Group Portfolio - Project Initiation Document

(PID)

Industrial Consulting Project (COM617)

Aeryk Del Mundo

Afonso Marques Mendes Cabecadas

João Estrela Leite

Rui Coelho Pinto

Tebogo Dube

Support Tutor: Craig Gallen

Project Sponsor: Website Monitoring – OpenNMS Group

Contents

PROJECT SUMMARY & INTRODUCTION 2

REQUIREMENTS | SCOPE AND EXCLUSIONS 3

MONITORING AND EVALUATION.....4

IMPLEMENTATION | RESULTS.....6

CONCLUSIONS AND RECOMMENDATION.....9

REFERENCES 11

APPENDICES..... 13

Figure 1: System Diagram representing all three main components and their containers 13

Figure 2: Main features of the Use Case Diagram 14

Figure 3: System concept Robustness Diagram 15

Figure 4: 3 Project Sprints..... 15

PROJECT SUMMARY & INTRODUCTION

When a large website has become faulty it can cause multiple sources of failure, for example, a website could have server overload, this occurs if the website has too many users on at the same time therefore it fails. Additionally, the browser could fail to load or have a very slow loading time and cause a negative impact on user experience (Cummings, 2018).

Our team has been involved in developing a software solution that can be used to monitor the user interface and reliability of the company's website automatically. We created a prototype solution that can be used to run simulated operations against the website and report on the results. We have used tools such as JMeter, InfluxDB and Grafana throughout this project; JMeter for performance, spike, and unit testing to determine the speed and responsiveness of the website, InfluxDB which is a time-series database written in Google's programming language "GO" that is optimized for storing and retrieving time series data, lastly, Grafana is being used to visualize, alert on, and explore metrics from any location, it enables us to create a variety of graphs and visualizations that come from InfluxDB as the test runs, this platform is using docker containers and it is hosted in Azure (Nigam, 2019; Gillis, 2019; Sarawagi, 2019).

REQUIREMENTS | SCOPE AND EXCLUSIONS

Core Functionalities - *version: 1.0.0*

Version	Requirement Text
1.1.1	Jmeter should be used to create a Test Plan that produces results, this will be the default test plan, which every user will be able to use and adjust with parameters
1.2.1	Jmeter Test Plan Results should be sent to Influxdb
1.3.1	Should be able to run Jmeter Test Plan within Java
1.3.2	The test plan should be able to continuously create results from the same Test Plan until the program is stopped
1.3.3	Parameters should be set in the Jmeter test plan and should be changeable within Java. This parameters should cover: - ThreadPool - Ramp up time - Target Url
1.4.1	Results will be displayed in Grafana for visualization

Network - *version: 2.0.0*

Version	Requirement Text
2.1.1	All three major components should be booted up with Docker-Compose - Jmeter Application running in java - Grafana - Influxdb database
2.1.1	Once the docker compose file is up the system should create results and display them in Grafana until the docker-compose file is termitted
2.1.2	Data should also be stored on the server, and kept until deleted
2.2.1	The service should run on a live server and be accessible via the web

User Friendly - *version: 3.0.0*

Version	Requirement Text
3.1.1	A user interface should be created to allow users to interact with the system and change parameters inside the Jmeter Application.
3.1.2	The changes in the Jmeter Application should then reflected in Grafana
3.1.3	Validation should be added to ensure correct user inputs
3.1.4	This user interface should be a web site and should also be booted by the docker compose file
3.2.1	Results in Grafana should be visible to the user after submitting a form which adjust the parameters in the Jmeter Application
3.3.0	Users should also be able to submit their own test plans and view the results of that test plan the same way as the default test plan

User Authentication - *version: 4.0.0*

Version	Requirement Text
4.1.1	Users should be able to create accounts on the system and should have to login to be able to use the system
4.1.2	Different user accounts should have their own databases where they can store their own results
4.2.1	Each of the results in Grafana should be tailored to that users stored results in Influxdb
4.2.2	Specific results in Grafana should only be accessible by the user who created them

User Interface - *version: 5.0.0*

Version	Requirement Text
5.1.1	Graphic interface should be styled to offer better user experience
5.2.1	A help page should be created to help users, use the System

MONITORING AND EVALUATION

The perceived load time, being one of the most critical features for users, websites need to have constant good performance to catch the attention of as many users as possible. Most websites could benefit a lot from web monitoring as even some businesses see a direct correlation between their website performance and their profitability due to increased customer satisfaction levels.

