Karelia
AMMATTIKORKEAKOULU

Testing report

WormGame

Version 1.1

Karelia	TIKO	LTP7024 Testausmenetelmät
Author: Juhani Pirinen		Printed: 18th oct 2015
Distribution: Teacher of the course, Public in GitHub		
Status of the document: draft		Edited: 18th oct 2015

VERSION HISTORY

Version	Date	Authors	Description
1.0	7th oct 2015	Juhani Pirinen	First draft (in english)
1.1	18th oct 2015	Juhani Pirinen	Tests are made and report written

1. PREFACE

This is a testing report of current WormGame prototype made with NodeJS, for testing methodolohy course. Testing was made according to Testing plan version 1.3, Technical specification version 1.0 and Software requirements specification 2.1.

According to testing plan, main objectives of the testing are:

- To check, that server API behaviour in this limited context does match the technical specification
- To check, is this application safe enough to run in public internet

Our assignment was to test *only one software component*. Restful API component that provides user authentication features (registration and login) was chosen for testing. Testing environment is exactly the same as specified in the testing plan. The latest Jasmine testing framework version 2.3 was used from npm package jasmine (not jasmine-node).

Installation and tests were run after package's instructions on page https://www.npmjs.com/package/jasmine. After installation of Jasmine, first the GameWorm application must be running (can be started with command *node server.js*) and then tests can be run (with the command *jasmine*).

2. CONTRADICTIONS AND ANOMALIES

Test cases T06, T07, T11 and T12 were not taken for testing, because it turned out, that chosen Jasmine framework isn't capable of testing software or WormGame's software structure needed changes in order to make it testable with Jasmine. The dropped test cases included workflow, where testing framework should be able to simulate both server and client side environments and read values of variables from certain breakpoint in a live situation, like debugger software (similar to node inspector) would be able to do.

This is only an elementary course of testing and writing a test driver like this would be too difficult and laborous. This time, I was only able to make a test driver, that simulates client by making HTTP requests to REST Api and is able to check from server side, how database has changed by the requests.

T07 included testing login feature with valid username and password combination. When test T07 isn't made, then testing the login with valid equivalence classes should be replaced with a new test case, that currently doesn't exists in the testing plan. Due to lack of time, this new case was not made. However, T02 covers testing this feature partially (of server-side only). Test case T08 is about testing with invalid username and password combinations, and this was made.

Test framework crashed, when T08 was attempted with password length 1024000 chars. It didn't make sense to write a test case, that couldn't be executed at all. Therefore, this test was made with password length 10000 chars.

Additionally T11-T12 included things, that were not yet in technical specification or systems requirements specification, and therefore were not implemented.

3. COVERAGE

As specified in the testing plan, the following test cases include testing of Restful API module:

- T01: Username is unique and can be found from the database
- T02: User registration to database with parameters from valid equivalence classes
- T03: User registration to database with parameters from invalid equivalence classes
- T06: Disconnected user is no more logged-in to the system
- T07: Disconnected user can re-login to the system
- T08: User login with invalid username or password
- T11: Abusive user is banned from accessing the system by IP
- T12: Abusive user is banned from accessing the system by username

Made testing covers only tests T01, T02, T03 and T08. Reason for dropping T06, T08, T11 and T12 is explained in the chapter 2 of this report.

4. RESULTS

Test cycle 1

Test case: T01 Username is unique and can be found from the database **Status:** pass			
Spec	**Expected result**	**Actualized result**	**Inspections if test failed**
has registered users in the database	true	true	
has each username in database exists once and only once	true	true	

Test case: T02: User registration to database with parameters from valid equivalence classes **Status:** fail			
Spec	**Expected result**	**Actualized result**	**Inspections if test failed**
username length 4 valid characters, password length 16	true	true	

valid characters			
username length 16 valid characters, password length 8 valid characters	true	true	
has all registered usernames in database	true	true	
has encrypted passwords in database that match with encrypted test passwords	true	false	Database column field size for passwords seems to be only 20 varchars. Passwords in encrypted format are longer than the passwords in plain text format. Long passwords are then saved only partially to the database and therefore doesn't match with the right value entered by the user.

