****

**Department Of Computer Science & Engineering**

**Testing Tools Report**

**Group Member Information**

| **Ovie Rahaman Sheikh** | **011201306** |
| --- | --- |
| **Nuren Akter** | **011201291** |

**Submitted To:**

**Md. Mohaiminul Islam (Lecturer)**

**Date : 23 December 2023**

**Table of Content**

1 Introduction

1.1 Objectives

1.2 Tools Selection

1.3 System Selection

2 Overview of Use Case

3 Test Suite Design and Justification

3.1 Test Suite Design

3.1.1 Environment

3.1.2 Initialization

3.1.3 Test Step

3.1.4 Oracle

3.2 Justification

4 Result (Output)

5 Conclusion

6 Contribution

6.1 Team Members Contributions

6.2 Github Link

1 Introduction

1.1 Objectives

The primary objective of incorporating Selenium WebDriver and Gatling into our testing framework is to enhance the efficiency, reliability, and comprehensiveness of our software testing processes.

**Selenium WebDriver:**

Selenium WebDriver is a popular open-source tool used for automating web browsers. Selenium WebDriver is a web framework that permits us to execute cross-browser tests. Parallel test execution is also supported. Selenium WebDriver supports multiple programming languages, including Java, Python, C#, and JavaScript, making it versatile on different platforms.

**Gatling:**

Gatling is an open-source performance testing tool designed for load and performance testing of web applications. It is often used by developers to simulate a large number of users accessing a web application simultaneously.

1.2 Tools Selection

**Tools 1: Selenium WebDriver**

Implement Selenium WebDriver to automate the functional testing of web applications. Create robust and maintainable automated test scripts for various browsers, ensuring cross-browser compatibility. Verify the accuracy of web application functionality by simulating user interactions and validating expected outcomes. Conduct regression testing to identify and rectify any potential issues resulting from code changes or updates.

**Tools 2: Gatling**

Utilize Gatling for performance testing to assess the scalability, responsiveness, and stability of our web applications under different load conditions. Design and execute performance tests to identify bottlenecks, latency issues, and overall system performance metrics. Generate detailed performance reports and analysis, including response times, throughput, and error rates, to facilitate informed decision-making.

Conduct stress testing to evaluate the system's behavior under extreme conditions, ensuring its reliability during peak usage periods. By integrating Selenium WebDriver for functional testing and Gatling for performance testing, we aim to achieve a comprehensive testing approach that addresses both the functional correctness and the performance aspects of our web applications. This will result in a more robust and reliable software release, enhancing the overall quality and user experience of our applications.

1.3 System Selection

**System : Swag Labs**

Swag Labs is an open-source platform offering a cost-effective solution for testing websites. Its accessibility empowers developers to conduct comprehensive testing, ensuring the reliability and performance of web applications.

We've chosen Swag Labs for Gatling and Selenium WebDriver testing due to its seamless integration of both tools, providing a unified solution for performance and functional testing. Swag Labs enables realistic testing scenarios, scalability with Gatling, and cross-browser compatibility with Selenium WebDriver. Its user-friendly setup, strong community support, and integration with CI/CD practices enhance testing efficiency. This cost-effective platform ensures comprehensive testing without compromising quality. In summary, Swag Labs aligns with our testing goals by offering a holistic, accessible, and budget-friendly solution for ensuring the reliability and performance of our applications.

2 Overview of Use Case

**Tool 1: Selenium WebDriver**

Selenium WebDriver automates interactions with web browsers, allowing testers to simulate user actions such as clicking, typing, and navigating through web pages.

**Cross-Browser Testing:**

It enables testing across different web browsers (e.g., Chrome, Firefox, and Safari), ensuring consistent behavior and compatibility.

**Dynamic Content Testing:**

Selenium can interact with dynamic content on web pages, making it suitable for testing modern web applications that rely on AJAX, JavaScript, and dynamic elements.

**Data-Driven Testing:**

Selenium WebDriver can be used with data-driven frameworks, enabling the execution of the same test with multiple sets of input data.

