Test Summary Report

Spring Pet clinic

Elpida Atmatzidou, Nathan Lewis,
Joshua Lewis, Ned Kingdon, Nichola Ward

Testing purpose

The aim of this document is to report on the results of the test activities performed on the Spring pet clinic website. The tests performed have been broken down into three categories as follows.

Functional Testing of the presentation layer

Functional testing of the presentation layer has been carried out to ensure that the components of the website which the end user interact with function as expected e.g. when a user clicks on the home button they are taken to the homepage

Functional Testing of the business layer

Functional testing of the business layer ensures that the business layer of the website functions as expected. The business layer of the website contains the business logic which determines how the data can be created, stored and changed.

Non-functional testing

Non-functional testing techniques have been used to ensure that the website can handle the expected number of users without crashing or malfunctioning.

Application overview

The application requirement is for an information system that is accessible through a web browser. The users of the application are employees of the clinic who in the course of their work need to view and manage information regarding the veterinarians, the clients, and their pets. The application has the following functionality:[1]

- Viewing of a list containing the veterinarians employed by the spring petclinic and their specialities
- Viewing of the information of a pet owner
- Adding an owner
- Updating an owners information
- Viewing of the information of a pet

- Adding a pet
- Viewing of a pets visiting history to the clinic
- Adding to a pets visiting history to the clinic

Business rules

An owner may not have multiple pets with the same case-insensitive name. [1]

Scope

Items that are within the scope of testing

- The business layer of the spring petclinic website
- The presentation layer of the spring pet clinic website

Types of testing performed

Functional testing of the business layer

To ensure that all of the end points for each api were tested, tests were performed to assess the functionality of errors, owners, pets, pet type, root, vet specialities, users, vets and visits.

The create read update delete model (CRUD model from here onwards) was followed to ensure that the major functions of each endpoint were tested. The source code for these tests can be accessed at https://github.com/nedkingdon/spring-pet-clinic.

Problems encountered:

The functionality for /error and / (root) endpoint has not been made yet and therefore it was not possible to test for this functionality. The plan for testing the error functionality in the future is to follow the same CRUD model and ensure that every end point has been tested to work. This will include regression testing to ensure that no other classes and functionality have been affected by the further development of end points and tests.

Functional testing of the presentation layer

In order to sufficiently test all front-end functionality we seperated testing into 4 distinct test suites. The first suite focussed on the website's functionality in regards to the owners. This included Add and Edit functionality, where adding owners was tested both directly and indirectly via all owners page. The second suite concentrated on the testing of the vet functionality. This also included Add, Edit and Delete functionality, where adding vets was once again tested directly and indirectly via the all vets page.The last two test suites involved both Pets and Specialties functionality, where both suites where tested for Add, Edit and Delete functionality. The source code for these tests can be accessed at https://github.com/nedkingdon/spring-pet-clinic.

Problems Encountered:

The main problems during the testing of front-end functionality was mainly regarding the retrieval of specific elements. For instance, there was a lack of unique IDs and xpaths for certain elements within the HTML, this made it difficult to retrieve and test values. By giving each element a unique ID, the testing of the front-end functionality could be much simpler. These amendments would include regression testing to see whether any previous working functionality has been effected.