Test case: T03: User registration to database with parameters from invalid equivalence classes
Status: fail

Spec	**Expected result**	**Actualized result**	**Inspections if test failed**
username length 3	NOK	OK	Seems that there's no any input value

valid characters, password length 16 valid characters			data filters implemented.
username length 17 valid characters, password length 8 valid characters	NOK	OK	Seems that there's no any input value data filters implemented.
username length 4 valid characters, password length 17 valid characters	NOK	OK	Seems that there's no any input value data filters implemented.
username length 16 valid characters, password length 7 valid characters	NOK	OK	Seems that there's no any input value data filters implemented.
username length 4 invalid characters, password length 16 valid characters	NOK	OK	Seems that there's no any input value data filters implemented.
username length 16 valid characters, password length 8 invalid characters	NOK	OK	Seems that there's no any input value data filters implemented.
username length 3 invalid characters,	NOK	OK	Seems that there's no any input value data filters implemented.

password length 16 valid characters			
username length 4 valid characters, password length 17 invalid characters	NOK	OK	Seems that there's no any input value data filters implemented.
username length 3 invalid characters, password length 16 invalid characters	NOK	OK	Seems that there's no any input value data filters implemented.
username length 17 invalid characters, password length 7 invalid characters	NOK	OK	Seems that there's no any input value data filters implemented.
none of the tested usernames were found from database	0	10	All tested usernames were saved to the database, when nothing should be saved. Seems that there's no any input value data filters implemented.

Test case: T08: User login with invalid username or password **Status:** fail			
Spec	**Expected result**	**Actualized result**	**Inspections if test failed**

login correct username and incorrect password	NOK	NOK	
login incorrect username and correct password	NOK	NOK	
login empty username and correct password	NOK	NOK	
login some other user's correct username and own correct password	NOK	NOK	
login correct username and incorrect misc password of length 10000 characters	NOK	NOK	
login correct username + sql injection string " or 1=1' and correct password	NOK	NOK	
login username as only sql injections	NOK	no result	There's no result, because the server crashes before it responds. Seems that

string ”') or true--” and incorrect password			there's no any input value data filters implemented.

Test cycle 2

Test case: T01 Username is unique and can be found from the database **Status:** pass			
Spec	**Expected result**	**Actualized result**	**Inspections if test failed**
has registered users in the database	true	true	
has each username in database exists once and only once	true	true	

Test case: T02: User registration to database with parameters from valid equivalence classes **Status:** pass			
Spec	**Expected result**	**Actualized result**	**Inspections if test failed**
username length 4 valid characters, password length 16 valid characters	true	true	
username length 16 valid characters,	true	true	

password length 8 valid characters			
has all registered usernames in database	true	true	
has encrypted passwords in database that match with encrypted test passwords	true	true	

Test case: T03: User registration to database with parameters from invalid equivalence classes **Status:** pass			
Spec	**Expected result**	**Actualized result**	**Inspections if test failed**
username length 3 valid characters, password length 16 valid characters	NOK	NOK	
username length 17 valid characters, password length 8	NOK	NOK	

valid characters			
username length 4 valid characters, password length 17 valid characters	NOK	NOK	
username length 16 valid characters, password length 7 valid characters	NOK	NOK	
username length 4 invalid characters, password length 16 valid characters	NOK	NOK	
username length 16 valid characters, password length 8 invalid characters	NOK	NOK	
username length 3 invalid characters, password length 16 valid characters	NOK	NOK	
username length 4 valid characters, password length 17 invalid characters	NOK	NOK	

username length 3 invalid characters, password length 16 invalid characters	NOK	NOK	
username length 17 invalid characters, password length 7 invalid characters	NOK	NOK	
none of the tested usernames were found from database	0	0	

Test case: T08: User login with invalid username or password
Status: pass

Spec	**Expected result**	**Actualized result**	**Inspections if test failed**
login correct username and incorrect password	NOK	NOK	
login incorrect username and correct password	NOK	NOK	

login empty username and correct password	NOK	NOK	
login some other user's correct username and own correct password	NOK	NOK	
login correct username and incorrect misc password of length 10000 characters	NOK	NOK	
login correct username + sql injection string " or 1=1' and correct password	NOK	NOK	
login username as only sql injections string ”') or true--” and incorrect password	NOK	NOK	

5. EVALUATION

Only two test cycles were needed in order to have all tests to pass.

Most test failed on first test run phase. The test cases were quite trivial, so results of test case T03 were expected to fail n the first test phase as it was known that there was no input checking. Fail of test case 02 was a surprise in the first test phase and a new issue was found in database schema. Pass of most expectations of test case 08 was suprising, even when there was not any input checking in the first test phase. Test case 08 suprisingly crashed the server, that was not expected either.

All tests passed on second test run phase.

6. TAKEN MEASURES

Learning how to user Jamine framework correctly with REST API testing and database testing was somehow challenging, now when it was tried the very first time, and test cases itself needed some debugging before working correctly. Especially getting things tests working properly with asynchronous callbacks was tricky, but the latest version of Jamine made it much easier, than it may have been earlier.

7. ACCEPTANCE

The whole rest api isn't tested, so the testing hasn't yet reached it's objectives, however results of the tested parts of rest api are acceptable. From these parts the software now matches technical specification, but still the application isn't safe to run in public

internet, as there's many obvious shortages in software design and incomplete coverage of testing.