A.5 Usability Evaluation Task Definition

This section contains the definition of the usability assessment task and the questionnaire.

Introduction:

Please note, that you are free to cancel your participation at any time without any disadvantages. Your data will be handled completely anonymous.

You are about to participate in a usability study of the MiSArch Experiment Tool. This is a tool designed to help engineers create, configure, and execute load tests in combination with chaos engineering for microservice-based systems.

In this test, your role is to act as a tester or performance engineer responsible for assessing the resilience of the distributed application MiSArch under stress and failure conditions. The goal is to make it easier to simulate real-world production issues by combining load testing (e.g., high traffic scenarios) with controlled chaos events (e.g., service crashes, network delays, or CPU saturation).

You will be asked to complete a set of predefined tasks using the MiSArch Experiment Tool. These tasks will involve:

- Designing an experiment setup involving both load and chaos components
- Executing the experiment scenario
- Observing and interpreting the outcome

We are not testing your performance, but rather evaluating how understandable, usable, and effective the software is for someone in your role. We are especially interested in:

- How intuitive the interface and workflows feel
- Whether you can confidently perform tasks without confusion
- If the tool helps you reason about system behavior under failure conditions

After completing the tasks, you will be asked to fill out a short questionnaire about your experience. This will include both quick ratings and open-ended questions.

Please feel free to think aloud as you work through the tasks, and share any thoughts, frustrations, or surprises you encounter along the way. Your feedback is extremely valuable in helping us evaluate the functionality of the **MiSArch Experiment Tool.**

Let's begin!

MiSArch Background

Before starting with the actual experiment let's give you some information on MiSArch. MiSArch is short for MicroService Architecture and represents a reference architecture in form of a component-based microservice application in the e-commerce domain. It is designed to be used in research and teaching for cloud-native applications. MiSArch consists of 16 microservices using different technology stacks, such as Kotlin with Spring Boot, Typescript with NestJS or Vue.js, and Rust with axum. The different microservices are shipped as containers and deployed in a Kubernetes cluster.

The different components are e.g. a gateway, a catalog, a payment, and an order service.

Each of the 16 components is equipped with sidecar containers, one for specific failure configuration and a dapr sidecar for handling network traffic. The dapr sidecars communicate using an event bus. The configuration sidecar, in combination with the configuration service, centrally enables researchers to configure specific environment variables and failures for each component (this is called MiSArch Experiment Config).

MiSArch components can be monitored using application performance monitoring with Open-Telemetry.

The MiSArch Experiment Tool is an additional layer consisting of four components that interact with the MiSArch application. You will evaluate the MiSArch Experiment Tool in this task.

The tool is designed to create and execute complex tests consisting of load tests and chaos engineering concepts. It consists of a frontend (which you will use in this evaluation), and components for configuring and executing load tests with Gatling and Chaos Experiments with ChaosToolkit as well as the MiSArch Experiment Config. Finally, a self-hosted Grafana instance allows you to visualize the experiment results in the end.

Experiment Definition:

- Open Grafana for once: https://misarch-experiment.gropius.dev/
- Then open the Experiment Tool under: https://misarch-experiment.gropius.dev/frontend/
  - Please use it in full screen, otherwise it might not work 100% correct.

Start: Get familiar with the tool. Open the existing experiment, look around and get an understanding of the functionality. Especially, read the different help sections to get a grasp of the different features.

Task: Create and execute a new Experiment:

- Create a realistic load test with at max 9 arriving users per second and a total duration of 3 minutes.
- Configure the following parameters of the experiment:
  - Add a warm-up phase of 1 minute with 10 active users.
  - Ensure the gateway's pod is deleted 5 seconds after the experiment starts.
  - Ensure 20s after the experiment starts 50% of the HTTP requests towards the catalog have a delay of 2s and 100% of the HTTP requests result in a HTTP 500 every time.
  - Ensure that after the failure for catalog was running for 60s. catalog is not having a failure anymore (i.e., after 80s of the total experiment). In the same failure set configure that 50% of the HTTP messages towards payment should have a delay of 2s and 100% do result in a 404 every time. This config should last for another 60s (i.e. until 140s of the experiment).
  - Configure the following goals for the requests:
    - Maximum response time for all requests of 1500ms (choose red color).
    - Mean response time for all requests of 750ms (choose red color).
    - Maximum 20% failed requests (choose red color).
  - Ensure that in the abortedBuyProcessScenario the pause between fetching the specific address and the shopping cart entry is between 1000ms and 3000ms.
  - Configure the load of buyProcessScenario from second 120 to 130 to be 5 users arriving per second.
- Execute the experiment and wait for it to finish (grab a coffee; this can take 4-6 min).
- After successful execution open the results dashboard.
- Try to understand the dashboard and see the load, the failure points that have been introduced, and if the goals are violated.

Finally: Rate your experience and the functionality in the questionnaire.

THANK YOU for your participation!

Section A: Usability & Task Completion (Likert-Scale)

(1 = Strongly Disagree, 5 = Strongly Agree)

1. I found the process of creating an experiment intuitive.
2. The software made it easy to combine load and chaos testing.
3. Executing the experiment was straightforward.
4. The software provided sufficient feedback and guidance during the task.
5. I understood the relationship between load testing and chaos injection.
6. I was able to configure the experiment the way the task requested.
7. I think the tool combines all necessary features for creating load and chaos experiments.
8. I would feel confident using this software in a real project.

Section B: System Usability Scale (SUS)

(1 = Strongly Disagree, 5 = Strongly Agree)

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

Section C: Open-Ended Questions

1. What aspects of the software did you find most useful or intuitive?
2. Were there any features or workflows that you found confusing or frustrating?
3. Were there any features or workflows that you were missing?
4. Did the software support your understanding of chaos engineering concepts?
5. Did the software support your understanding of load tests?
6. If you could improve one thing about the experience, what would it be?
7. Do you have suggestions for features or improvements?