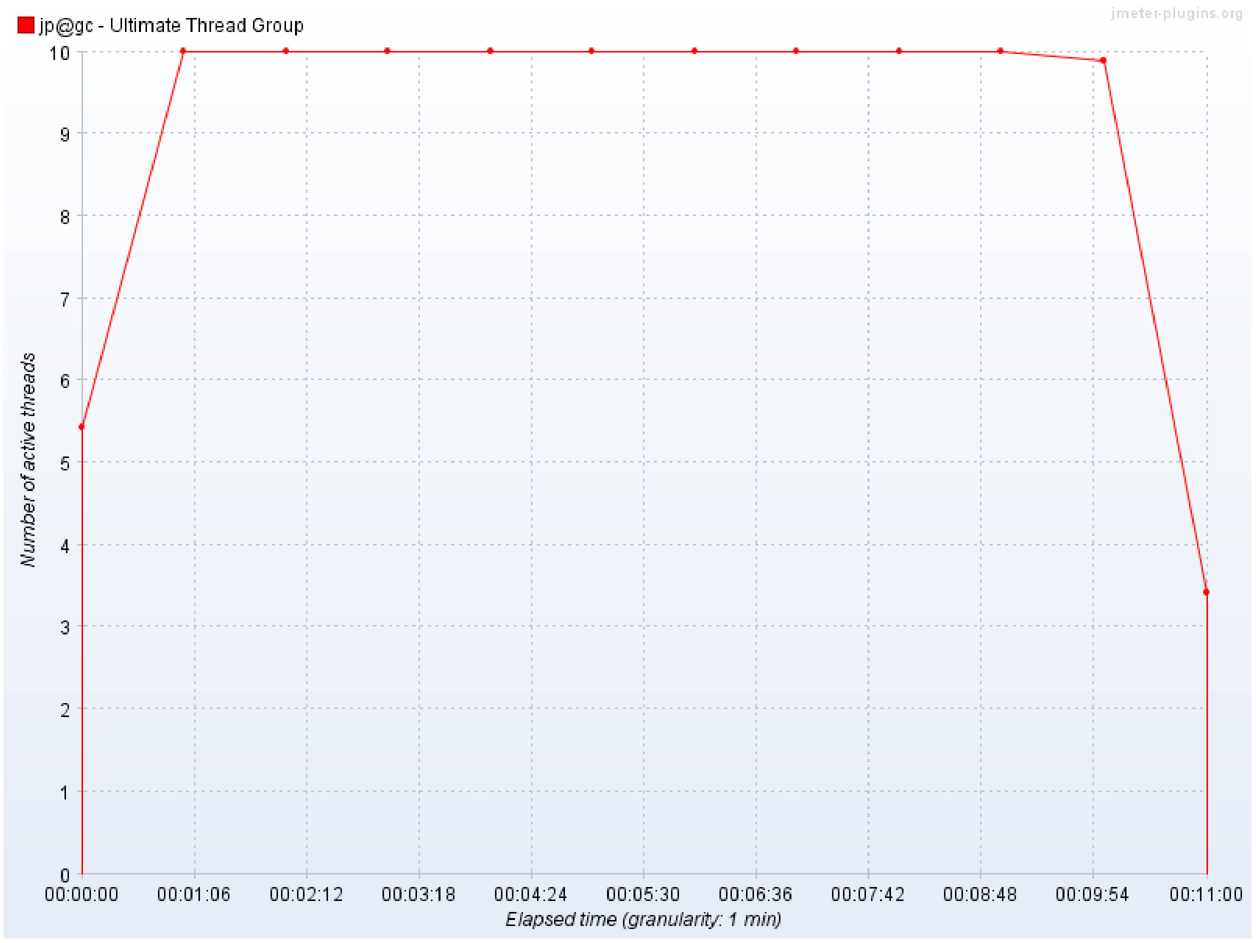
Non-functional testing of the business layer

Non functional testing of the business layer was carried out by performing three different tests: A smoke test to ensure basic functionality and absence of severe errors. A stress test to find the buckling point of the business layer and also measure the recovery and a spike test to measure the reaction when the business layer experiences an uncharacteristically high amount of traffic.

