ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Homework 6

Course: Testing

Student:

Nguyễn Phúc An - 22127002

Lecturer

Dr. Lâm Quang Vũ

Dr. Hồ Tuấn Thanh

Dr Trương Phước Lộc

Monday, 11th August, 2025

ID	Name	Scenario
22127002	**Nguyễn Phúc An**	**- Login and search and view product**
22127211	**Phạm Đình Khôi**	**- Checkout**
22127164	**Trần Nhật Huy**	**- Login as admin and add new product**
22127406	**Nguyễn Quốc Thuần**	**- Contact**

1. Tool Use

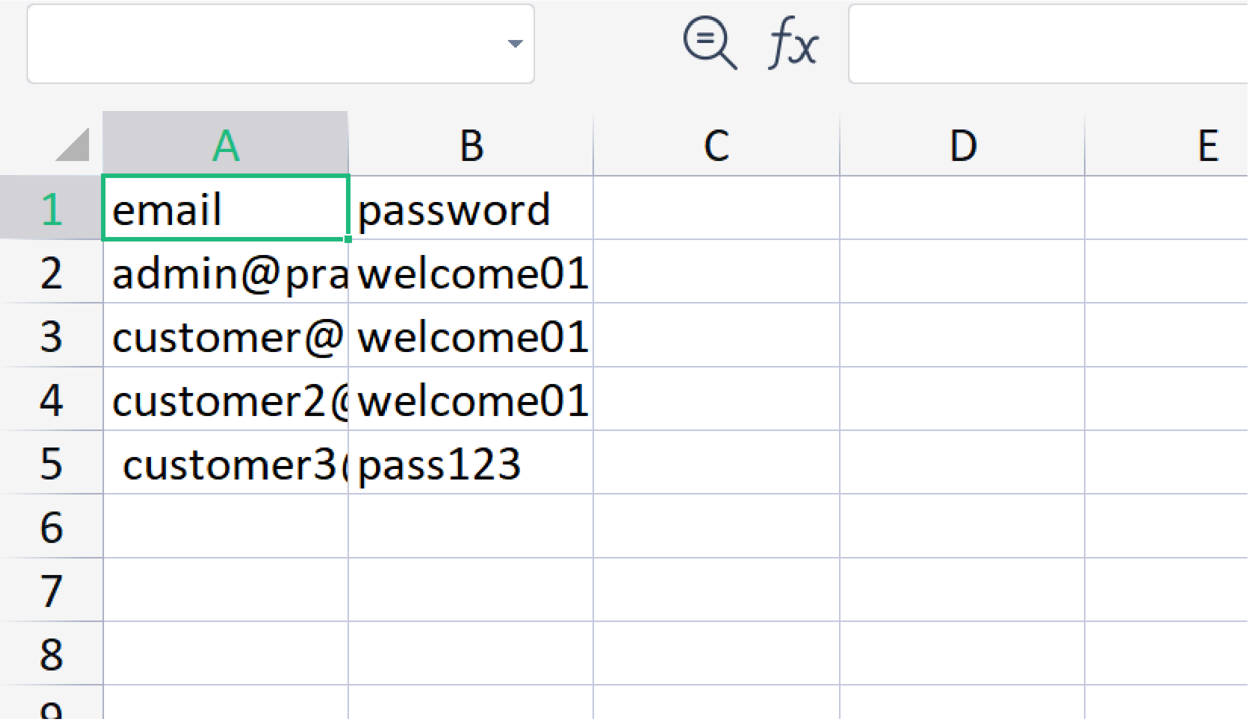
Component	Parameter
OS	Window 11 Pro 64-bit (10.0, Build 26100)
CPU	Intel(R) Core(TM) i7-8650U CPU @ 1.90GHz (8 CPUs), ~2.1GHz
RAM	16GB
Brand	Asus Zenbook
Software	Clone it from Github repo: https://github.com/testsmith-io/practice-software-testing.git Then use the migrate command to seed the database.