We have developed a synthetic transaction monitoring (STM) system that provides the ability to test and visualize a website's availability and performance. STM work by simulating end-user experience in a controlled environment, some of its advantages over other types of monitoring systems are, (1) a more accurate representation of general website performance, (2) can be used to test a website before release or update and (3) allows for the manipulation of test plans. The manipulation of test plans was included so we could dynamically test our artefact as well as to allow users to customize test settings very quickly, giving them the liberty to run constant monitoring as well stress tests against their website (Cito et al. 2015).

The system can be accessed through our website where users can edit settings about the test plan and view the data in real-time, this is being done with Grafana using a public dashboard modified for our needs. For our visualization, we are using a combination of graphs with time-series data and some statistics of requests information. The graphs show several important metrics for performance testing such as transaction time or networks traffic and clearly display the website performance and availability over time. The statistics panels can be used to gain further insight into the requests being made from your website.

In the initial design, we wanted to use Grafana as our main interface for the application because the tool supports several functionalities and external plugins that could be used for our purposes. One of them would be to support different users simultaneous because we moved to an external webpage for the interface, this idea was no longer viable and implementing a login system from scratch would be too time-consuming at that stage. Another point and the reason we had to move to a separate page was due to the panels in Grafana, our first idea was to create our form panel that would communicate the user input to our application. Creating a panel turned out to using completely different technologies and after some weeks of research, we had to drop it and work with the tools we are familiar with.

We have tested our project against several websites to see noticeable changes in the graphs. Our approach was to monitor before and after new releases to the server to see what panels of our solution show differences. Depending on the magnitude of the update or complexity of a faulty component, the graphs should show a direct correlation between size and request loading time, if the results are clear our solution is monitoring as intended.

IMPLEMENTATION | RESULTS

The project was designed to be a cloud-hosted web application using three main components with a simple UI where a user can view and interact with a test plan. As shown in the Figure 1 diagram, the three main components of the project are JMeter, Grafana and InfluxDB.

• JMeter - An open-source software, built with java, designed to perform several different tests, such as load tests, performance tests and functional behavior, on the web application. JMeter is the main component of the application and it provides the logic to run the tests and provides a REST API that allows a user to modify the said test.

• InfluxDB - A time-series database optimized for fast storage in fields such as monitoring, IoT and performance metrics. The connection between JMeter and influx was done through an existing backend listener specifically built to feed data gathered from a test to influx database.

• Grafana – An open-source analytics interactive web application. Grafana was the application used to provide a visual representation of the data stored in the database. Grafana is a flexible platform, developers can build their own panels or plugins depending on their specific needs.

During the designing phase, the group agreed on the conservative scope for the project. The project consists of two main use cases shown in Figure 2 such as, using a simple HTTP request to check the availability of the website every however many seconds and uploading a specific test plan developed by the user. With these two features, the platform provides an entry point for a more junior developer with quick test and a more complex route for more experienced developers.

The robustness diagram shown in Figure 3, is a visual representation of how the data and requests are flowing through all the components. As all the main platforms have already built-in protocols to ease the interaction between them the part where we had to build logic was the API from the web application for visualization and the JMeter running on the backend. Implementation Complications At the implementation phase, the project got narrowed into two routes of how we would be able to integrate user input to change parameters in the test plan and upload a file with a custom plan. The main idea was to build a dashboard containing panels that would reflect a general view of metrics and a panel that a user would be able to input information to change the test plan. Grafana allows this by creating panels using a framework called react.js and a variant language of JavaScript called TypeScript. Some issues regarding the development of the plugin caused the team to pivot in favour of the second approach, which was building a separate front-end UI that would serve as the point of contact with the user to get data and to display data from Grafana. This approach consists of a simple HTML and CSS page with a form that takes user input to change parameters in the JMeter class running in the same instance. Implementation Pivot

Our team decided to go back to using an HTML page with a form to gather necessary data and embedded graphs from the Grafana dashboard. Users will use the form to change any available parameters. When submitting, the data is passed to the HTTPserver.java class. Inside that class, the scripts open a connection with a port of 5008 which will then take the gathered data and pass it through a loop, after the data is passed to a scheduler to store the data in a task. The same data is passed to a class called PerformanceTest.java, this class is the class that will run JMeter with the data gathered from the user form. It will use the data to change the parameters in the .JMX file and running the test afterwards.

