Beet Seed — відпрацюй навички на базовому рівні.

1. Створи набір з GET, POST, PUT, PATCH, DELETE запитів до [JSONPlaceholder](https://jsonplaceholder.typicode.com/), які надсилаються протягом 10 секунд у 3 ітерації.

2. До кожного з запитів застосуй 3 різних assert’и.

3. Результати виконання тестів мають бути отримані за такими Listener’ами:

* View Results in Tree
* Summary Report.

4. Опиши висновки щодо результатів тестування (базуючись на репортерах) в окремому файлі.

5. Створений тест-план та документ з описом результатів виконання репортерів додай в свій репозиторій в GitHub.

# **Performance Test Plan**

### **1. Introduction**

The purpose of this performance test plan is to evaluate the performance of the application's API endpoints. The endpoints under test include GET, DELETE, POST, PUT, and PATCH requests. The primary objectives are measuring response assertion, size, and response assertion.

### **2. Objectives**

* Measure Response Time: Determine the time taken for each type of request (GET, DELETE, POST, PUT, PATCH) to receive a response.
* Response Size: Evaluate the size of the responses returned by each endpoint.
* Response Assertions: Verify that the responses meet the expected criteria (e.g., correct status codes, specific content in the body).

### **3. Scope**

* Endpoints: The test will cover the following HTTP methods: GET, DELETE, POST, PUT, PATCH.
* Metrics: Response time, response size, and response assertions.
* Environment: The tests will be conducted in a staging environment that closely mirrors the production setup.

### **6. Test Scenarios**

* Baseline Test: Measure the performance under normal load conditions.
* Load Test: Gradually increase the load to assess performance under stress.
* Stress Test: Push the system to its limits to find the breaking point.

**6.1 Baselining**

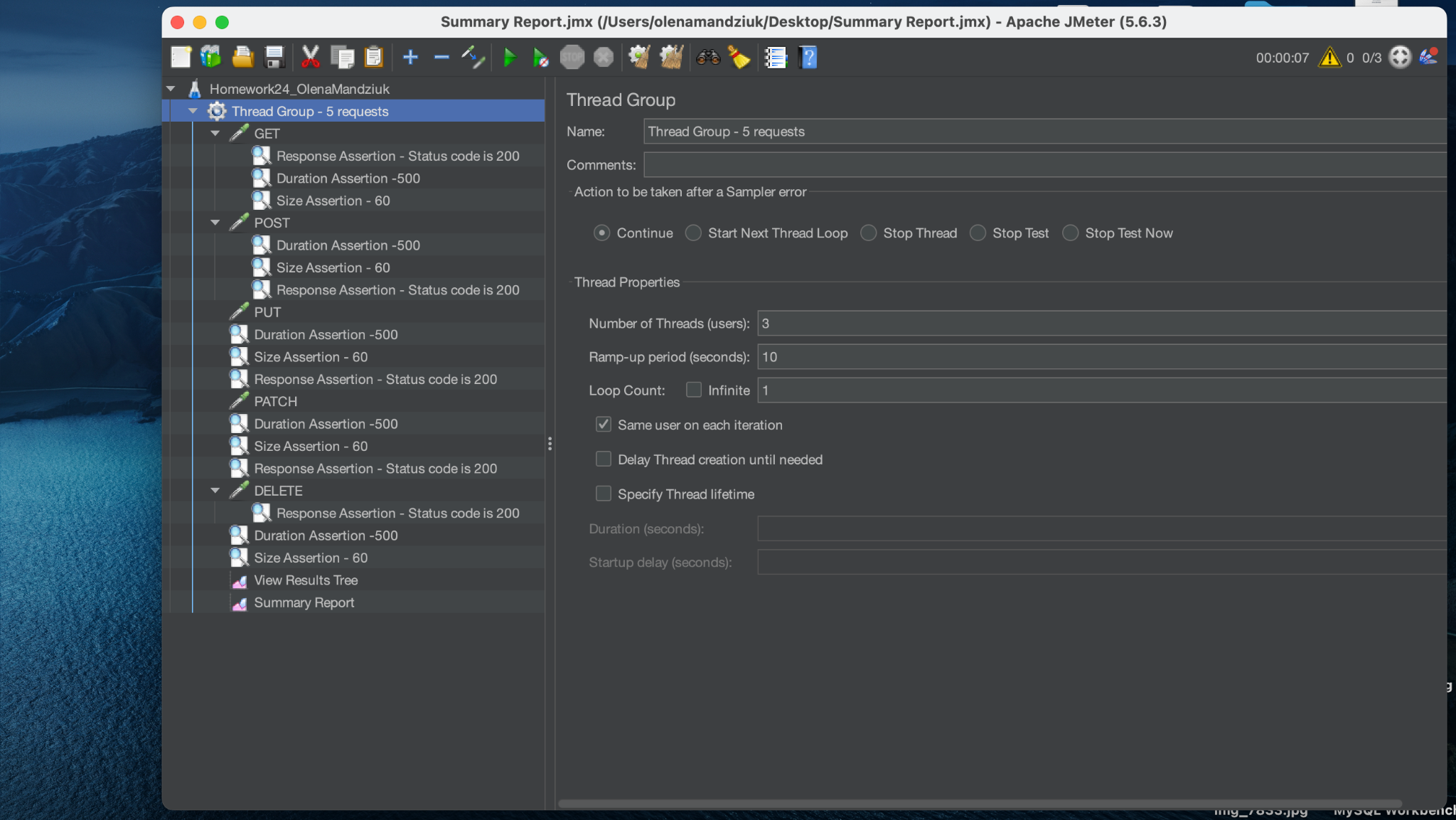
The initial step is to establish a baseline. A test was conducted to verify that the response assertion, size, and response assertion were passed with one user within 10 seconds.

