“Generic Kubernetes based load testing framework”

Trinity College Dublin

The University of Dublin

Software Design Specification Document

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

1. Introduction

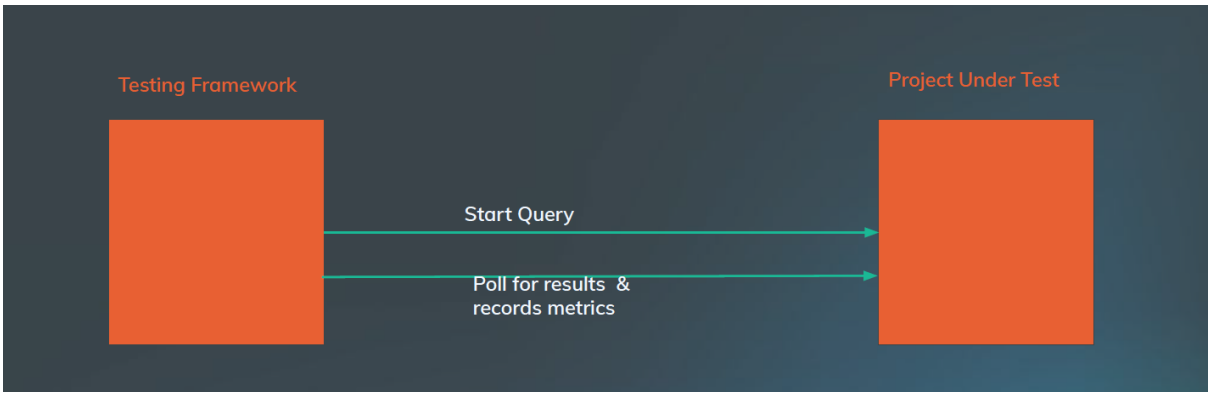
1.1. Overview - Purpose of the System

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 analyze the changes in the 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 that can be used with Kubernetes and Docker Containers to run requests against HTTP APIs, time their responses, and obtain results. The main goals for this project: the ability to track changes in performance, be 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. 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. System Design

2.1. Design Overview

The system has multiple components that can be worked on together in multiple separate teams. There are three sub-teams in our group with 2-3 members each.

The first component is the project under test. This is the piece of software whose performance will be measured. The project under test can be any piece of software, but our client has requested that we make a specific software for testing. It will be an HTTP server that will search through a large text file and search for a given string and report back the results.
The HTTP server will be able to accept a POST and a GET request. When a POST request is received, the server will begin to search for a string provided in the body of the request, in a file whose name is also provided in the body of the request. Then, the server will begin the search and send back a unique ID. This ID can be then used to view the results of this

search using a GET request. When this happens, the server will report the number of matches found, along with the status of the search. The client will be testing internal software that resembles the functionality of the project under test that we will develop.

The second component is the test runner and test runner configuration. This is the software that will perform the tests on the project under test. The test runner will be a script that sends commands to the project under test, waits for responses, and records the results and performance. The test runner will be able to take in a configuration file as input called the test runner configuration. This file will contain all the commands that will be run by the test runner. This means that the test runner will be able to run different tests based on the configuration file that is provided.

The third component will be packaging up the above components into docker and Kubernetes containers so that they can be run in isolation. A dockerfile will be required for this that initializes the containers. The containerization will also allow for the performance monitoring of containers using built-in functionality. This component will also involve an auto-scaling feature for the project under test that will allow it to use all of the resources it has access to efficiently.

2.1.1 High-level overview of how the system is implemented, what tools,

frameworks and languages are used etc.

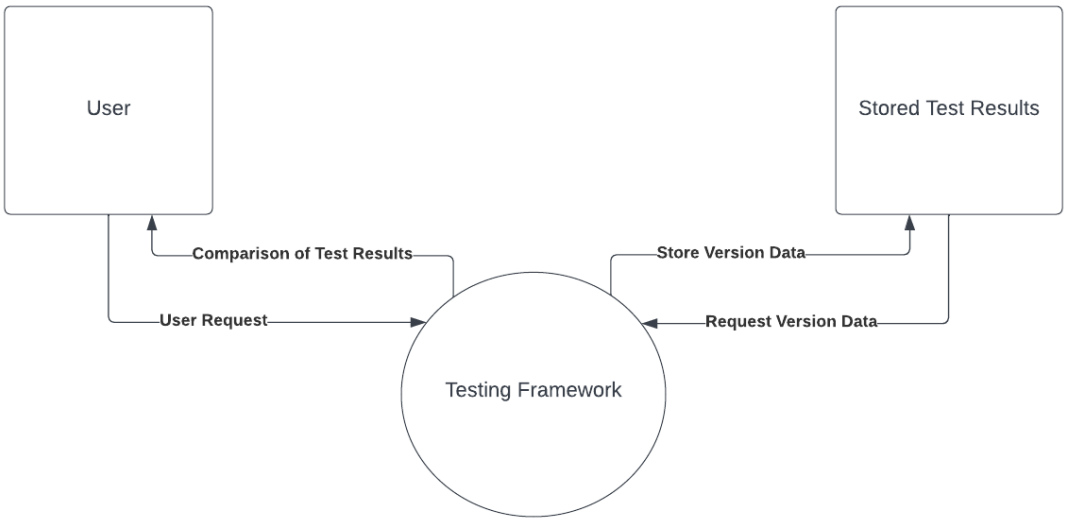
The main objective of the proposed system is to analyze the performance of a piece of software. In our client's case, this software is a search engine that searches vast amounts of data for specific information they need. The data that is being searched can reach terabytes in size, so it is crucial for 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.