**UI/Functional Testing:**

Selenium WebDriver is primarily used for testing the user interface and functional aspects of web applications.

**Browser Automation:**

It allows the automation of repetitive tasks in web browsers, such as form submissions, navigation, and data extraction.

**Tool 2: Gatling**

"Gatling" is an open-source load testing framework. Gatling is used to simulate the behavior of users on a web application and measure its performance under different conditions. Here's an overview of use cases for Gatling:

**1. Load Testing:**

* Scenario Simulation: Gatling allows you to define user scenarios that simulate the behavior of real users interacting with your web application. This helps in assessing how your system performs under different loads.
* Concurrent Users: You can simulate a large number of concurrent users to see how your application handles heavy traffic and identify performance bottlenecks.
* Stress Testing: Gatling can be used to apply stress to your system by simulating a higher load than what it is expected to handle. This helps you identify the breaking point of your application.

**2. Performance Testing:**

* Response Time Analysis: Gatling provides detailed reports that include response times for different requests. This helps in identifying slow-performing components and optimizing them.
* Throughput Analysis: You can measure the throughput of your application, which is the number of requests processed per unit of time, to ensure it meets the required performance standards.

**3. Scalability Testing:**

* Horizontal Scaling: Gatling can be used to assess how well your application scales horizontally by adding more servers. This helps in optimizing your infrastructure for increased demand.
* Vertical Scaling: Evaluate the performance improvements achieved by upgrading individual components, such as the CPU or memory of a server.

Overall, Gatling is a versatile tool that helps teams ensure the reliability, scalability, and performance of their web applications. It is widely used in the software development lifecycle to catch performance issues early and deliver high-quality applications to end-users.

3 Test Suite Design And Justification

3.1 Test Suite Design:

3.1.1 Environment:

**Tool 1: Selenium WebDriver**

1. Setup Environment

Install Java JDK.

Install the Eclipse IDE.

2. Configure Selenium Dependencies

Create a new Java project in Eclipse.

Download Selenium WebDriver JAR files.

Add Selenium JAR files to your project.

3. Create a Java Class

In Eclipse, create a new Java class.

Write a simple class with a `main` method.

4. Write a Simple WebDriver Test

Use the Selenium WebDriver API to perform actions (e.g., open a website, interact with elements).

Save your Java class.

5. Run the Test

Right-click on your Java class in Eclipse.

Select `Run As > Java Application`.

**Tool 2: Gatling**

To run Gatling tests on our PC:

1. Install Gatling:

Download the Gatling bundle from the official website.

Extract the downloaded archive to your preferred location.

2. Navigate to the Gatling Directory:

Open a terminal or command prompt.

Change to the Gatling directory using the `cd` command.

3. Write a Gatling Test:

Create or place your Gatling test script in the `user-files/simulations` directory.

4. Run the Gatling Test:

Execute the Gatling script using the following command:

./bin/gatling.sh -nuren.package.GatlingTesting

5. Select Simulation:

Gatling will list available simulations; choose the number corresponding to your simulation.

6. Run Configuration:

Confirm the run configuration or customize it as needed.

7. View Results:

Open the report in a web browser to analyze the test results.

