Funky Torrents

Software Test Plan

Version 0.8
IT University of Göteborg
TIG029 – Software Architecture for Distributed Systems

1. Revision History

Version	Date	Description	Author
0.1	2010-11-25	Test Plan	Ali Issa
0.2	2010-11-30	Introduction added	Ali Issa
0.2	2009-12-3	Added test cases	Ali Issa
0.3	2009-12-8	Requirements list added	Ali Issa
0.4	2009-12-14	Requirements testing matrix added	Ali Issa
0.5	2009-12-15	Simulation Application test matrix added	Ali Issa
0.6	2010-12-16	Test cases added	Ali Issa
0.7	2010-12-17	Test cases modified	Ali Issa
0.8	2010-12-17	revision history added	Ali Issa

Table 1. Revision history

Table of Content

2. Introduction

This documentation describes the requirements specified for this project and the methods for testing the various modules in order to ensure that the requirements have been meet.

3. Testing

3.1.Requirements

Requirements	Description
A1	Hash password (not integrated)
B1	Get announce from torrent file
B2	Get comment from torrent file
B3	Get hash from torrent file
B4	Get length from torrent file
B5	Get piece length from torrent file
B6	Get filename
C1	Write to a specific position in a torrent file
C2	Read a torrent file
D1	Read and parse torrent file
D2	Read a torrent file which contains multiple files
D3	Read a torrent file which contains multiple announce
D4	Decode the bencode
E1	Download file
E2	Interacting with GUI to download a file (integration test)

F1	Login, using username and password
G1	Encode Integer
G2	Encode String
G3	Encode List

Table 2. List of requirements

3.2.Requirements Testing Matrix

Application test	Requirements covered
At1	A1
At2	B1-B6
At3	C1-C2
At4	D1-D4
At5	E1-E2
At6	F1
At7	G1-G3

Table 3. Requirements Testing Matrix

3.3.Simulation Application Test Matrix

Test ID	Comment	Result
At1	Not in a testable state	Fail
At2	Not in a testable state	Fail
At3	Not in a testable state	Fail
At4	Not in a testable state	Fail
At5	Not in a testable state	Fail
At6	Not in a testable state	Fail
At7	Not in a testable state	Fail

Table 4 Simulation Application Test Matrix (Test conducted during iteration 1)

Test ID	Comment	Result
At1	Test succeeded	Pass
At2	Not in a testable state	Fail
At3	Not in a testable state	Fail
At4	Not in a testable state	Fail
At5	Not in a testable state	Fail
At6	Not in a testable state	Fail
At7	Not in a testable state	Fail

Table 5. Simulation Application Test Matrix (Tests conducted during iteration 2)

Test ID	Comment	Result
At1	Test succeeded	Pass
At2	Test succeeded	Pass
At3	Read a torrent file:Pass Write to a specific position in a torrent: Fail, every time the code is recompiled, text is overwritten in torrent file.	Fail
At4	If the torrent file is large, it fails to read deep enough	Fail
At5	Not in a testable state	Fail
At6	Not in a testable state	Fail
At7	Test succeeded	Pass

Table 6. Simulation Application Test Matrix (Tests conducted during iteration 3)

Test ID	Comment	Result
At3	Test succeded	Pass
At4	Read a torrent file which contains multiple files: Fail, Read a torrent file which contains multiple announce: Fail, Decode the bencode:Pass, Read a torrent file which contains a single file:Pass. The team considers the behavior of the module acceptable therefore passed.	Pass
At5	Not in a testable state	Fail
At6	Problems with integrating the Gui to the Erlang code, therefore it is considered as not in a testable state	Fail

Table 7. Simulation Application Test Matrix (Tests conducted during iteration 3)

Test ID	Comment	Result
At5	Test succeed	Pass
At6	Test succeed	Pass

Table 8. Simulation Application Test Matrix (Tests conducted during iteration 3)