PROJECT ORGANISATION STRUCTURE | PROJECT MILESTONES & MANAGEMENT

The project adopts an agile methodology and is ran against three sprints that were provided at the beginning of the unit, shown in Figure 4. There are 3 total sprints, the first two sprints are approximately 3 weeks each and the final sprint is 1 month-long. Throughout each sprint, many tasks must be accomplished and Agile provides a structured and dynamic way to organize and distribute the work, this work is organized weekly and is achieved by considering the following Agile concepts:

- **Working over documentation:**
  The main goal of the first sprint was to define requirements for the project, after accomplishing this, the team documented these requirements as various types of diagrams that served as the base of future development.

- **Working software:**
  An agile development encourages small and consistent working software.

- **Adaptability to change:**
  Organizing the work weekly, allows the team to welcome any unforeseen change that needs to be implemented on the project, these unexpected changes would be discussed after the meetings with Craig and considered for the following week.

The second sprint's main goal is to create an Initial Proof of Concept, which aimed to prove the feasibility of this project. To accomplish this, the documentation created in the first sprint was updated and improved based on the continuous research that was carried out, various experiments were developed that aimed to test the various tools that would be used and how they communicate with each other and would work collaboratively.

The third sprint aims to build a minimal viable product that represents a version of the project that is usable but can be further improved to a higher level. For this, we

further implemented the project based on the requirements and the documentation created before to have a goal of a working application that would satisfy the requirements.

Throughout the development of the project, the GitHub repo kept track of all the implementations and contained a weekly Kanban board that kept track of the completion status of all the tasks related to the project and Wiki pages that explain concepts and tools related to the project, finally, there is also a GitHub Actions script that aims to automate some functionalities of the project and re-apply some DevOps concepts learned in previous units (GitHub, 2021).

Project organisation structure:

The project had a project manager and a team leader that structured and ran the meetings; and every week, all the members of the team would get together then go through the project tasks that needed to be completed, the tasks would be assigned collectively considering individual confidence/motivation and who would be the best fit for the task at hand. This happened throughout the entirety of the project and logs of the authors of all the tasks were kept on Discord.

CONCLUSIONS AND RECOMMENDATION

Conclusion:

The project ran smoothly, with two meetings held each week, on Tuesdays and Thursdays. During the design phase, our group decided on two possible solutions for the implementation. (talk about the two solutions) Both ideas were explored, but due to the learning curve involved with using Grafana, it was decided that we would simplify the front end of the project by using an HTML page with embedded graphs. With one member focusing on the project management, and another focusing on leading the team throughout the project, this allowed us to quickly tackle any issues

that would occur when developing the app so that our team would not have to worry about things such as miscellaneous tasks, although important, such as the creation and management of the GitHub repository, overall timekeeping, and task management. The leader of our team could focus on the scope of the project and implementation alongside the rest of the team. Overall, the project was successful, as the minimal viable product was achieved with the main intended feature of changing a test plan by using user input.

Recommendations:

This project could be further improved, this could be done by implementing the remaining features that were discussed during the designing phase, such as, allowing a user to upload a custom test plan and possibly storing that plan for future usage if intended. For this approach, a user onboarding would also have to be implemented and verification would be needed to make sure there are no security violations such as, a different user using the test plan that is not his/hers. Another approach to this would be to use Grafana as the host and develop a plugin for the platform, as it offers a flexible way of doing what the MVP does, without having to worry about user onboarding as Grafana already has it. By using this approach, it would also solve a problem for future scalability.