3.1.2 Initialization

**Tool 1: Selenium Webdriver**

**Test Case Flow Chart:**

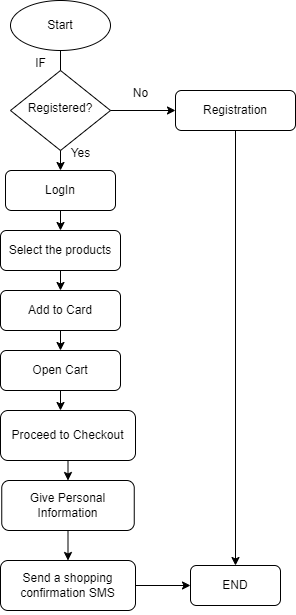
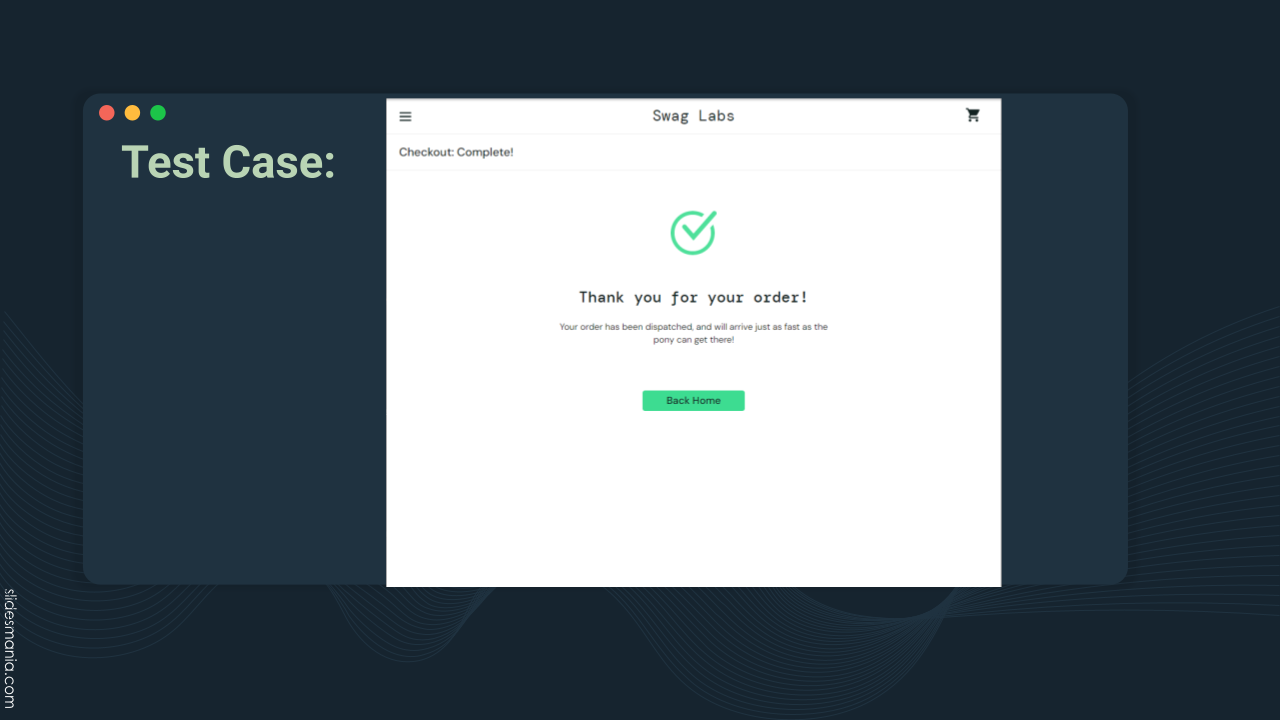
****

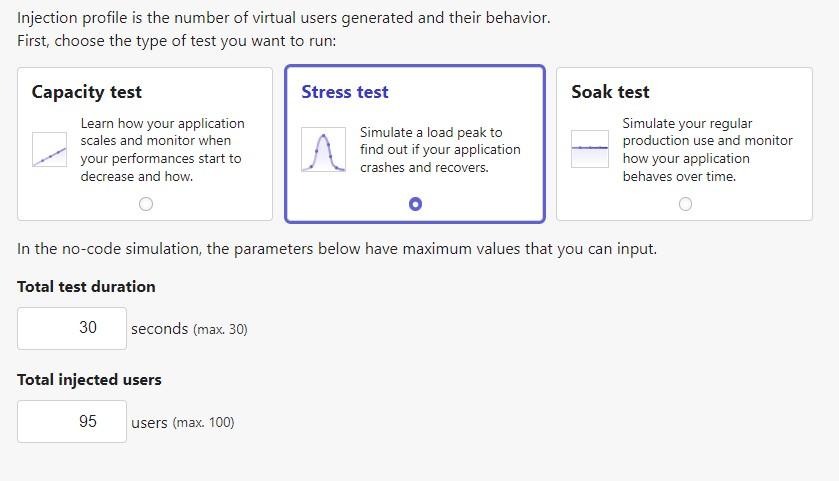
Fig 1.1: Flow Chart of Selenium WebDriver Testing

