CS202122-CSU33013 Requirements Documentation

“Generic Kubernetes based load testing framework”

Requirements Documentation

This document contains, through both diagrams and descriptions a listing of what functionalities we wish to implement in our testing framework.

Client | Client: Rapid7

Group 5 | Developers:

Second Years:
Holly Daly - 20331814
Declan Quinn - 20334565
Miguel Arrieta - 20332427
Abushark Abdelaziz - 20332134

Third Years:
Aaron Byrne - 19334098
Tomasz Bogun - 19335070
Vitali Borsak - 19335086

Table of Contents

1. Introduction

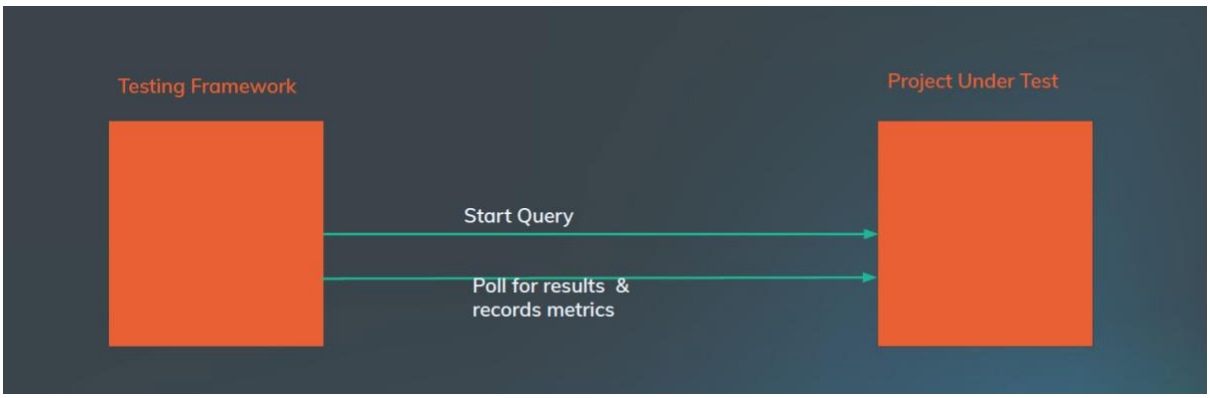
1.1. Overview - Purpose of system

This software is a search engine that searches vast amounts of data for specific information they need. The data that is being searched can reach terabytes in size, so it is crucial for the search engine to be as efficient as possible to speed up workflow. The proposed system will run performance tests on the piece of software and report back the results. In order to eliminate any external factors that can affect the performance, this testing will be run on an isolated Kubernetes Cluster. This system will be able to analyse the changes in performance of the different software release versions and be able to see if a new release has a negative or positive impact on the performance. The user will be able to query and poll the software under test using custom file formats and testing configurations.

1.2. Scope

The scope for this project is a configurable testing framework which can be used with Kubernetes and Docker Containers to run requests against HTTP API'S, time their responses and obtain results. The main goals for this project: ability to track changes in performance, reproducible for every software release and performance testing of a search engine. This project will include different components such as Test Runner, Sample Dataset, Kubernetes, etc.

Template of the whole process:



1.3. Objectives and success criteria

Our objectives are to split the work up into three sections and have groups working on each. One group will be working on a project under test using a sample database, another will work on a test runner and test runner configuration and the

last on Dockers and Kubernetes. Our project will be deemed as a success if we have a configurable performance testing framework, which can be used with Docker Containers and Kubernetes to run requests against HTTP APIs, time their responses, and obtain results.

1.4. Definitions, abbreviations

- HTTP - Hypertext Transfer Protocol
- API - application programming interface
- CPU - central processing unit
- Docker - A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.
- Kubernetes - Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.

2. Current system

Currently there is no system in place for performance testing that works with Docker Containers and Kubernetes that runs requests against HTTP APIs in an isolated test environment. Due to this the system needs to be created from scratch.

3. Proposed System

3.1. Overview

The main objective of the proposed system is to analyse the performance of a piece of software. In our client's case, this software is a search engine that searches vast amounts of data for specific information they need. The data that is being searched can reach terabytes in size, so it is crucial for the search engine to be as efficient as possible to speed up workflow. The proposed system will run performance tests on the piece of software and report back the results. In order to eliminate any external factors that can affect the performance, this testing will be run on an isolated Kubernetes Cluster. This system will be able to analyse the changes in performance of the different software release versions and be able to see if a new release has a negative or positive impact on the performance. The user will be able to query and poll the software under test using custom file formats and testing configurations. Our client has asked us to deploy the project under test to MiniKube which is a local Kubernetes cluster. The system will start off as a

command line program, but if time permits, may also have a graphical user interface. After discussion with our team, we decided that it will be best to use either python or java for this project as our team is familiar with these two languages.

3.2. Functional Requirements

- Performance testing application accessed through a command line interface.
- Should include time testing and if constraints permit, testing of:
    - CPU utilisation
    - Memory utilisation
    - Disk utilisation
    - Network utilisation
- Ability to keep track of changes in performance, for example in different versions of the software.
- Needs to be reproducible for every software release.
- Provides tests in isolation, through the use of a separate Kubernetes cluster.