- The primary tool used in this homework is **Grafana k6**, a modern, developer-friendly open-source load testing tool designed to measure the performance and reliability of APIs, web applications, and microservices. k6 allows us to define performance test scripts in plain JavaScript, making them easy to write, maintain, and integrate into development workflows.
- In this homework, K6 will be demonstrating for evaluating the performance of the Toolshop application in a specific scenario: Login and Search the product.
- We will use **Docker** to run the application locally via command line: **docker-compose up -d** or GUI desktop.
- On the other hand, **data-driven techniques** will be applied on four different types of performance testing: **stress test, spike test, load test and volume test.**

2. Step-by-step instructions

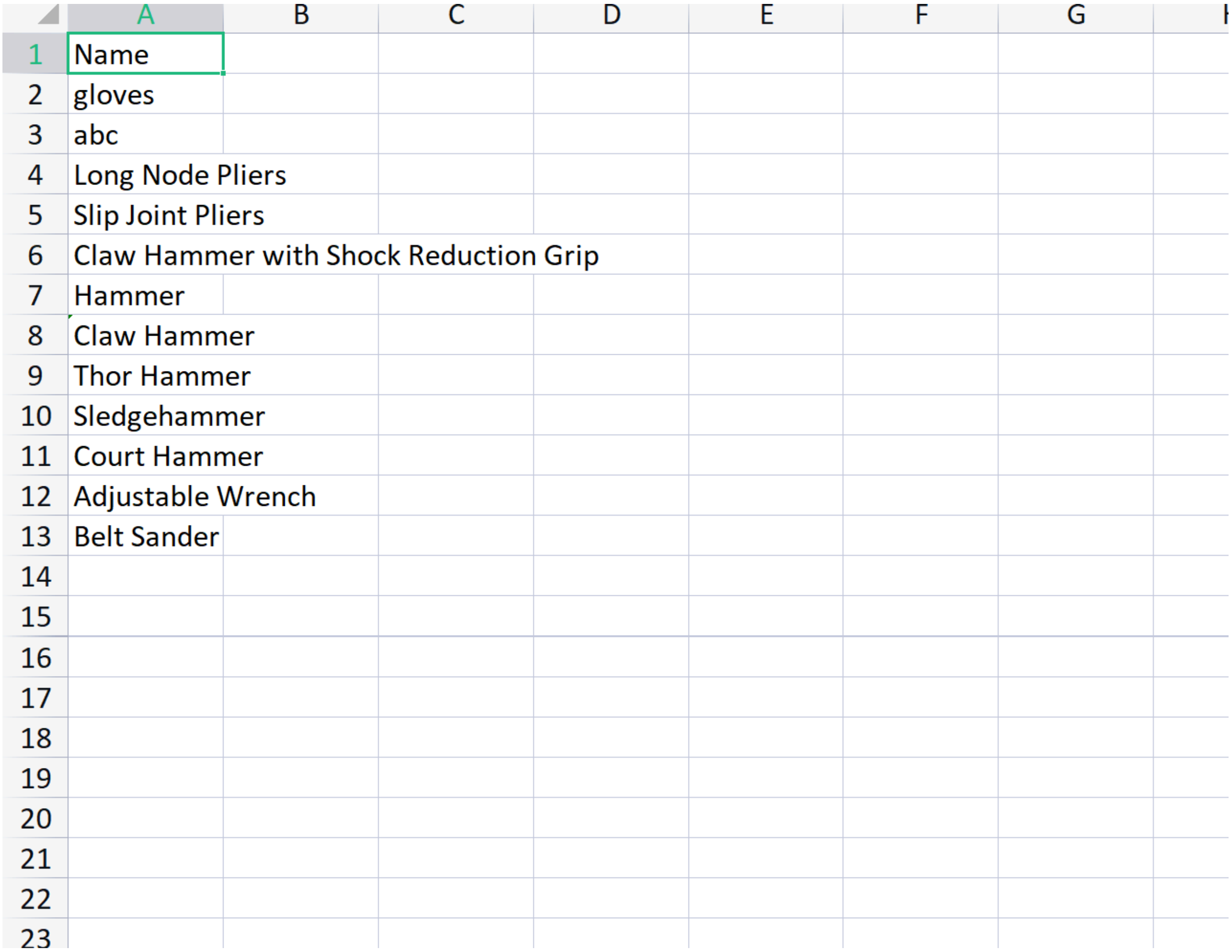
2.1 CSV datasets for data-driven testing

- CSV datasets will be provided for 2 stages: Login and searching. Here are the features of each dataset:
    - Login.csv: email, password
    - Products.csv: Product name



	A	B	C	D	E
1	email	password			
2	admin@pra	welcome01			
3	customer@	welcome01			
4	customer2(	welcome01			
5	customer3(	pass123			
6					
7					
8					

Login dataset



	A	B	C	D	E	F	G
1	Name						
2	gloves						
3	abc						
4	Long Node Pliers						
5	Slip Joint Pliers						
6	Claw Hammer with Shock Reduction Grip						
7	Hammer						
8	Claw Hammer						
9	Thor Hammer						
10	Sledgehammer						
11	Court Hammer						
12	Adjustable Wrench						
13	Belt Sander						
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							

Product dataset

2.2 Write a script file

- The test flow includes 3 processes:

1. **Login** (testing via fetching API /users/login) – cache JWT token per Virtual user (VU)
2. **Search** (testing via fetching API /products/search)
3. **Detail** (tetsing via fetching API /products/{id})

Use handleSummary() to output summary.html (tables) and report.html (charts).