**Test case initialization :** Our primary objective is to seamlessly navigate to the designated page without encountering any interruptions and successfully validate the confirmation message "Thank you for your order!" displayed. This crucial step in our testing process ensures that end-to-end functionality is robust, providing users with a smooth and reliable experience. By meticulously verifying the completion of the order and the corresponding acknowledgment, we affirm the integrity of the entire transaction process, contributing to the overall quality and user satisfaction of our application. This meticulous testing approach aligns with our commitment to delivering a flawless and dependable user journey.



If the result matches the actual text,which is “Thank you for your order!” then we can understand that our shopping is complete and the test is done.

**Tool 2: Gatling**

**Test case initialization :**

**Test Case Flow Chart:**

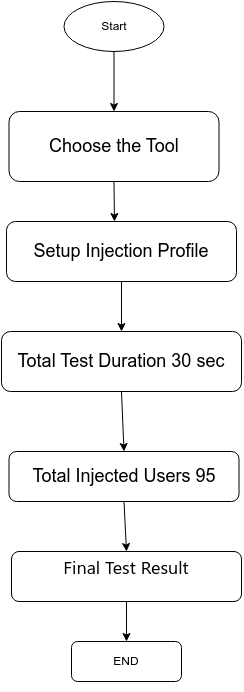


Fig 1.1: Flow Chart of Gatling Testing

3.1.3 Test Step

**Tool 1: Selenium WebDriver**

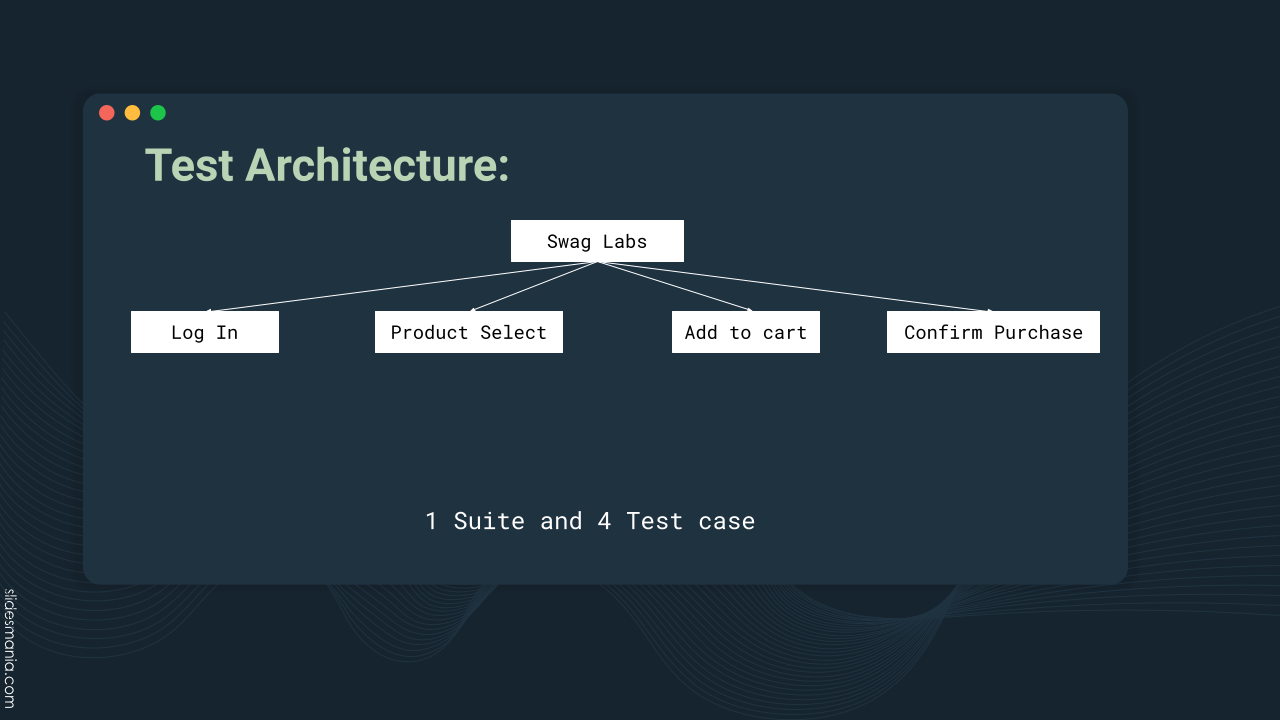
Step 1: Login

Step 2: Click on a Product

Step 3: Add to Card

Step 4: Confirm Purchase

**Test Architecture:**

****

**Tool 2: Gatlings**

Step 1: Setup Injection Profile

Step 2: Total Test Duration 30 sec

Step 3: Total Injected Users 95

Step 4: Final Test Result

3.2 Justification:

**Tool 1: Selenium WebDriver**

Objective: Simulate real user interactions and verify transactional flows.

Reasons: Complex user interactions, transaction integrity validation, comprehensive UI testing, realistic test scenarios.