3.3. Non-functional requirements

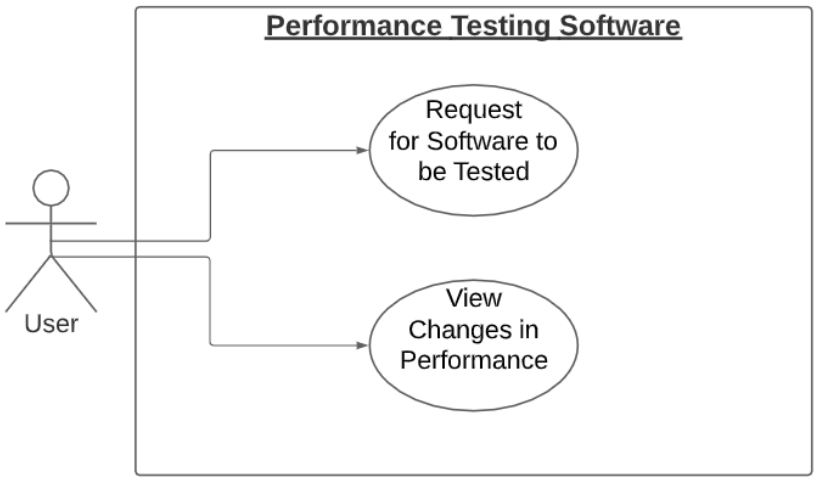
- Ease of use. The project should make it easy for the client to test the project under test within a Kubernetes cluster and obtain and view the results.
- The project should be easily scalable to fit the needs of the client's actual project they want to test.Include the possibility of automating the expansion process to add in more docker containers to improve performance of the test runner.
- The project under test has to perform well as it needs to handle files with millions of lines of text and search through them.
- Our client's project is a long term project and requires this test runner to be easily maintainable and portable (Hence docker and Kubernetes).

3.4. System prototype (models)

3.4.1. User interface mock-ups

A user interface is not required, as a command-line interface will suffice.

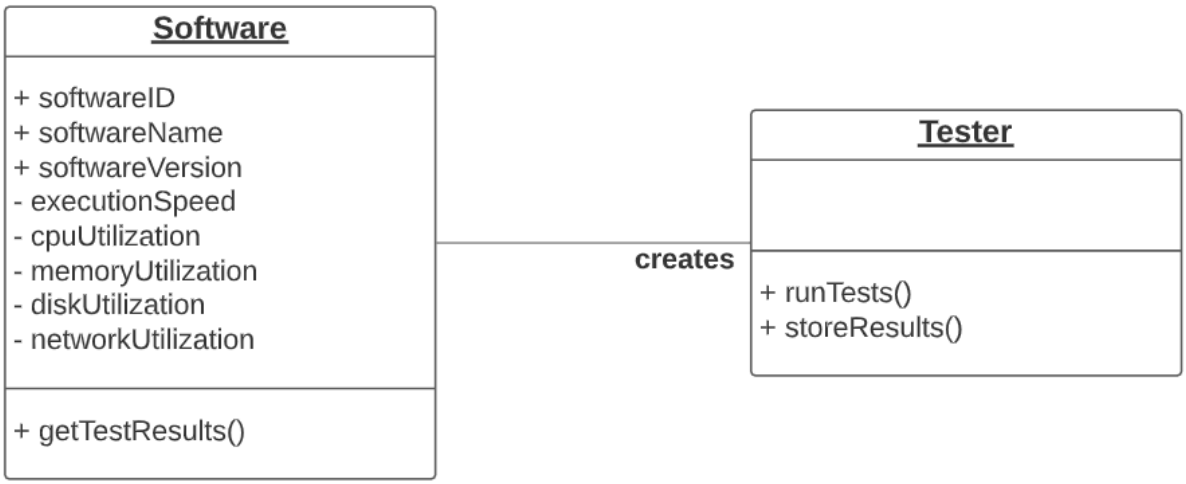
3.4.2. Use cases (including text narratives)