REFERENCES

BECK, K. et al., 2019. Manifesto for Agile Software Development [online] [viewed 22 Feb 2021]. Available from: https://agilemanifesto.org/

CITO, J. et al., 2015. Identifying Web Performance Degradations Through Synthetic And Real-User Monitoring. Journal of Web Engineering [online], 14(5-6), 414–442 [viewed 7 Apr 2021]. Available from: https://journals.riverpublishers.com/index.php/JWE/article/view/3845/2655

CUMMINGS, J., 2018. 12 Possible Causes of a Failing Website [online] [viewed 13 Apr 2021]. Available from: https://www.websitepulse.com/blog/12-possible-causes-of-a-failing-website#:~:text=Whether%20it%20is%20a%20technical

GILLIS, A., 2019. What Is Apache JMeter? - Definition from WhatIs.com [online] [viewed 13 Apr 2021]. Available from: https://searchsoftwarequality.techtarget.com/definition/Apache-JMeter#:~:text=In%20addition%20to%20load%20testing

GITHUB, 2021. About Project Boards - GitHub Docs [online] [viewed 20 Apr 2021]. Available from: https://docs.github.com/en/github/managing-your-work-on-github/about-project-boards

GURU99, 2019. What Is JMeter? Introduction & Uses [online] Available from: https://www.guru99.com/introduction-to-jmeter.html

NIGAM, V., 2019. Processing Time Series Data in Real-Time with InfluxDB and Structured Streaming [online] [viewed 13 Apr 2021]. Available from: https://medium.com/analytics-vidhya/processing-time-series-data-in-real-time-with-influxdb-and-structured-streaming-d1864154cf8b

SARAWAGI, S., 2019. What Is Grafana? Why Use It? Everything You Should Know about It [online] [viewed 13 Apr 2021]. Available from: https://www.8bitmen.com/what-is-grafana-why-use-it-everything-you-should-know-about-it/

VOLMINGER, A., 2020. CI/CD for Java Maven Using GitHub Actions [online] [viewed 20 Apr 2021]. Available from: https://medium.com/@alexander.volminger/ci-cd-for-java-maven-using-github-actions-d009a7cb4b8f

APPENDICES

Figure 1: System Diagram representing all three main components and their containers –