**Tool 2: Gatling**

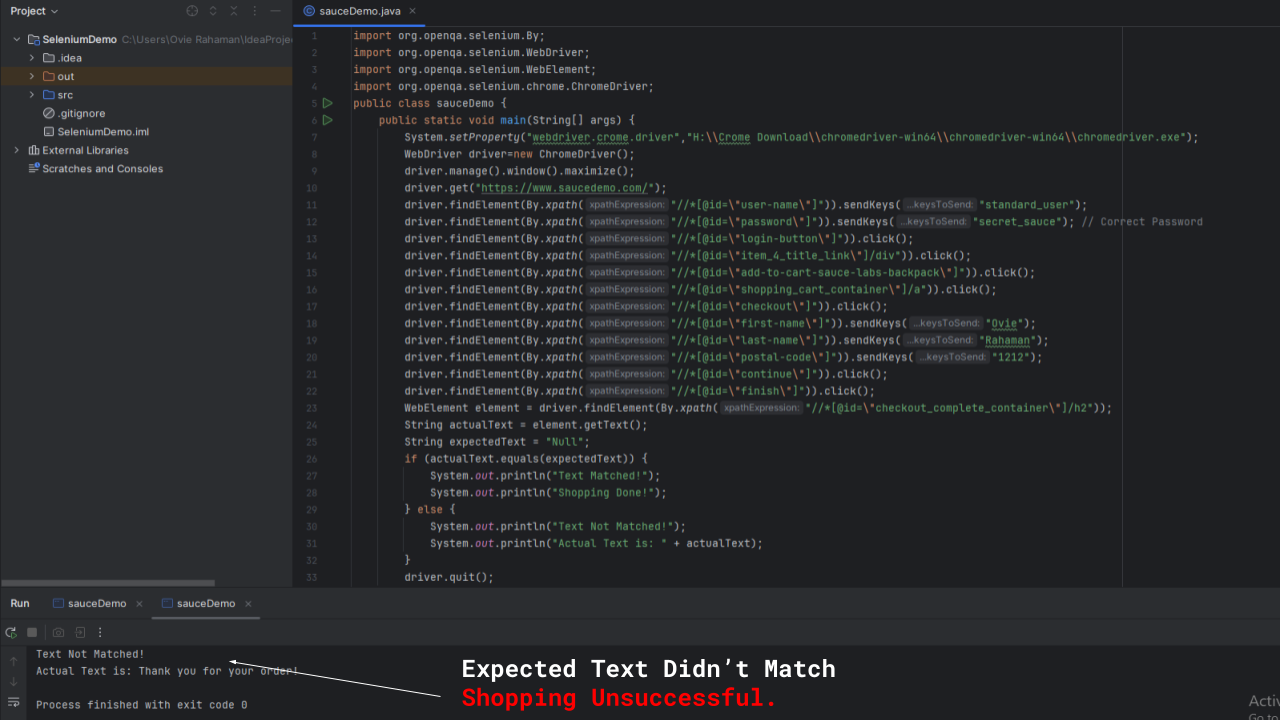
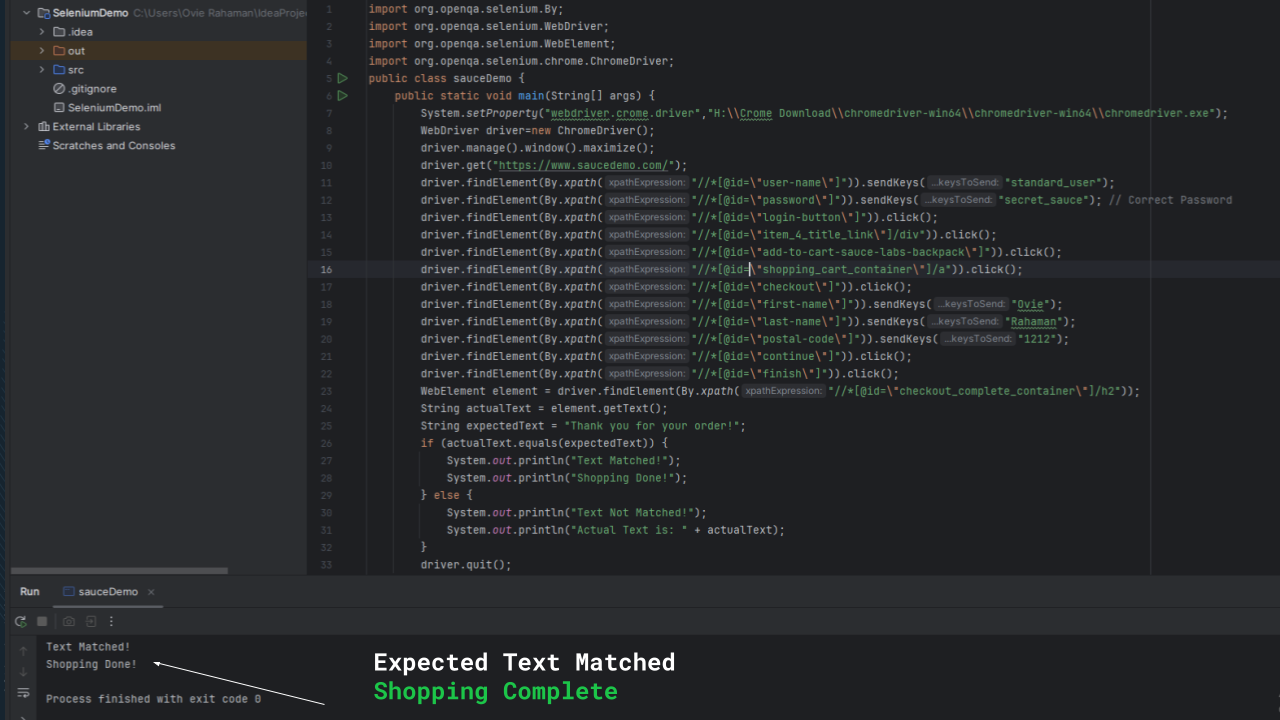
Objective: Evaluate stress testing and identify bottlenecks.

Reasons: Real-world scenario emulation, proactive issue identification, optimizing resource allocation, scalability assessment.