2.3 Threshold expectation

Type	**Error rate**	**Global response time**	**Login p(95)**	**Search p(95)**	**Detail p(95)**	**Check pass rate**
Stress test	5%	1300ms	1200ms	1500ms	1200ms	95%
Spike test	10%	1800ms	1500ms	2000ms	1500ms	90%
Volume test	5%	1500ms	1500ms	1800ms	1500ms	95% Iteration duration p(95): 3500ms
Load test	3%	1000ms	900ms	1200ms	900ms	97%

2.4 Configure Test Modes

Use `MODE` environment variable to switch between:

- **Load** → Baseline moderate load
- **Stress** → Gradually increase to find break point
- **Spike** → Sudden traffic surge and drop
- **Volume** → Large total requests over time
  Each mode has its own **stages** (VU pattern) and **thresholds** (SLA targets for p95 latency, error rates, and checks).

For more definitions about above performance test types, please refer to this link: https://grafana.com/load-testing/types-of-load-testing/

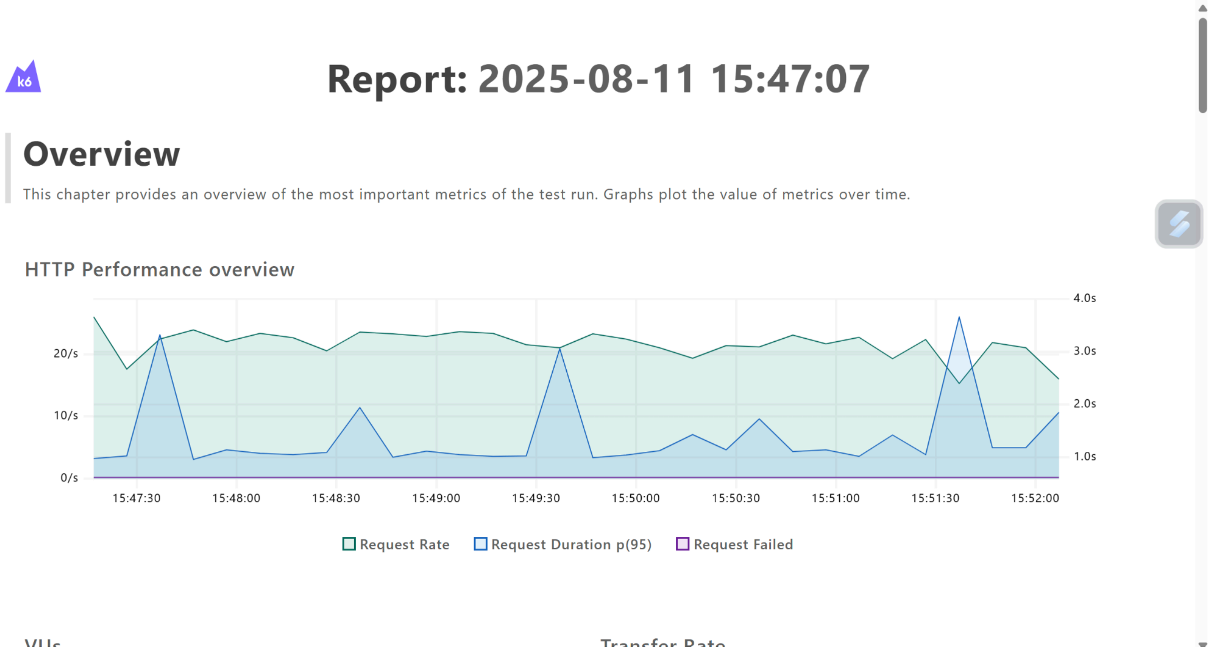
2.5 Run & Export reports

```
# MODE can be “stress”, “load”, “spike” and “volume”
$env:MODE = "stress"
$env:K6_WEB_DASHBOARD = "true"
$env:K6_WEB_DASHBOARD_EXPORT = "report.html"
k6 run .\tests\product.js
```

Once the execution completes, 2 HTML files will be automatically generated, including:

- **report.html** → time-series charts (dashboard export)
- **summary.html** → table stats from htmlReport