3.4.Test Cases

Test Name	At1
Purpose	Verify requirement A1
Steps	Enter a string in the hash function and bound it to Try1 Enter the same string in the hash function and bound it to Try2 Try1 == Try2, should return true Enter another string in the hash function with a lower case letter and bound it to try3. Enter Try3=Try2, should return false. Enter a string with Scandinavian letters (like öä) and bound it to Try4 Enter the same string and bound it to Try5 Enter Try5==Try4, should return true. Means that the function can handle Scandinavian letters. Enter a string in the hash function and bound it to NotEq1 Enter a different string in the hash function bound it to NotEq2 Enter NotEq1==NotEq2, should return false Enter combination of integer in the hash function and bound it to INTEGER1 Enter a the same combination of integers in the hash function and bound it to

	INTEGER2 INTEGER1==INTEGER2 should return true

Table 9 Application test 1

Test Name	At2
Purpose	Verify requirements B1-B6
Steps	Open the pars module Enter pars:get_announce(“atorrentfile.torrent”) Should return the announce in the torrent file Enter pars:get_comment (“atorrentfile.torrent”) Should return the comment in the torrent file Enter pars:get_pieces(“atorrentfile.torrent”) Should return the pieces in the torrent file Enter pars:get_hash(“atorrentfile.torrent”) Should return the hash in the torrent file Enter pars:get_piece_length(“atorrentfile.torrent”) Should return the length of the piece in the torrent file Enter pars:get_length(“atorrentfile.torrent”) Should return the length of the torrent file Enter pars:get_file_name(“atorrent.torrent”) Should return the name of the torrent file

Table 10. Application test 2

Test Name	At3
Purpose	Verify requirements C1-C2
Steps	Open the filedata module Implement an io:format in the read function,to print out content of file Enter filedata:read(File) Should printout the content of the file Enter filedata:writer("torrentfile.torrent",10,"Test") Open torrentfile.torrent "Test" should have been added in position 10 Enter filedata:writer("torrentfile.torrent,30,"testagain") Open torrentfile.torrent "testagain" should have been added in position 30 and "Test" should still be at position 10

Table 11. Application test 3

Test Name	At4
Purpose	Verify requirements D1-D4
Steps	Open pars and filedata module Enter pars:bencode_reader("4:test") Should return {{string,"test"},[]} Enter pars:bencode_reader("i100e") Should return {{integer,100},[]}

	Enter pars:bencode_reader("d4:test2:is5:valid6:greate") Should return [{{string,"valid"},{string,"greate"}},{{string,"test"},{string,"is"}}] Enter {Datainfile,_}=filedata:read(torrentfile) ==pars:search(Datainfile) Should return true, shows that the torrent file has been read and parsed. Same as step 8, enter name of a torrent file in the read function but with a torrent file which contains multiple files Should return true Same as step 8 and 10 enter name of a torrent file but with a torrent file which contains multiple announce Should return true

Table 12. Application test 4

Test Name	At5
Purpose	Verify requirements E1-E2
Steps	Run all modules included in the project Enter connector:download(“the seeded torrentfile”) Should return an atom as last returned value, download_complete the torrent file should then be located in the specified folder Interact with the GUI, login, search for a file and download it The file should be located in the specified folder

Table 13. Application test 5

Test Name	At6
Purpose	Verify requirements F1
Steps	Interact with the user interface by entering a registered username and password Select login button User should receive a confirmation that logging in succeeded Select logout button User should receive a confirmation that logging out succeeded Enter unregistered or invalid username and password User should receive a confirmation that logging in failed

Table 14. Application test 6

Test Name	At7
Purpose	Verify requirements G1-G3
Steps	Run encoder module Enter encoder:enc({integer,2}). Should return “i2e” Enter encoder:enc({string,”test”}). Should return “4:test” Enter encoder:enc(encoder:enc({list,[{string,"test"}]}. Should return "l4:teste" Enter encoder:enc({list,[{integer,2}]}). Should return "li2ee"

Table 15. Application test 7

4. References

- Integration plan template. Project Connections. [Online]. Available: http://www.projectconnections.com/knowhow/subsets/sample-templates/integration plan.doc
- How to design test cases [Online], Available at: http://it.toolbox.com/blogs/enterprise-solutions/a-test-case-template-17193