4 Result

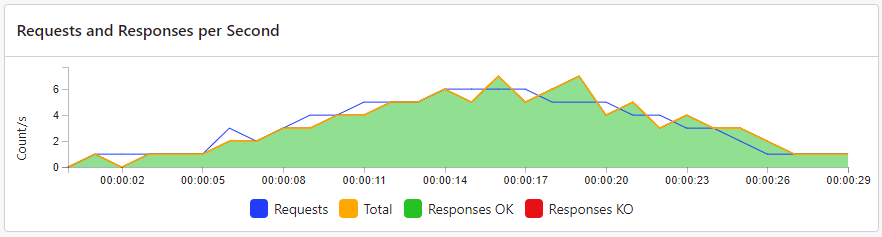
**Tool 1: Selenium WebDriver**

If expected Text matched, then shopping will be complete, and the testing will be complete.

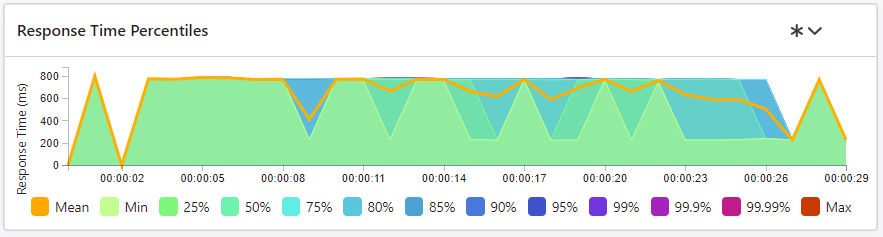


**Tool 2: Gatling**

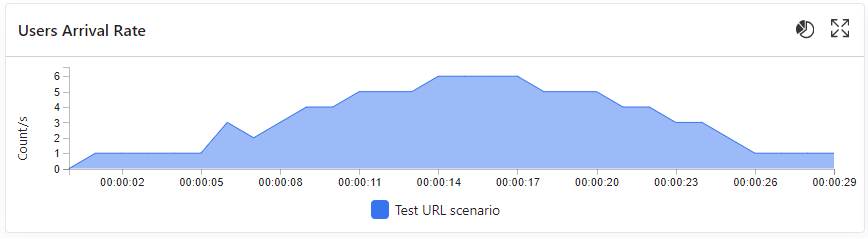
Requests and Responses per Second:



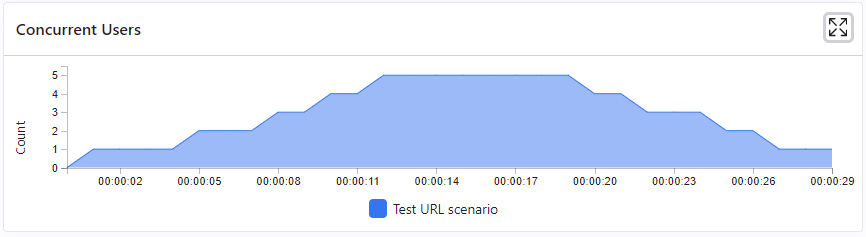
Response Time Percentiles:



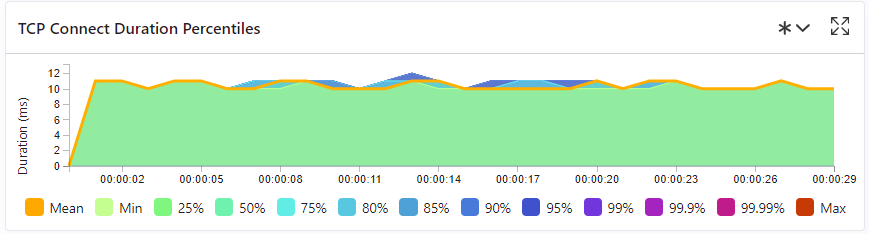
User Arrival Rate:



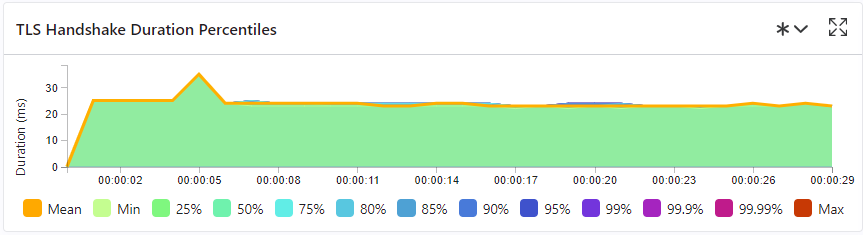
Concurrent Users:



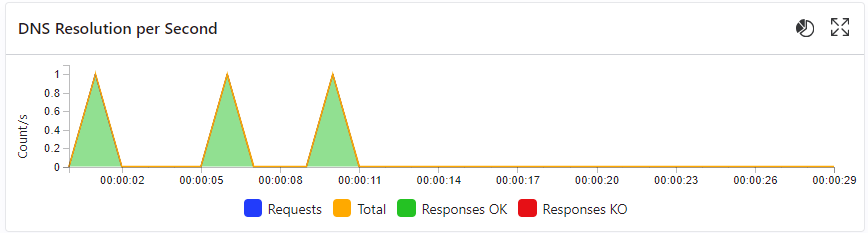
TCP Connect Duration Percentiles:



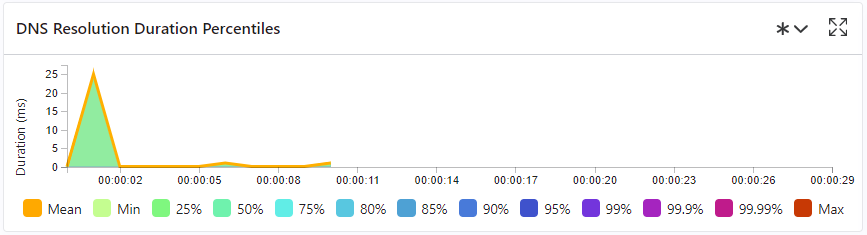
TLS Handshake Duration Percentiles:



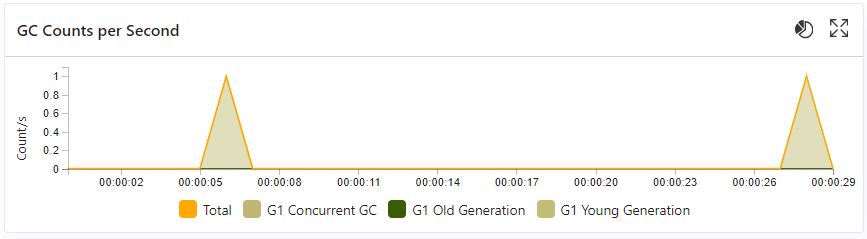
DNS Resolution per Second:



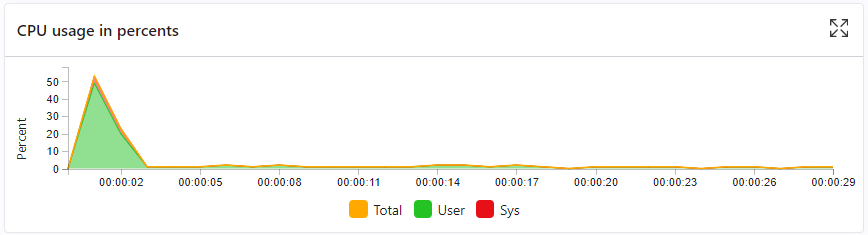
DNS Resolution Duration Percentiles:



GC Counts per Second:



CPU usage in percents:



5 Conclusion

In summary, the combination of Gatling for performance testing and Selenium WebDriver for functional testing provides a powerful and comprehensive approach to software quality assurance. This integration facilitates early issue detection, reduces manual testing efforts, and supports continuous integration and delivery practices. Together, these tools contribute to the development of high-quality, performant software with efficient testing processes.

6 Contribution

6.1 Team Members Contributions

| **Name** | **Tools** |
| --- | --- |
| Ovie Rahaman Sheikh | Selenium WebDriver |
| Nuren Akter | Gatling |

6.2 Github Link

Selenium Testing Github Repository: https://github.com/ovierahaman/SeleniumTesting.git

Gatling Testing Repository: https://github.com/nurenakter/GatlingTesting.git