Karelia
AMMATTIKORKEAKOULU

Testing plan

WormGame

Version 1.3

Karelia	TIKO	LTP7024 Testausmenetelmät
Author: Juhani Pirinen		Printed: 6th oct 2015
Distribution: Teacher of the course, Public in GitHub		
Status of the document: draft		Edited: 6th oct 2015

VERSION HISTORY

Version	Date	Authors	Description
1.0	2nd oct 2015	Juhani Pirinen	First draft (in english)
1.1	3rd oct 2015	Juhani Pirinen	First actual version
1.2	5th oct 2015	Juhani Pirinen	Added test cases. Nearly finished version.
1.3	6th oct 2015	Juhani Pirinen	Small corrections made

1. PREFACE

1.1 Purpose and extent

This is a testing plan for testing, how well game server API is able to handle game client's game UI related requests. Testing is limited to a few use cases only, not the whole application. This testing plan is only a course work for a course project, and targeted to teachers and students of network application development and testing methods courses in our UAC.

1.2 Product and environment

This is a multi-player online version of the classic worm game, that will run as client-server application in web browser and web server. Only authenticated users can play the game. Additionally users can chat and see ranking lists.

Client is a web browser, that has support to HTML5 and ECMAscript version 5 of javascript. Game server runs in Node.JS framework, that users Express.JS framework, MySQL database server and Socket.IO javascript library for using web sockets protocol.

1.3 Objectives of testing

In this course, our assignment is to create a system testing plan for some application. It might be more interesting to test game API than UI, but it wouldn't meet requirements of the assignment, because testing plan must be limited only to a few use cases. Unfortunately requirements of the actual game are not captured by the use cases. Use cases covers only user interface of the game, but not the game

itself. Additionally, game API is still very incomplete, so it isn't yet in testable condition, but the final deadline of the testing course is dated before it's ready; very soon. However, game UI is already in testable condition. For these reasons, *this testing plan must be limited to the game UI only.*

In client-server application, it is possible to test only the client, only the server or both client and server. To keep amount of course work in recommended limits, testing plan is limited to server-side only. Server API is capable to handle game's all UI related events. Therefore, they are fully testable without a client, altough client is the only part of the system, that would produce the frontend UI, while server is just a headless backend. In testing environment, a test driver software is going to simulate the API request on behalf of the client.

Main objectives of the testing plan are:

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

1.4 Definitions, notations, abbrevations

API	Application Protocol Interface
Client-Server	Archit. of two systems communicating over network
Ecmascript5	Edition 5 of JavaScript language
ExpressJS	NodeJS web application server framework
HTML5	Hypertext Markup Language, version 5
HTTP	Hypertext Transfer Protocol
Jasmine	Behavour-driven testing framework for JavaScript
TCP	Transmission Control Protocol

MySQL 5.5	MySQL relational database server version 5.5
NodeJS	Runtime environment for server-side web applications
Socket.IO	JavaScript library for realtime web applications
Websockets	Bi-directional communication protocol over TCP
WormGame	Multi-Player version of classic Snake video game

1.5 References

Automattic. Socket.IO library. http://www.socket.io 3rd oct 2015.

Ecma International. ECMAScript 5.1. http://www.ecma-international.org/ecma-262/5.1/ 3rd oct 2015.

Kasurinen J.P. 2013. Ohjelmistotestauksen käsikirja. Jyväskylä: Docendo.

IETF. Hypertext Transfer Protocol, HTTP/1.1. RFC2616. https://www.ietf.org/rfc/rfc2616.txt 3rd oct 2015.

IETF. The WebSocket Protocol. RFC6455. https://www.ietf.org/rfc/rfc6455.txt 3rd oct 2015.

Node.js foundation. NodeJS environment. https://nodejs.org 3rd oct 2015.

Oracle Corp. MySQL database server. https://www.mysql.com/ 3rd oct 2015

Pivotal Labs. Jasmine - a Behavior Driven Development testing framework for JavaScript. http://jasmine.github.io/ 3rd oct 2015.

StrongLoop, Inc. ExpressJS framework. http://expressjs.com/ 3rd oct 2015.

Wikipedia. Snake (video game). https://en.wikipedia.org/wiki/Snake_%28video_game%29 3rd oct 2015.