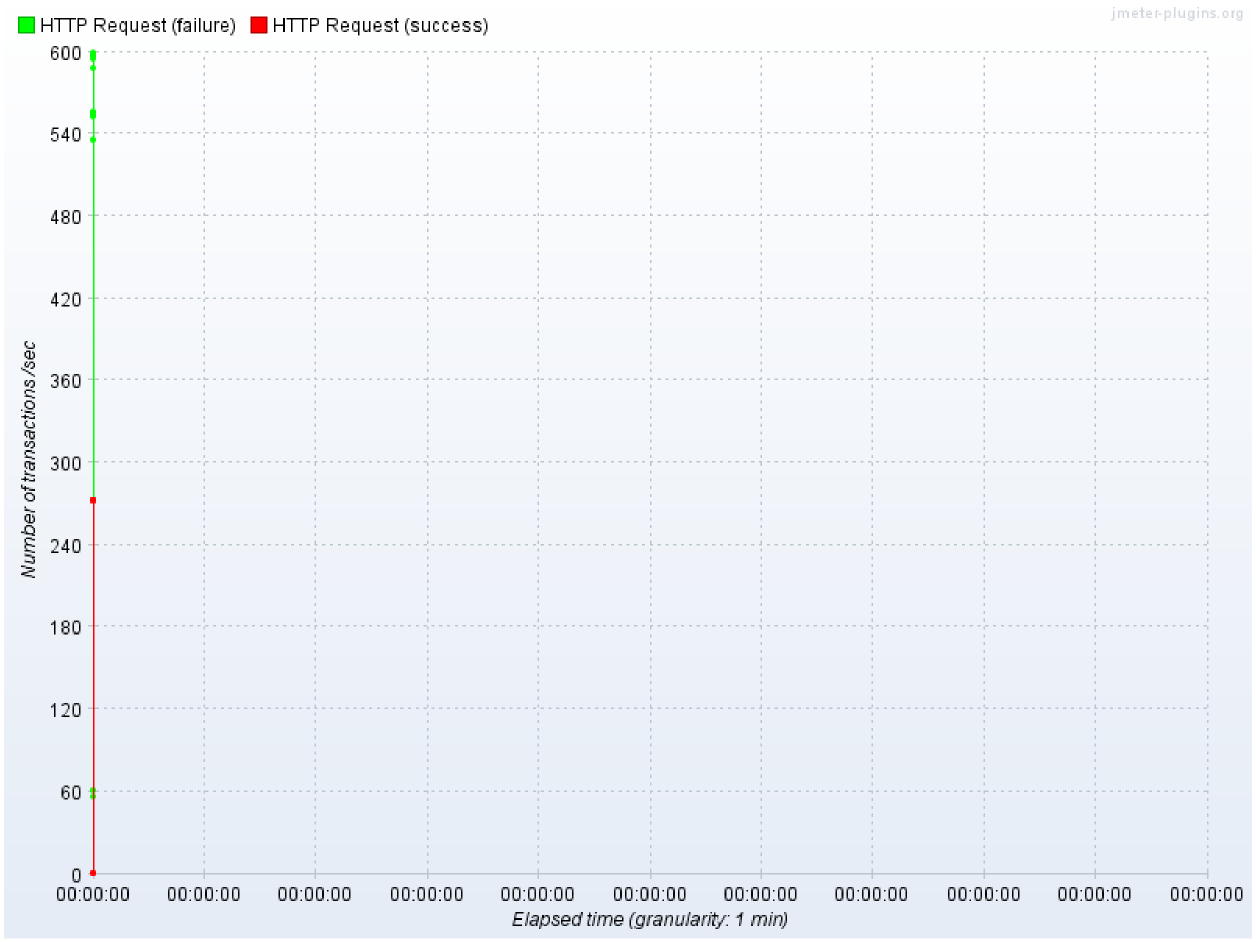
Results

Smoke test

The smoke test involved 10 active threads being held for eleven minutes to check for any severe errors.

