be performed.

APPLICATION/INTERRELATIONSHIP

Portions of this plan may be bound separately if this approach enhances their usability. Examples include plans for software configuration management and software quality assurance.

The Contract Data Requirements List (CDRL) should specify whether deliverable data are to be delivered on paper or electronic media; are to be in a given electronic form (such as ASCII, CALS, or compatible with a specified word processor or other support software); may be delivered in developer format rather than in the format specified herein; and may reside in a computer-aided software engineering (CASE) or other automated tool rather than in the form of a traditional document.

PREPARATION INSTRUCTIONS

General instructions.

a. Automated techniques. Use of automated techniques is encouraged. The term "document" in this means a collection of data regardless of its medium.

b. Alternate presentation styles. Diagrams, tables, matrices, and other presentation styles are acceptable substitutes for text when data required can be made more readable using these styles.

c. Title page or identifier. The document shall include a title page containing, as applicable: document number; volume number; version/revision indicator; security markings or other restrictions on the handling of the document; date; document title; name, abbreviation, and any other identifier for the system, subsystem, or item to which the document applies; contract number; CDRL item number; organization for which the document has been prepared; name and address of the preparing organization; and distribution statement. For data in a database or other alternative form, this information shall be included on external and internal labels or by equivalent identification methods.

d. Table of contents. The document shall contain a table of contents providing the number, title, and page number of each titled paragraph, figure, table, and appendix. For data in a database or other alternative form, this information shall consist of an internal or external table of contents containing pointers to, or instructions for accessing, each paragraph, figure, table, and appendix or their equivalents.

e. Page numbering/labeling. Each page shall contain a unique page number and display the document number, including version, volume, and date, as applicable. For data in a database or other alternative form, files, screens, or other entities shall be assigned names or numbers in such a way that desired data can be indexed and accessed.

f. Response to tailoring instructions. If a paragraph is tailored out of this document, the resulting document shall contain the corresponding paragraph number and title, followed by "This paragraph has been tailored out." For data in a database or other alternative form, this representation need occur only in the table of contents or equivalent.

g. Multiple paragraphs and subparagraphs. Any section, paragraph, or subparagraph in this DID may be written as multiple paragraphs or subparagraphs to enhance readability.

h. Standard data descriptions. If a data description required by this document has been published in a standard data element dictionary specified in the contract, reference to an entry in that dictionary is preferred over including the description itself.

i. Substitution of existing documents. Commercial or other existing documents, including other project plans, may be substituted for all or part of the document if they contain the required data.