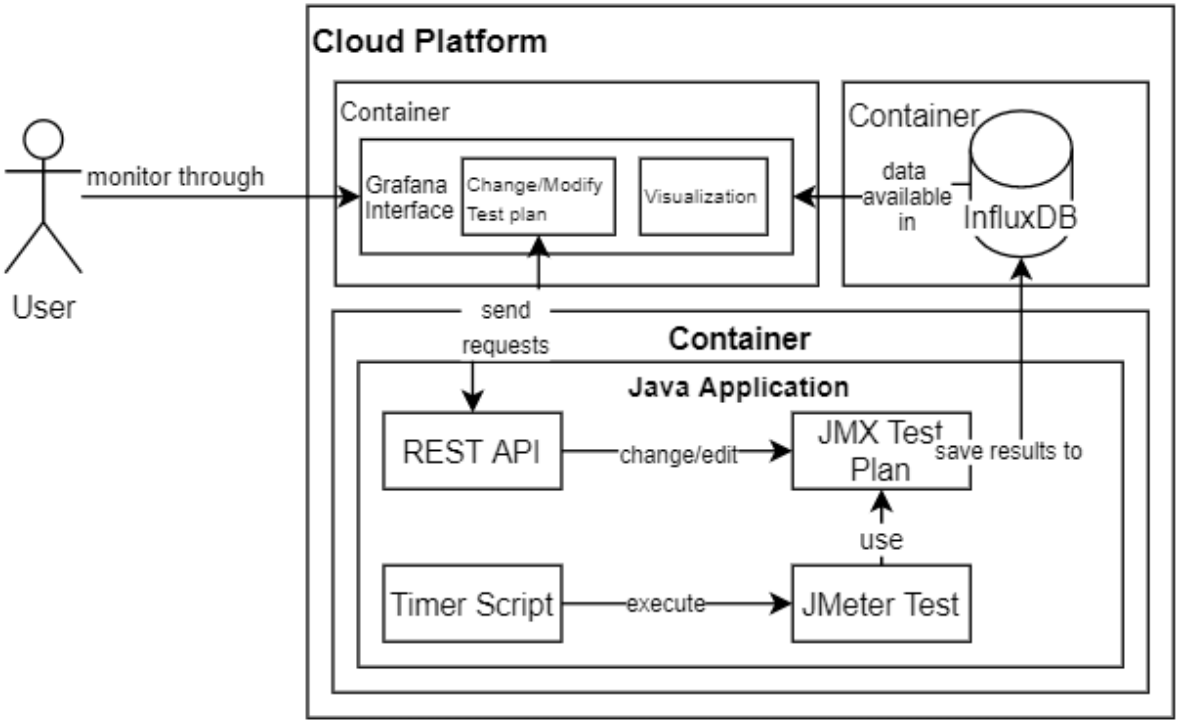

Figure 2: Main features of the Use Case Diagram –

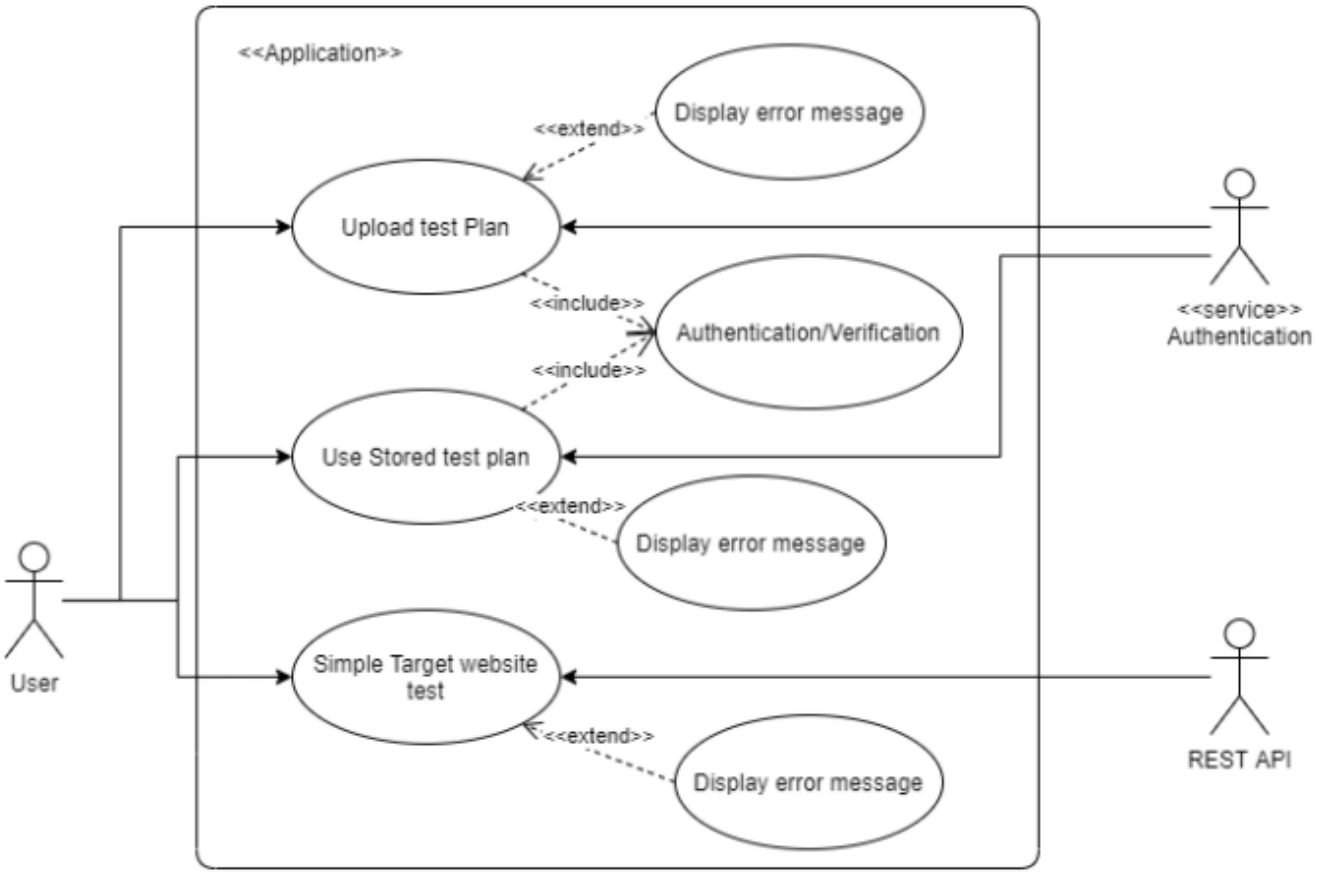

Figure 3: System concept Robustness Diagram -