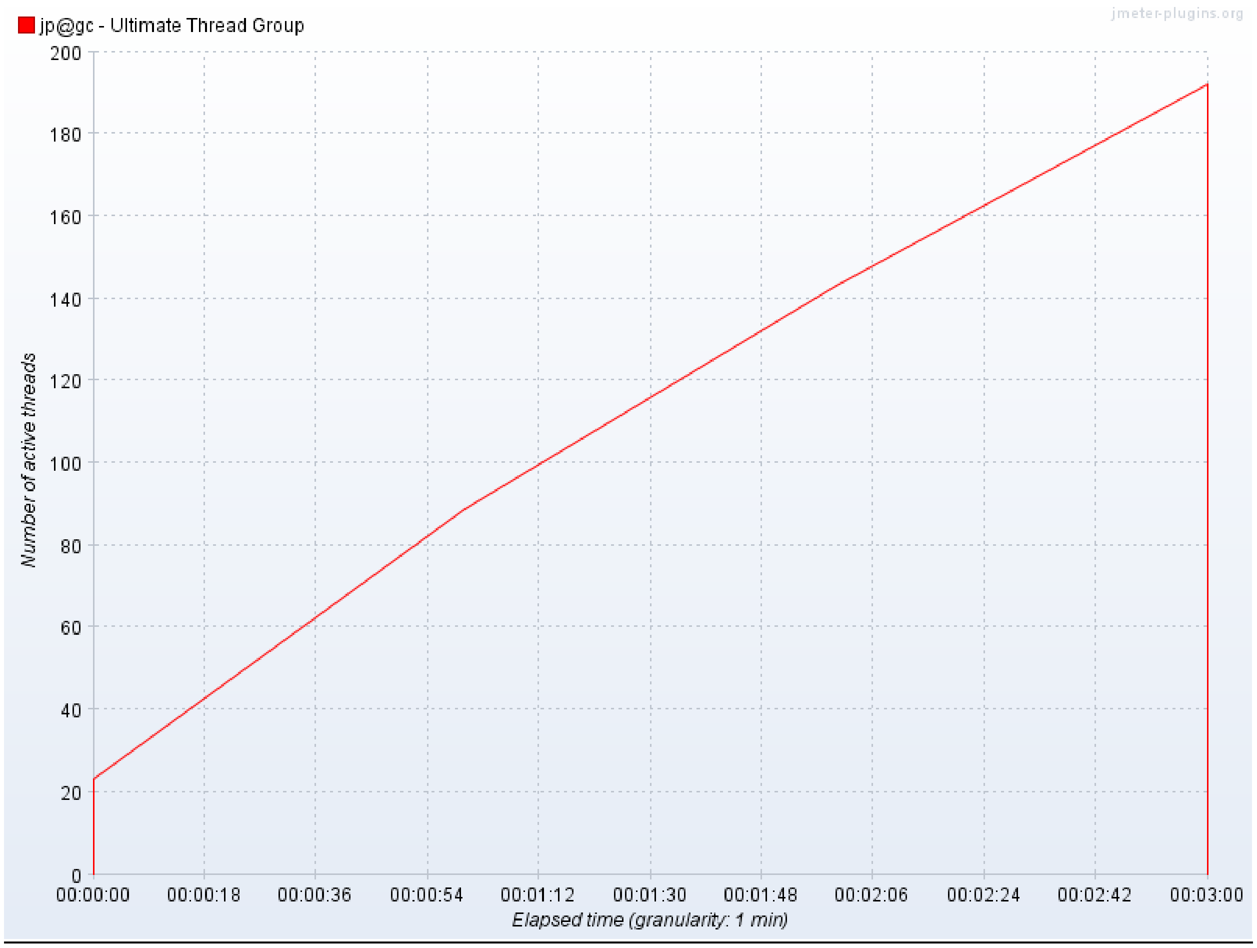


🟥 jp@gc - Ultimate Thread Group
jmeter-plugins.org
Number of active threads
0
1
2
3
4
5
6
7
8
9
10
00:00:00
00:01:06
00:02:12
00:03:18
00:04:24
00:05:30
00:06:36
00:07:42
00:08:48
00:09:54
00:11:00
Elapsed time (granularity: 1 min)