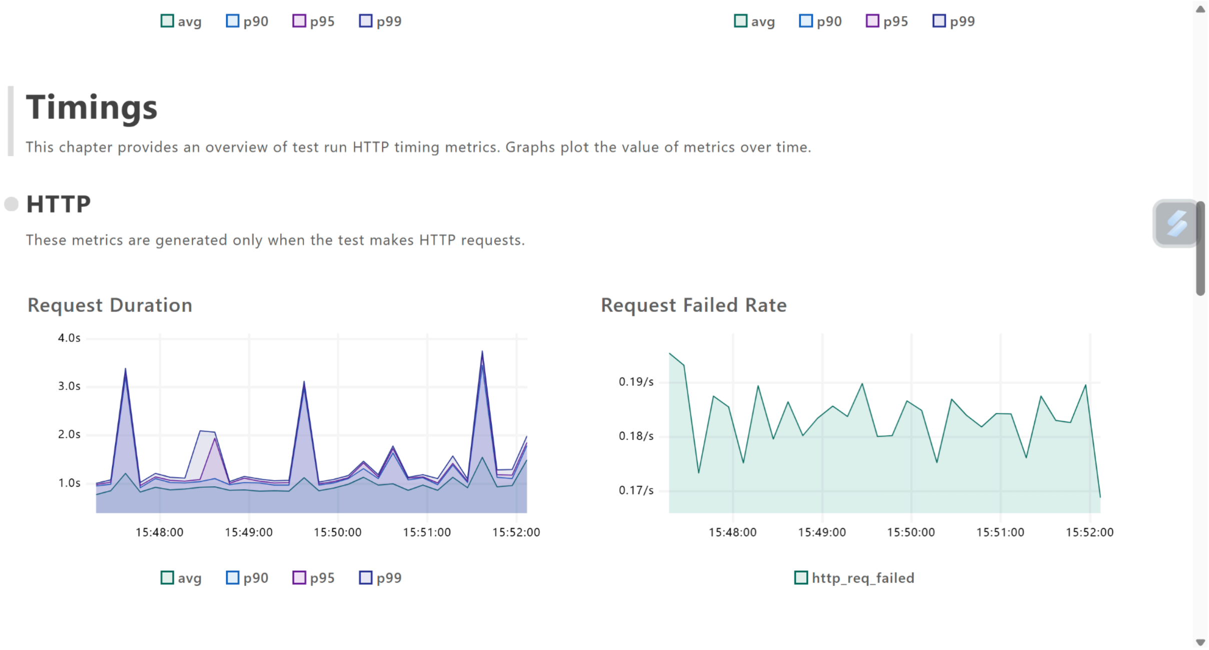
Here are the 2 reports:


Report: 2025-08-11 15:47:07

Overview

This chapter provides an overview of the most important metrics of the test run. Graphs plot the value of metrics over time.

HTTP Performance overview



Timings

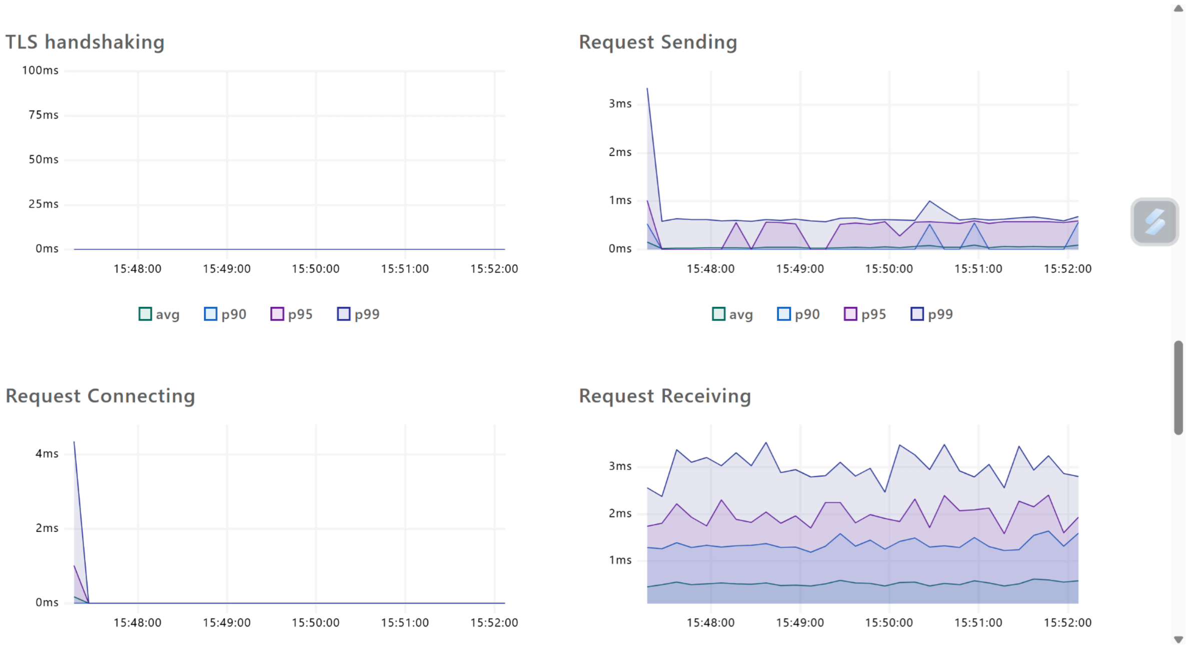
This chapter provides an overview of test run HTTP timing metrics. Graphs plot the value of metrics over time.