- ➔ **Programming Languages and Framework**
- ◆ The project will be built using:
- Python
- Java
- Dropwizard
- Flask
- Docker
- Kubernetes
- ➔ **Version Control**
- ◆ Git and GitHub are used for version control.
- ◆ Jira will be used for workflow management
- ◆ Jenkins will be used as our CI/CD platform

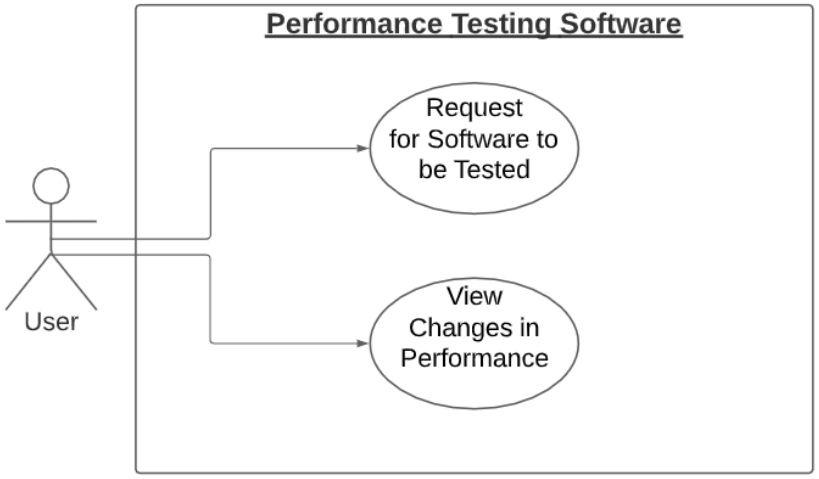
2.2 System Design Models

2.2.1 System Context

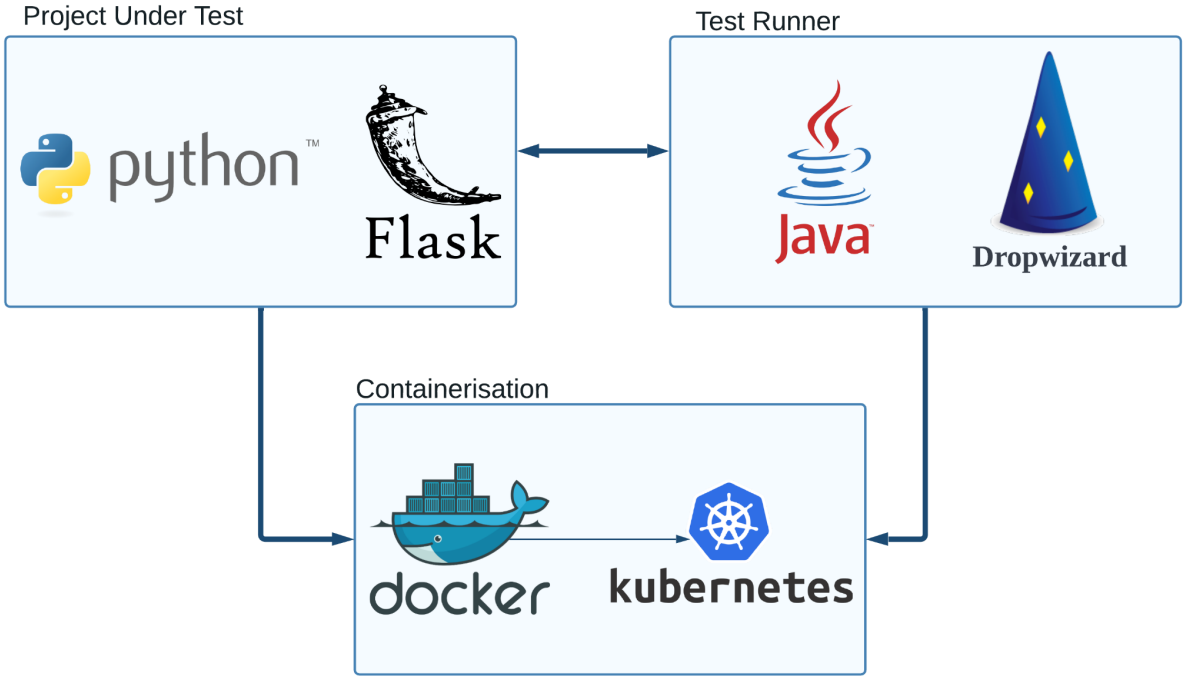
Our system allows the user to send a request to test a piece of software. This is tested and the results of the current version are saved. The user is then returned the test data along with information on how previous versions of the software performed under testing if requested.