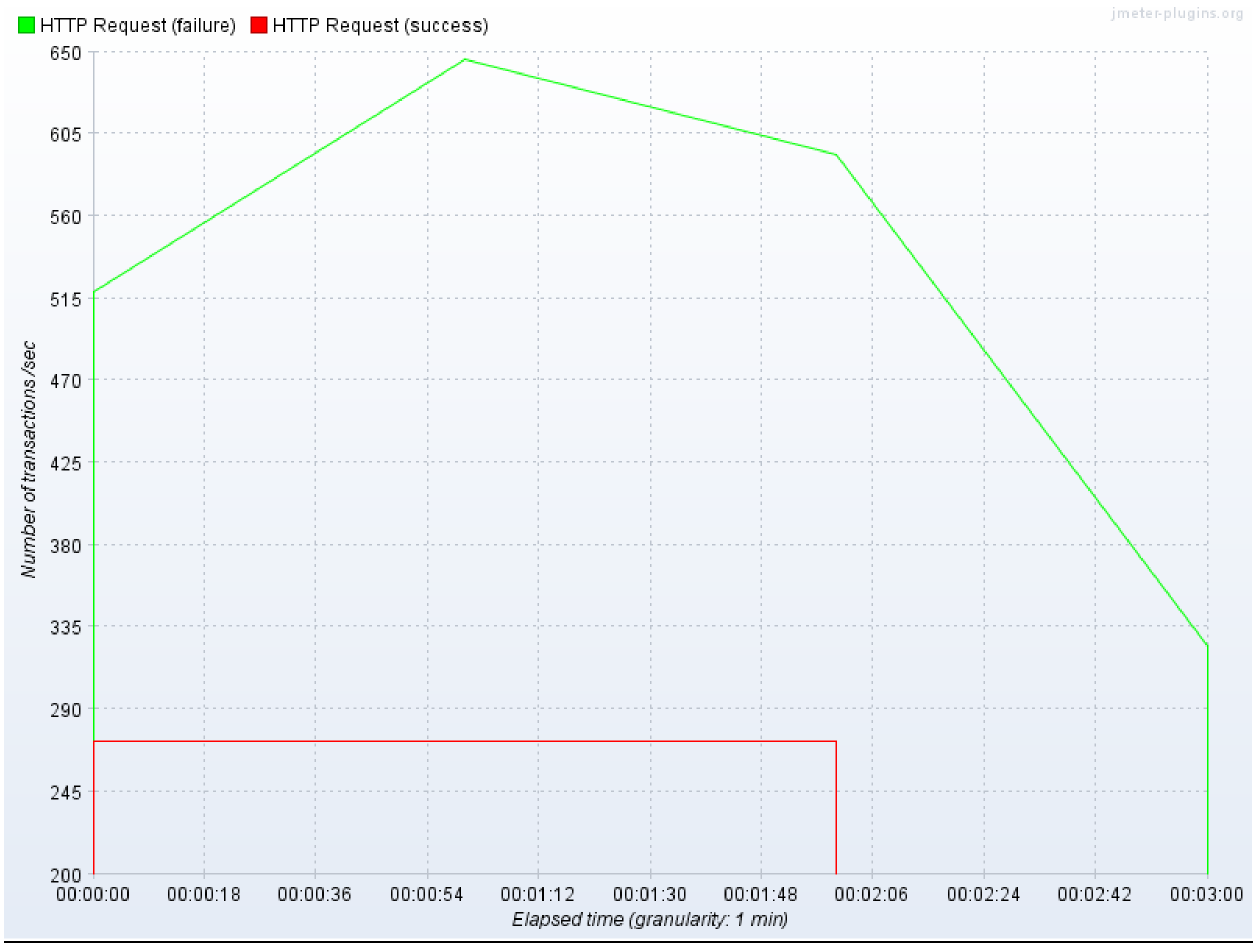


Stress test

The stress test performed involved increasing the number of active threads from 0 to 200 over 3 minutes. As seen in figures below This resulted in a number of failures throughout the test as shown in figures below.