Word Wide Web Consortium. Cascading Style Sheets, CSS. http://www.w3.org/Style/CSS/ 3rd oct 2015.

Word Wide Web Consortium. Hypertext Markup Language, HTML5. http://www.w3.org/TR/html5/ 3rd oct 2015.

1.6 Overview to the document

This document begins from determinations of objects of testing, testing methods and testing environment, progresses to test cases listing, and finishes to testing requirements and criterions.

2 OBJECTS OF TESTING

The following Node.JS server side components if the game will be tested:

- Rest API of user login and registration features
- WebSockets API of chat feature
- Game server application

3 TESTING APPROACH

- Black box testing is used as a primary testing methodology. Boundary analysis is made when planning test cases. This testing method is chosen over glass-box testing, because:
  a) we test only APIs so that test driver simulates the client,
  b) we've got a very limited time for planning and testing, and
  c) we want to know how safe is it run the application in public internet, can it be abused somehow (→ this is why there exists a few tests that are not directly based on system requirements)

- In a few cases glass-box-like testing methods are applied when accessing database or system variables directly to ensure better the inner state of the system.

- Jasmine testing framework is used for running the tests.

- Motivation to testing is just to learn basics of NodeJS testing, and how to write testing plans and testing reports.

- Due to lack of time, and to stay in limits of assignment, testing is limited only to three features of game server API.

- All chosen test cases will be carried out carefully and correctly.

- Software will be debugged during testing so that all failed tests will pass in the last testing cycle.

- Despite the previous statement, some tests are allowed to fail, if deadline of testing is reached or repairing the code or software design would be too difficult or time consuming.

- All testing results will be documented in testing report.

4 HUMAN RESOURCES AND EDUCATION REQUIREMENTS

4.1 Staff

This is a course work of a single student.

4.2 Education, knowledge and skills

Basic knowledge of JavaScript, NodeJS, web applications and software testing.

5 AREAS OF RESPONSIBILITY

Not applicable to this testing plan.

6 TESTING ENVIRONMENT

6.1 Hardware

Testing will be done in small NodeJS development environment that runs inside VirtualBox, in any modern laptop or desktop computer.

6.2 Software

- Oracle VirtualBox: Linux Lubuntu OS as Guest, Windows 10 as Host
- NodeJS environment with ExpressJS, Socket.IO and MySQL server. Some NodeJS tools like Node Inspector may be used.
- Sublime Text 3 code editor
- Shell console in Linux
- Firefox and/or Chrome/Chromium web browsers
- Libre Office for writing testing plan and report.

6.3 Security

Normal development environment security level is enough. Both operating systems should be up to date and important files should be backed up.

6.4 Equipments and testing data

No special equipments needed. Testing data can be produced manually or with help of data generators like www.generatedata.com.

7 TEST CASES

7.1 Database testing

TEST CASE ID: T01

NAME:

Username is unique and can be found from the database

PRIORITY:

High

PRECONDITION:

1. Database server is running.
2. There is registered users in the database.

DESCRIPTION OF WORKFLOW:

1. Read all usernames from the database
2. Check that each username in list exists once and only once

EXPECTED RESULTS:

Test will pass. All usernames are unique.

POSTCONDITION:

All usernames in database are unique.

REQUIREMENTS TO TEST:

User registration

TEST CASE ID: T02

NAME:

User registration to database with parameters from valid equivalence classes

PRIORITY:

High

PRECONDITION:

1. The system is running
2. Used usernames doesn't exists already in the database

DESCRIPTION OF WORKFLOW:

1. Using following test values of valid usernames (length between 4 and 16, allowed characters a-z,A-Z,0-9) and passwords (length between 8 and 16, allowed characters a-z,A-Z,0-9):
a) username length 4 valid characters, password length 16 valid characters
b) username length 16 valid characters, password length 8 valid characters

2. Read system response

3. Read usernames and passwords from the database

4. Encrypt passwords

5. Check that all the following are true:
a) system responded that registration was successful
b) all registered usernames were found from database
c) password was encrypted: it didn't match the plain password
d) decrypted password matched with the plain password

EXPECTED RESULTS:

Test will pass. Checks A-D are all true.

POSTCONDITION:

There's two user accounts in the database and REST API functions properly for user registration.