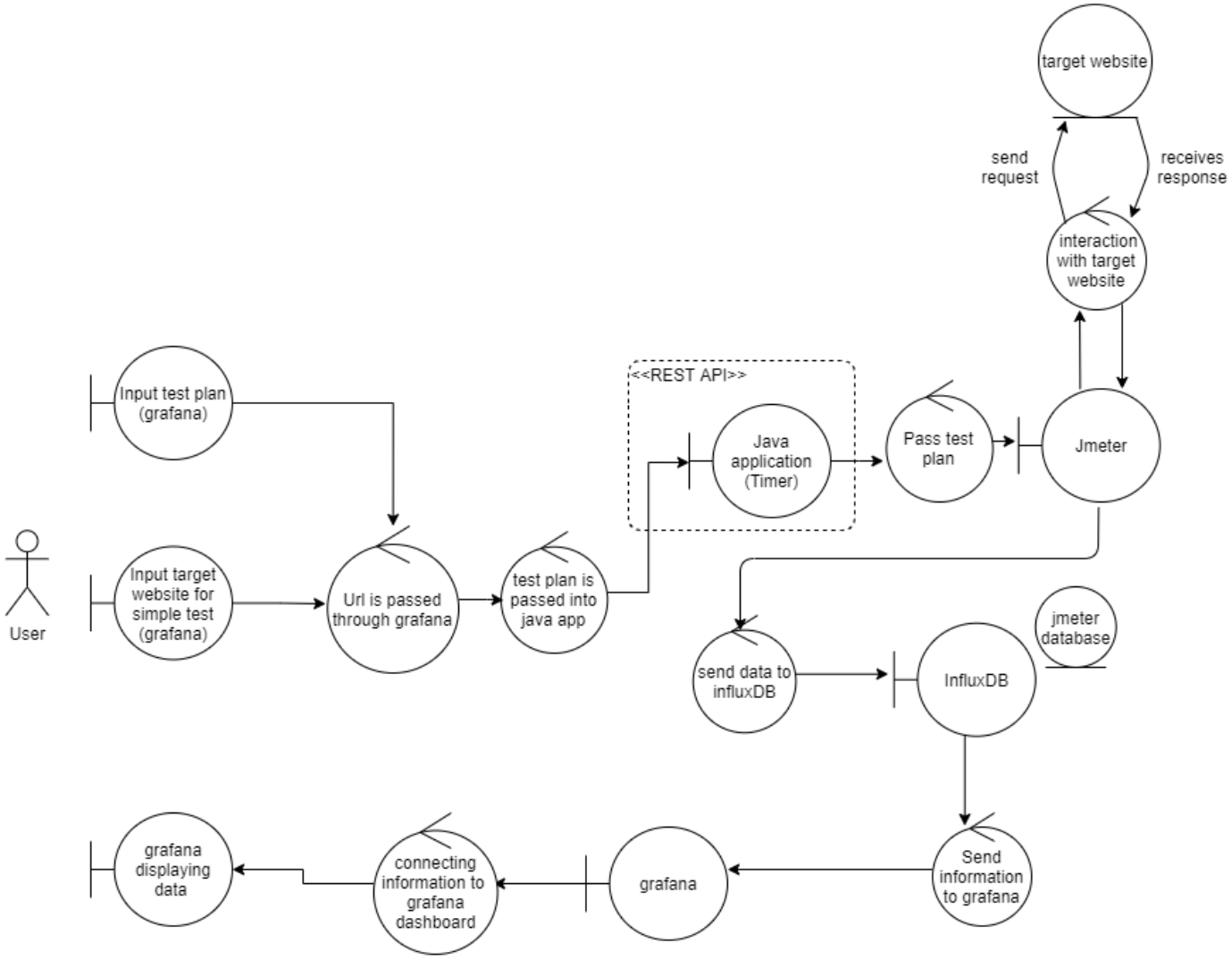

Figure 4: 3 Project Sprints -

Sprint 1) 9/2/21 - 26/2/21	Sprint 2) 1/3/21 - 20/3/21	Sprint 3) 22/3/21 - 23/4/21
Gather requirements	**Proof of Concept**	**Minimal Viable Product**