Generic Kubernetes based load testing framework

Group 5

Client: Rapid7

2nd YEARS

Holly Daly - 20331814

Declan Quinn - 20334565

Miguel Arrieta - 20332427

Abdelaziz Abushark - 20332134

3nd YEARS

Aaron Byrne - 19334098

Tomasz Bogun - 19335070

Vitali Borsak - 19335086

Overview

- Create a mock search engine that searches vast amounts of data for specific information they need.
- The proposed system will run performance tests on the piece of software and report back the results.
- The test runner and project under test must be packaged up to a docker container and deployed to a Kubernetes cluster.
- Monitor and graph the performance of the system.

The team and roles

Project under test:

- Tomasz Bogun - 3rd Year
- Declan Quinn - 2nd Year

Docker and Kubernetes:

- Aaron Byrne - 3rd Year
- Abdelaziz Abushark- 2nd Year

Test Runner and Configuration:

- Vitali Borsak - 3rd Year
- Holly Daly - 2nd Year
- Miguel Arrieta - 2nd Year

Overview of the proposed system

- The project under test will be made with python and the server module Flask, along with multiprocessing.
- We will use MiniKube for our Kubernetes deployment.
- The project under test, test runner will be run on separate containers.
- The project under test will search through a large downloaded text file.
- The test runner will be a command line program and will also be made with python.
- It will use custom designed configuration commands.
- The performance will be tested using Kubernetes observability.
- If time permits, the project under test will also be able to be autoscaled using Kubernetes, and there will have a graphical interface for visualising the performance of the project under test.

Objectives and Success Criteria

- Our objectives are to split the work out into three sections as stated before.

- We can deem our project 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.

Requirements

Functional

- Performance testing system with a command line interface
- Should test time, CPU, memory, disk and network utilization.
- Keep track of these performance measurements through different software versions
- Provides tests in isolation, through the use of a separate Kubernetes cluster.

Non- Functional

- The end result should make it easy for our client to test their desired software
- Project under test needs to perform well as it searches through terabytes of data.
- Kubernetes must be easily scalable to test the clients project

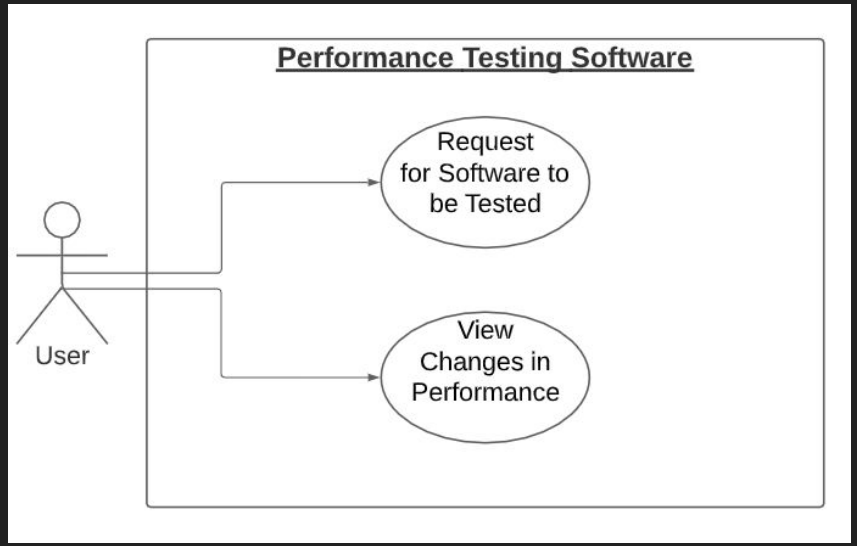
Current System

- Rapid7 have no current system in place for analyzing HTTP API’s in an isolated testing environment

- Components such as the Test Runner and Project Under Test are to be built from scratch

- We have not been given a sample dataset to work with so our testing will be done on data we source ourselves