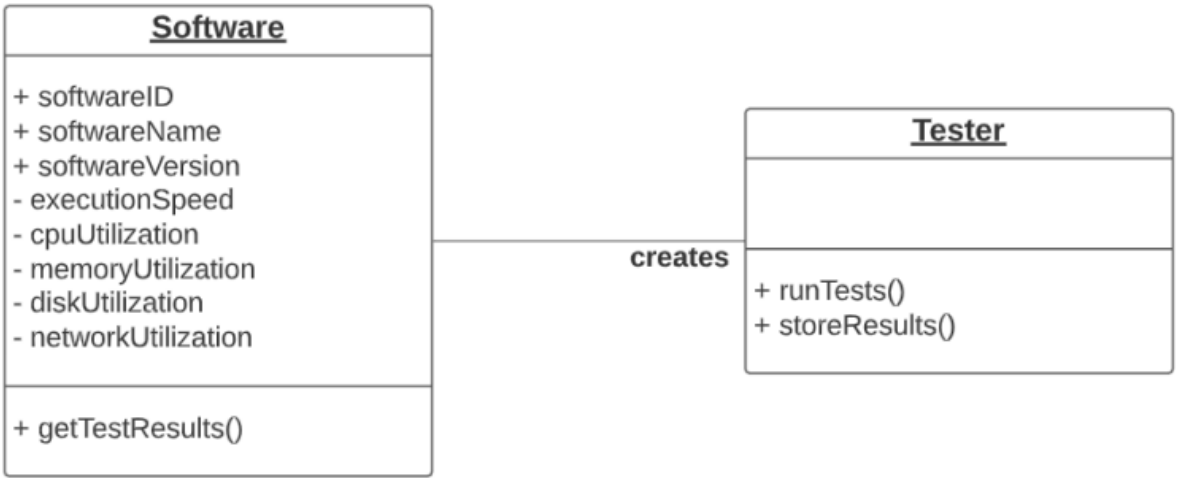
2.2.2 Use Cases



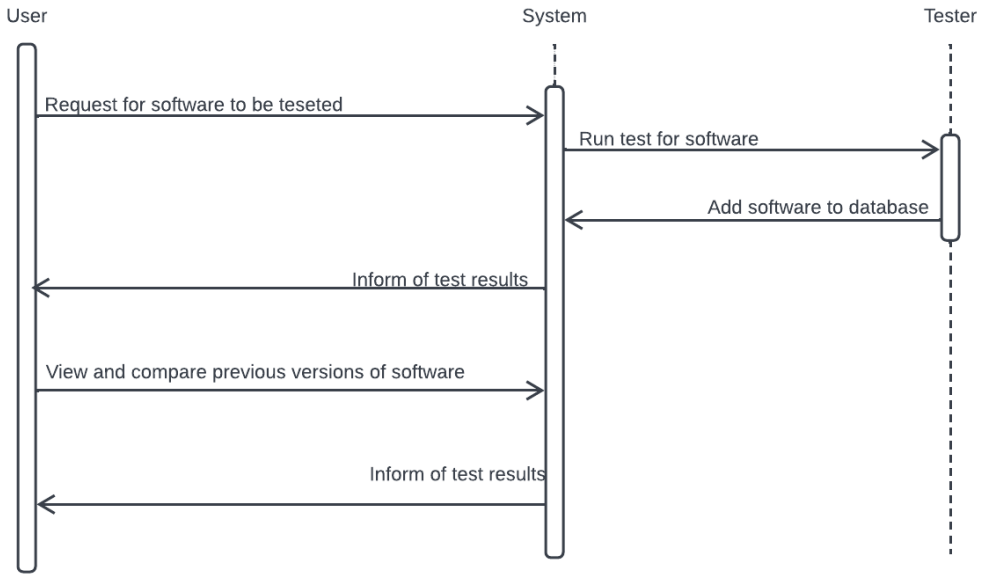
2.2.3 System Architecture



2.2.4 Class Diagrams



2.2.5 Sequence Diagrams



2.2.6 State Diagrams