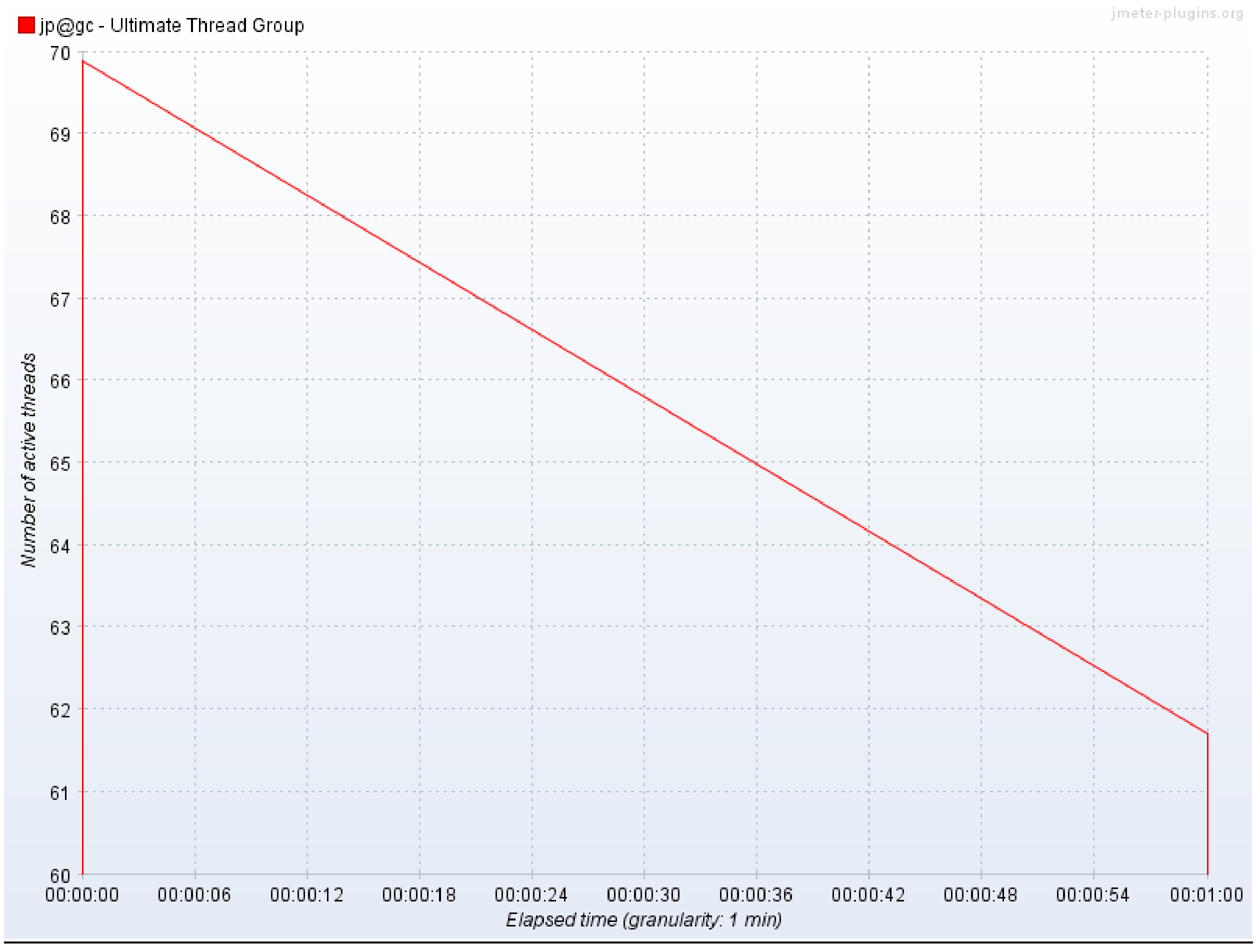


🟥 jp@gc - Ultimate Thread Group
jmeter-plugins.org
Number of active threads
0
20
40
60
80
100
120
140
160
180
200
00:00:00
00:00:18
00:00:36
00:00:54
00:01:12
00:01:30
00:01:48
00:02:06
00:02:24
00:02:42
00:03:00
Elapsed time (granularity: 1 min)

