Assignment 2: Test Case Planning

CSCI265

Sami Al-Qusus

Dec 04, 2017

VIU CSCI 265

Project: Contact Hash Builder
Products: buildContactHash & buildReverseHash

TEST PLAN

Document Version 1.0
04/12/2017

VERSION HISTORY

Version #	Implemented by	Revision Date	Approved By	Approval Date	Reason
1.0	Sami Al-Qusus	04/12/17			Test Plan

TABLE OF CONTENTS

1 INTRODUCTION

1.1 PURPOSE OF THE TEST PLAN DOCUMENT

The purpose of this document is to describe the collection of test cases, strategy and approach to be followed in testing the project's products. The intended audience of this document is the instructor, software development team, and software testing team. Portions of this document may be shared with individuals whose input or approval into the testing process is needed and understand the scope of work that must be accomplished by testing entity.

2 USE CASE TESTING

2.1 TEST RISKS/ISSUES

- Tester may not be provided or have access to all required software required for integration and system testing.
- Software may behave different on different platforms or operation environments.

2.2 ITEMS TO BE TESTED / NOT TESTED

Item to Test	Description	Date	Resposibility
buildContactHash	takes a (reference to) a contact list as its parameter, # fills in a contact hash, and returns a reference to the contact hash	04/12/17	Tester
buildReverseHash	takes a (reference to) a contact hash as its parameter, # fills in a reverse contact hash, and returns a reference to the reverse hash	04/12/17	Tester

* validEmail and validPhone are not to be tested.
* Tests are not testing for functionality.

2.3 TEST APPROACH(S)

Unit testing will be performed on products as they are developed. Test cases will be used to test against system boundaries with respect to possible user inputs. New test cases should be added as components are further developed for constraints and system boundaries. If previous constraints are transformed, existing test cases are to be modified to comply with latest development. Tests are to be automated with use of perl's test modules. Integration test will be performed by each component tester.

2.4 TEST REGULATOR / MANDATE CRITERIA

Use perl's Test::More to automate the testing .Test cases should be stored in same directory but in a separate file from perl test script and should only be read by the script. lab08.pl is to be used by perl script to access components that needed testing. Lab08.pl will be stored in same directory as well.

2.5 TEST PASS / FAIL CRITERIA

Tests executed on components only pass when they fulfill constraints prescribed by component's specification. Test cases set will express pass or fail, as determined by perl's Test::More.

2.6 TEST ENTRY / EXIT CRITERIA

Automated perl script is executed to start testing and testing is automatically stopped at end of perl script.

2.7 TEST DELIVERABLES

Deliverables that will result from the testing process is perl's Test::More output as TAP.

2.8 TEST SUSPENSION / RESUMPTION CRITERIA

When components are deemed ready to test by developer, test cases automated through perl script should be run. If test fails for a particular case, testing will suspend until developer fixes issues with components. Once components are deemed ready to test by developer test resumes. Once test passes, it is suspended.

2.9 TEST ENVIRONMENT / STAFFING / TRAINING NEEDS

Products are required to be functional for testing. Testing should be competent with perl.

2.10 TEST CASES

2.10.1 Invalid_format

Description	Checks if passed contact is structurally invalid, valid meaning a list of lists, where the inner lists each contain exactly two strings
Test Data	[“justemail@gmail.com”]
Expected Result	'Error: invalid contacts'
Expected Return Value	0
Product Case Tests	buildContactHash
Dependencies	-

2.10.2 Invalid_email

Description	If any entry contains an invalid email address the function displays a warning and ignores that email entry
Test Data	[“wronggmail.com”, “1234567890”]
Expected Result	'Warning: invalid email'
Expected Return Value	No return value
Product Case Tests	buildContactHash, buildReverseHash
Dependencies	validEmail, validPhone

2.10.3 Invalid_phone

Description	If any entry contains an invalid phone number the function displays a warning and ignores that phone entry
Test Data	[“sami@gmail.com”, “1wrong#34”]
Expected Result	'Warning: invalid phone'
Expected Return Value	No return value
Product Case Tests	buildContactHash, buildReverseHash
Dependencies	validEmail, validPhone

2.10.4 Duplicate_email

Description	If a duplicate email address is encountered the function displays a warning and replaces the old associated phone number with the new one
Test Data	[“sami@gmail.com”, “1234567890”] given twice
Expected Result	'Warning: duplicate email'
Expected Return Value	No return value
Product Case Tests	buildContactHash
Dependencies	-

2.10.5 Duplicate_phone

Discription	If a duplicate phone number is encountered the function displays a warning and replaces the old associated email address with the new one
Test Data	[“joh@gmail.com”, “0333111999”] given twice
Expect Result	'Warning: duplicate phone'
Expected Return Value	No return value
Product Case Tests	buildReverseHash
Dependencies	-

***Note: no test case here should test more than one of the tests. That is to know specifically which test fails for which requirements.**