REQUIREMENTS TO TEST:

User registration

TEST CASE ID: T03

NAME:

User registration to database with parameters from invalid equivalence classes

PRIORITY:

High

PRECONDITION:

1. The system is running

DESCRIPTION OF WORKFLOW:

1. Using following test values of invalid usernames (valid when length between 4 and 16, allowed characters a-z,A-Z,0-9; otherwise invalid) and passwords (valid when length between 8 and 16, allowed characters a-z,A-Z,0-9; otherwise invalid) combinations:
a) username length 3 valid characters, password length 16 valid characters
b) username length 17 valid characters, password length 8 valid characters

c) username length 4 valid characters, password length 17 valid characters

d) username length 16 valid characters, password length 7 valid characters

e) username length 4 invalid characters, password length 16 valid characters

f) username length 16 valid characters, password length 8 invalid characters

g) username length 3 invalid characters, password length 16 valid characters

h) username length 4 valid characters, password length 17 invalid characters

i) username length 3 invalid characters, password length 16 invalid characters

j) username length 17 invalid characters, password length 7 invalid characters

2. Read system response

3. Read usernames and passwords from the database

4. Check that all the following are true:

a) system responded that registration was insuccessful

b) none of the tested usernames were found from database

EXPECTED RESULTS:

Test will pass. Invalid usernames and passwords are not allowed, registration fails and no new accounts are found from the database.

POSTCONDITION:

The system's condition is the same than before the test. There is no new registrations.

REQUIREMENTS TO TEST:

User registration

TEST CASE ID: T04

NAME:

Chat messages with parameters from valid equivalence classes

PRIORITY:

Medium

PRECONDITION:

1. The system is running

DESCRIPTION OF WORKFLOW:

1. Access system with two logged-in and two anonymous users at the same time

2. Use following test values of valid chat messages (length between 2 and 128, allowed characters a-z,A-Z,0-9, comma, dot, single quote, question mark, minus sign, exlamation point, space, colon):
a) all allowed values (length 70 characters) as logged-in user
b) two random allowed character as logged-in user
c) 128 characters length string as logged-in user
d) all allowed values (length 70 characters) as anonymous user
e) two random allowed character as anonymous user
f) 128 characters length string as anonymous user

3. Save exact send time of each message

4. Read sent messages / system response with all 4 user clients

5. Read sent messages from chat log of database

6. Check that all the following are true:
a) All sent messages are received by 4 clients, with the same string, same nick and same time (+- 1 second)
b) All messages are found from the database, with the same string, same nick and same time (+- 1 second)

EXPECTED RESULTS:
Test will pass. All unmodified chat messages are found from database and received by connected clients.

POSTCONDITION:
Unmodified Chat messages, with nick and sent time are in database and sent to connected clients.

REQUIREMENTS TO TEST:
Chat

TEST CASE ID: T05

NAME:
Chat messages with parameters from invalid equivalence classes

PRIORITY:
Medium

PRECONDITION:
1. The system is running

2. Two registered user accounts can be used by testing environment

DESCRIPTION OF WORKFLOW:

1. Access system with two logged-in and two anonymous users at the same time

2. Use following test values, that are not valid chat messages (length between 2 and 128, allowed characters a-z,A-Z,0-9, comma, dot, single quote, question mark, minus sign, exlamation point, space, colon):

a) empty message (length 0 characters) as logged-in user

b) 1 valid character as logged-in user

c) 129 valid characters as logged-in user

d) 2 invalid characters as logged-in user

e) 128 invalid characters as logged-in user

f) 1 invalid character as logged-in user

g) 129 invalid characters as logged-in user

h) empty message (length 0 characters) as anonymous user

i) 1 valid character as anonymous user

j) 129 valid characters as anonymous user

k) 2 invalid characters as anonymous user

l) 128 invalid characters as anonymous user

m) 1 invalid character as anonymous user

n) 129 invalid characters as anonymous user

3. Read sent messages / system response with all 4 user clients

4. Read sent messages from chat log of database

5. Check that all the following are true:

a) All 4 clients have received no sent messages

b) Sent messages are not found from the database

EXPECTED RESULTS:

Test will pass. Invalid chat messages does not go through the system and are not saved to the system.