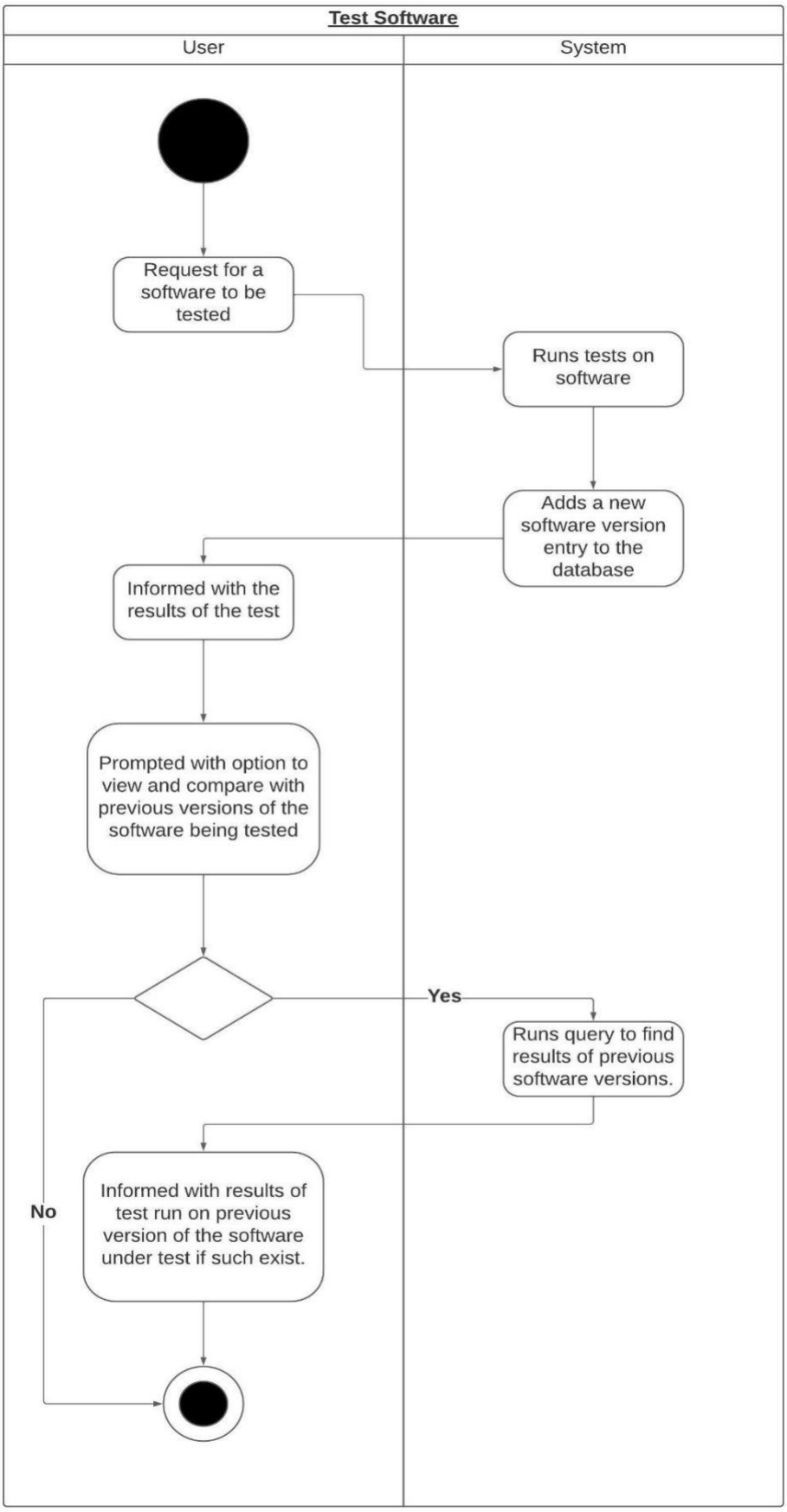
Request Software to be Tested	User	System
1.1	Request for a software to be tested for its efficiency	The system runs test on the software with various data.
1.2		The system returns the test results to the user and stores them.

Changes in Performance	User	System
1.1	Request to view and compare the test results of previous versions of the software under test.	The system checks if there is any test data on previous versions of the current software under test.
1.2 (a)		The system returns the old test results and compares them with the most recent test results for the software under test.
1.2 (b)		The system returns null as there is no test results for the software under test.

3.4.3. Object model

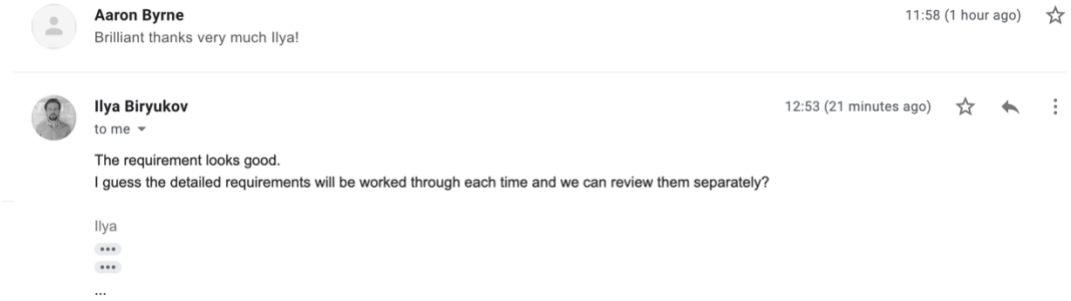


3.4.4. Dynamic model



4. Proof of client sign off:

Client signed off on the document and we have organised weekly meetings to review progress.



Aaron Byrne 11:58 (1 hour ago)
Brilliant thanks very much Ilya!

Ilya Biryukov 12:53 (21 minutes ago)
to me

The requirement looks good.
I guess the detailed requirements will be worked through each time and we can review them separately?

Ilya
...