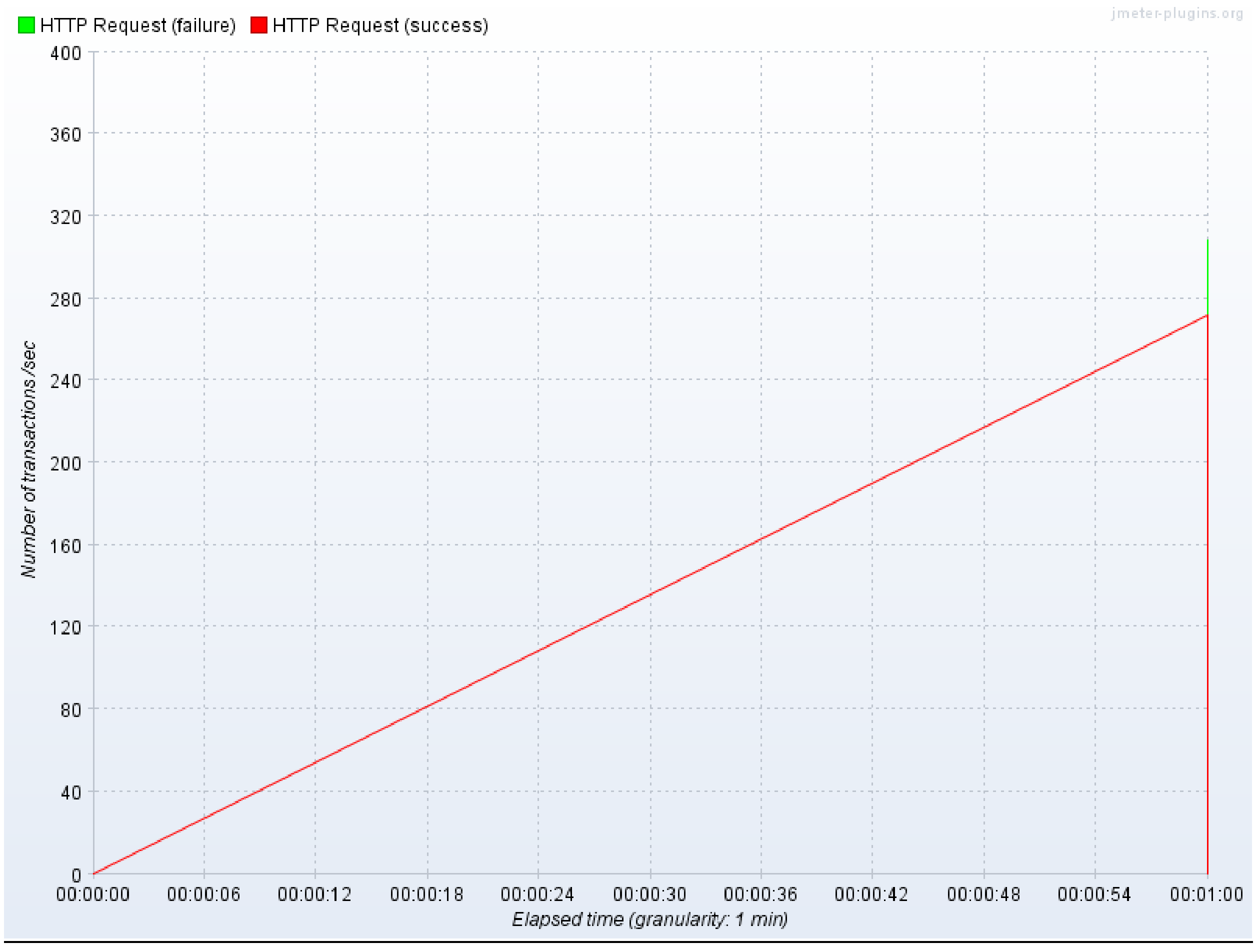


Spike test

The spike test involved holding a load of 1000 active threads for 10 seconds after which the load was reduced to one active thread for 30 seconds to check system recovery.



🟥 jp@gc - Ultimate Thread Group
jmeter-plugins.org
Number of active threads
70
69
68
67
66
65
64
63
62
61
60
00:00:00
00:00:06
00:00:12
00:00:18
00:00:24
00:00:30
00:00:36
00:00:42
00:00:48
00:00:54
00:01:00
Elapsed time (granularity: 1 min)



Test environment

The testing was split into functional, which tests how the software works and non functional; how well it works. Functional testing tests aspects of the presentation layer and business layer. The non functional testing checks that the software has the required functionality that is specified in the functional requirements. The front end functional testing environment used Selenium in eclipse to automate browser usage. The back end functional testing environment used restAssured in eclipse to test and validate restful endpoints. The non functional testing environment used Apache-JMeter which created a test suite in eclipse testing the load and performance capabilities in the application.

Conclusion

Testing activities focused on functional testing of both the presentation and business layers as well as non functional testing of the business layer. The functional tests focused on the CRUD method. These tests revealed that despite the business rules stating that it should not be possible to add a pet with the same name twice. This rule was not followed as it was possible to enter the same name multiple times. The non functional testing was somewhat inconclusive due to the instability of the jmeter testing software, however when spike test were performed the application did recover after 20 seconds

References

1. Use cases and business rules for spring pet clinic accessed at http://projects.spring.io/spring-petclinic/ on 20/12/18