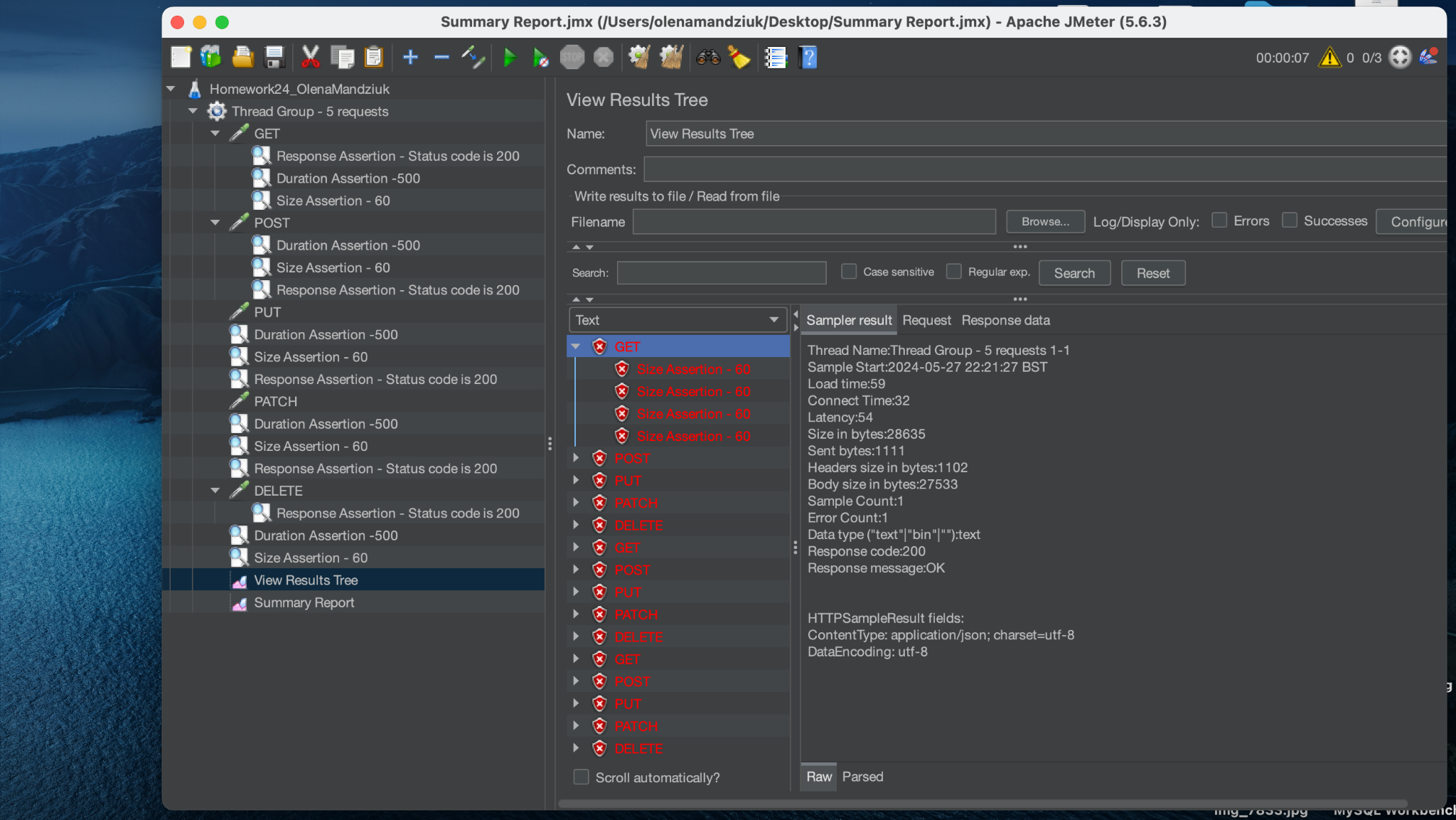
**6.2 Load Testing**

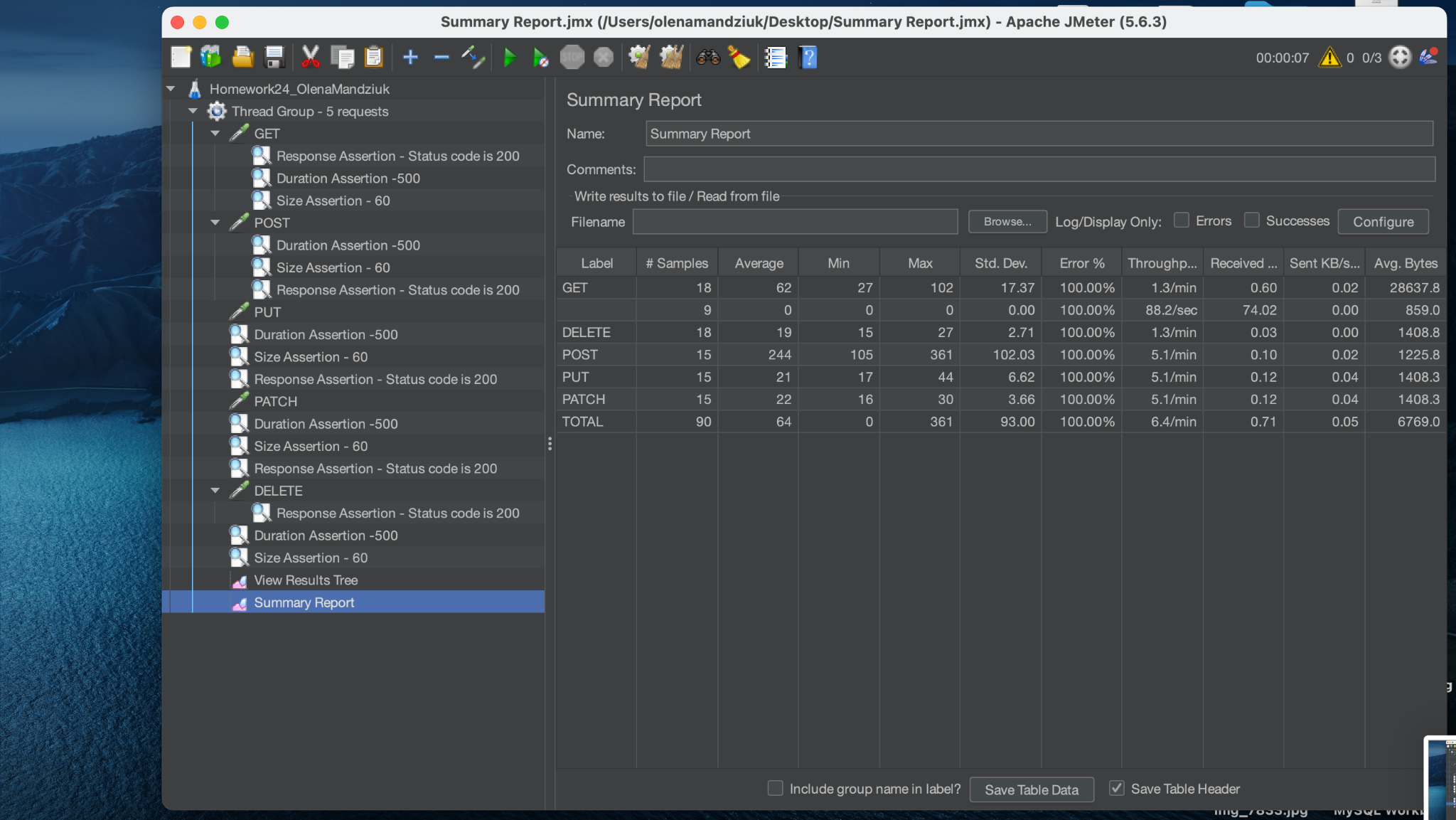
After the baseline metrics are gathered, then the same simulation, which simulates a load profile, is run with an increased number of users to test against the target volumes. 3 users for 10 seconds.

**6.3 Stress Testing**

The aim of stress testing is to find the breaking point of the system, i.e. at what point does the system becomes unresponsive. If auto-scaling is in place, the stress test will also be a good indicator at which point the system scales and new resources are added. For stress testing, the same simulation used for load testing is used but with a higher than expected load.







2. Створи новий тест-план на основі тест-плану з попереднього рівня. В ньому:

* для кожного із запитів зроби стрес-тест використаного API;
* потрібно виявити такі мінімальні комбінації параметрів, за яких вебсервіс перестає витримувати навантаження.