POSTCONDITION:

The system's condition is the same than before the test. There is no new chat messages.

REQUIREMENTS TO TEST:

Chat

User login

7.2 External associated parts testing

Not applicable to this testing plan.

7.3 User interface testing

Not applicable to this testing plan. We test the server API only, that is ”headless” during testing, without user interface.

7.4 Interfaces and extensions testing

TEST CASE ID: T06

NAME:

Disconnected user is no more logged-in to the system

PRIORITY:

High

PRECONDITION:

1. The system is running
2. Registered user accounts can be used by testing environment
3. Log-out feature isn't implemented or it isn't used

DESCRIPTION OF WORKFLOW:

1. Login to system with correct username and password.

2. Ensure that user is logged in: username can be found from websocket userlist and online players object array, and there are no errors

3. Disconnect user by the closing websocket connection or by closing browser window

4. Look for username from websocket userlist information from system

5. Look for username from online players object array from system

6. Check that all the following are true:
a) Username isn't found from websocket userlist
b) Username isn't found from online players object array

EXPECTED RESULTS:

The test will pass. In this system loosing a websocket connection should lead to a log-out too, to ensure that game works correctly.

POSTCONDITION:

After disconnection, user isn't logged-in to the system.

REQUIREMENTS TO TEST:

User login

TEST CASE ID: T07

NAME:

Disconnected user can re-login to the system

PRIORITY:

High

PRECONDITION:

1. The system is running
2. Registered user accounts can be used by testing environment
3. Log-out feature isn't implemented or it isn't used
4. Test case T06 did pass

DESCRIPTION OF WORKFLOW:

1. Accomplish test case T06 as a first step of this test case, so that test is passed

2. Login to system with correct username and password.

3. Ensure that user is logged in: username can be found from websocket userlist and online players object array, and there are no errors

EXPECTED RESULTS:

Test will pass. User can login after disconnection.

POSTCONDITION:

User is logged-in to the system.

REQUIREMENTS TO TEST:

User login

7.5 Printing functionalities testing

Not applicable to this testing plan.

7.6 Security testing

TEST CASE ID: T08

NAME:
User login with invalid username or password

PRIORITY:
High

PRECONDITION:
1. The system is running
2. There exists user account in the system

DESCRIPTION OF WORKFLOW:
1. Use invalid combinations of correct and incorrect usernames and password, with valid and invalid lengths and characters (valid when length between 4 and 16, allowed characters a-z,A-Z,0-9; otherwise invalid) and passwords (valid when length between 8 and 16, allowed characters a-z,A-Z,0-9; otherwise invalid):
a) correct username and incorrect password (both have valid lenght and only allowed characters)
b) incorrect username and correct password (both have valid lenght and only allowed characters)
c) empty username and correct password
d) some other user's correct username and own correct password
e) correct username and incorrect misc password of length 1024000 characters (username has valid lenght and only allowed characters)
f) correct username + sql injection string ” or 1=1” and correct

password (both have valid lenght and only allowed characters)
g) username as only sql injections string ”') or true--” and incorrect password (both have valid lenght, password also allowed characters)

2. Read system response

3. Read all http headers

4. Check that all the following are true:
a) login to system doesn't success with any values
b) server doesn't crash by any values
c) http headers responded by server doesn't contain any additional, perhaps sensitive data

EXPECTED RESULTS:
Test will pass. Login can succeed only with correct username and password combination.

POSTCONDITION:
The system's condition is the same than before the test. There is no new registrations.

REQUIREMENTS TO TEST:
User registration

TEST CASE ID: T09

NAME:
Chat abuse by ”flooding”

PRIORITY:

High

PRECONDITION:

1. System is running

2. There exists user account in the system

DESCRIPTION OF WORKFLOW:

1. Send 128 character length chat messages from ”Lorem Ipsum” randomly every 1-5 second, for 5 minutes

2. Read system response

3. Check that the following are true:

a) server will stop transmitting the flood under 1 minutes

EXPECTED RESULTS:

Test will pass. Flooding chat messages do not go through the system and are not saved to the system. If system is open game chat server in public internet, it must be able to reject flooding, at least.

POSTCONDITION:

The system's condition is the same than before the test. There is no new chat messages, after floowing was detected.