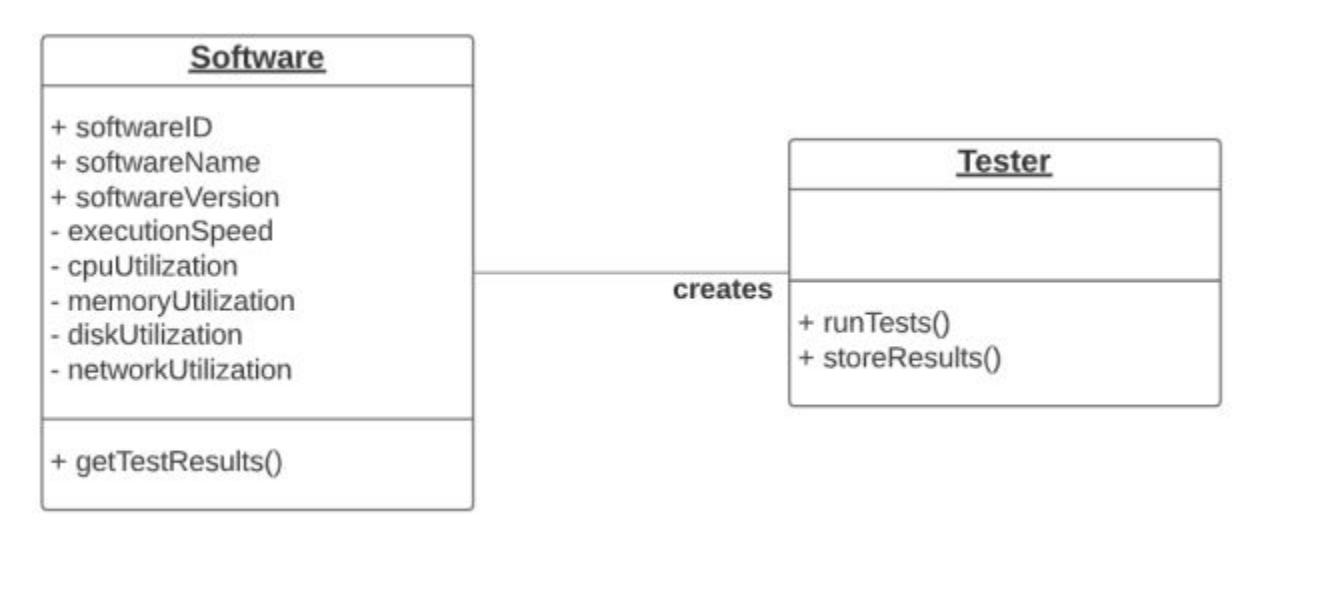
System Models - Use Case



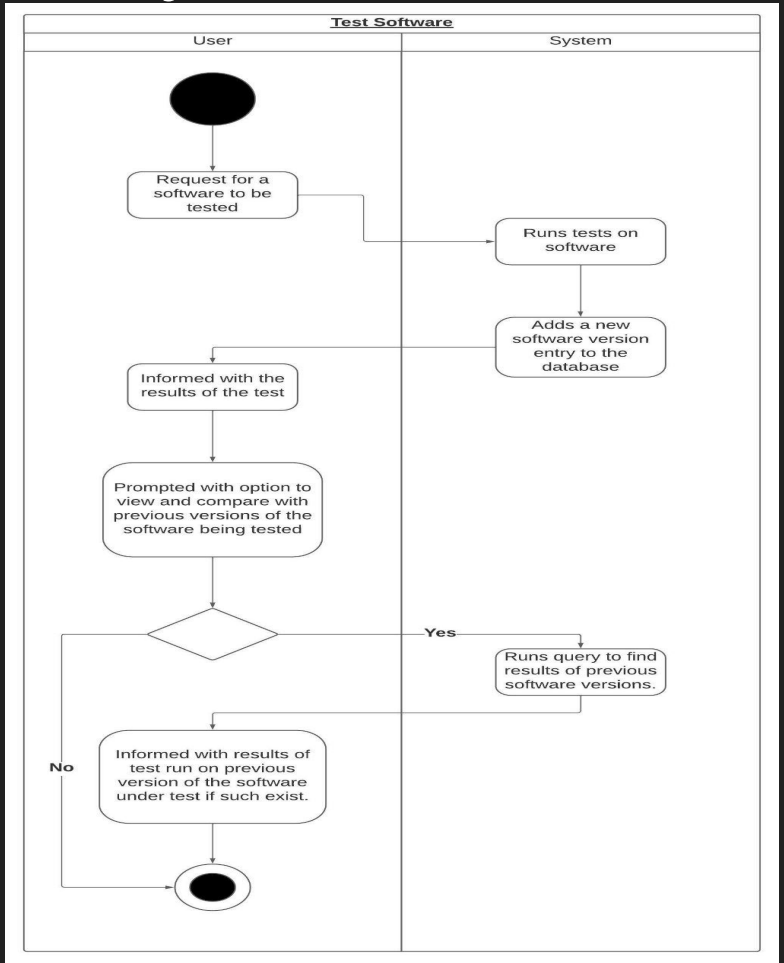
Request Software to be Tested	User	System
1.1	Request for a software to be tested for its efficiency	The system runs test on the software with various data.
1.2		The system returns the test results to the user and stores them.

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.

Object Model



Dynamic Model



Questions?