REQUIREMENTS TO TEST:

Not mentioned in requirements

TEST CASE ID: T10

NAME:

Chat abuse by ”bad words”

PRIORITY:

High

PRECONDITION:

System is running

DESCRIPTION OF WORKFLOW:

1. Send chat message containing random ”bad words” from this list https://gist.github.com/jamiew/1112488

2. Read server response

3. The following conditions are true:
a) server doesn't transmit messages with ”bad words”

EXPECTED RESULTS:

Test will pass. Invalid chat messages do not go through the system and are not saved to the system. If system is open game chat server in public internet, it must be able to filter spam, at least.

POSTCONDITION:

The system's condition is the same than before the test. There is no new chat messages.

REQUIREMENTS TO TEST:

Not mentioned in requirements

7.7 Recovery testing

Not applicable to this testing plan.

7.8 Performance testing

Not applicable to this testing plan.

7.9 Regression testing

Not applicable to this testing plan.

7.10 Installation and removal testing

Not applicable to this testing plan.

7.11 Usablity testing

Not applicable to this testing plan.

7.12 Special test cases

TEST CASE ID: T11

NAME:

Abusive user is banned from accessing the system by IP

PRIORITY:

High

PRECONDITION:

The feature is implemented into the system.

List of 5 banned IPs.

DESCRIPTION OF WORKFLOW:

Test environment is trying to access the system from banned ip.

EXPECTED RESULTS:

This test can't be made, because the feature will not be implemented

into the system. This is a security test case, but now a special test case, because we know already, that this system is not safe to run in public internet, as it lacks necessary features.

POSTCONDITION:

System is not accessed from banned IP

REQUIREMENTS TO TEST:

Not mentioned in requirements

TEST CASE ID: T12

NAME:

Abusive user is banned from accessing the system by username

PRIORITY:

High

PRECONDITION:

The feature is implemented into the system.

List of 5 banned usernames.

DESCRIPTION OF WORKFLOW:

Test environment is trying to access the system with banned username

EXPECTED RESULTS:

This test can't be made, because the feature will not be implemented into the system. This is a security test case, but now a special test case, because we know already, that this system is not safe to run in

public internet, as it lacks necessary features.

POSTCONDITION:
System is not accessed with banned username

REQUIREMENTS TO TEST:
Not mentioned in requirements

7.13 Acceptance testing

Not applicable to this testing plan.

8 CRITERIONS AND REQUIREMENTS OF TESTING

8.1 Acceptance criterions

Testing of the system will pass, if all test cases with high priority will pass the tests.

8.2 Disqualification citerions

Testing of the system will fail, if any of test cases with high priority will fail the tests at the end of the testing phase.

If any of the security tests fail, the application should not be runned in public internet, but may be used in local area network.

8.3 Suspension of testing

If over 50% of tests will fail, testing should be suspended. Developers will be notified.

8.4 Resumption of testing

When developers suggest, that system is in testable condition again, testing can be resumpted.

8.5 Finishing of testing

When all tests are made, and developers suggest that enough efforts are made to fix the system, in order to pass the failed tests.

8.6 Abandonment of code

When the code is not and is not going to be maintained, or if fixing the code properly would need too great efforts, then it should be abandoned.

9 RISK MANAGEMENT OF TESTING

There are at least following risks in accomplishing the testing plan:

1. Time table is too tight. Maybe some of the test cases can't be made properly enough.

2. Using planned testing environment (Jasmine) is too hard to get working properly with websockets and rest api , in a very small time, for a beginner.

3. There's not enough time try to fix defects found by the tests.

4. Specifications and requirements of the project may be incomplete, then it's questionable, are test cases planned correctly.

5. Project was made with applied agile methods, but assignments of project documents in the course, were maybe more from worlds of

waterfall project management methods. This may influence to this testing plan and process as well.

6. Force majeure events may happen anytime, that may confuse time tables of testing and reporting. For example, last week there was an autumn storm, that caused electricity and heating outage for 16 hours.

10 TIMETABLE AND WORKLOAD

Testing, that includes building testing environment, making tests and debugging and writing testing reports must be made in three working days, and take maximum 12 hours in total. Testing and all the related project documents must be ready until 7th oct 2015 23:59. Being late is strictly not allowed.