HTTP

These metrics are generated only when the test makes HTTP requests.

Request Duration

Request Failed Rate

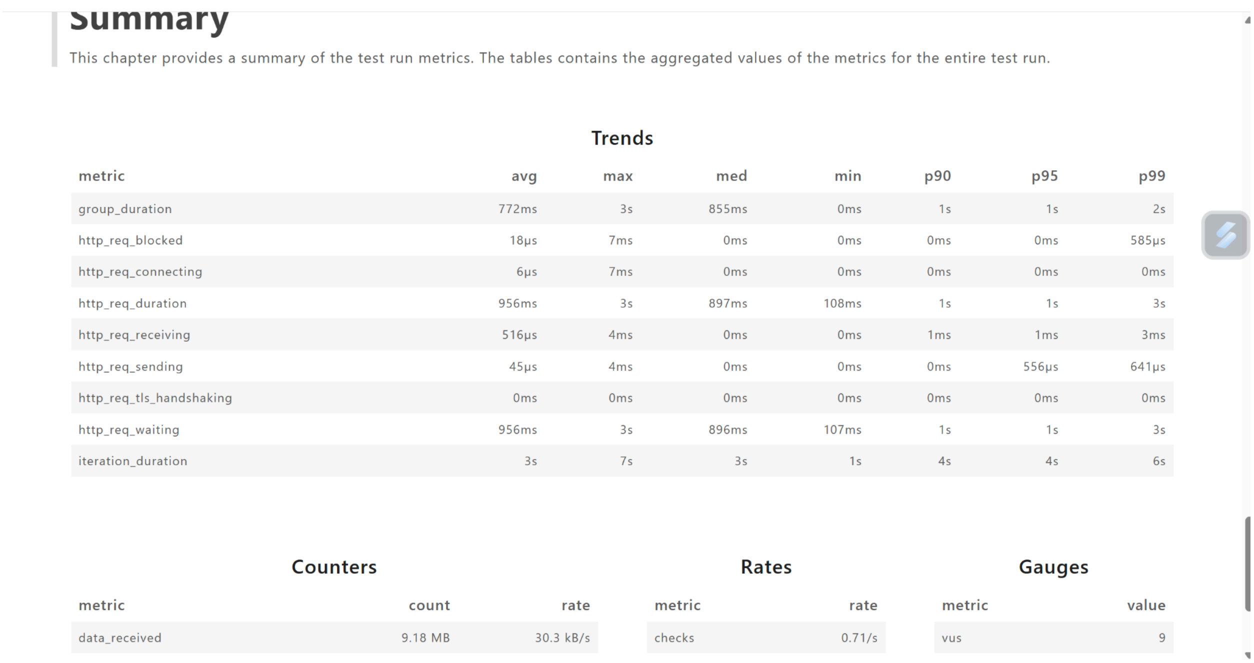


TLS handshaking

Request Sending

Request Connecting

Request Receiving



Summary

This chapter provides a summary of the test run metrics. The tables contains the aggregated values of the metrics for the entire test run.

Trends

metric	avg	max	med	min	p90	p95	p99
group_duration	772ms	3s	855ms	0ms	1s	1s	2s
http_req_blocked	18µs	7ms	0ms	0ms	0ms	0ms	585µs
http_req_connecting	6µs	7ms	0ms	0ms	0ms	0ms	0ms
http_req_duration	956ms	3s	897ms	108ms	1s	1s	3s
http_req_receiving	516µs	4ms	0ms	0ms	1ms	1ms	3ms
http_req_sending	45µs	4ms	0ms	0ms	0ms	556µs	641µs
http_req_tls_handshaking	0ms	0ms	0ms	0ms	0ms	0ms	0ms
http_req_waiting	956ms	3s	896ms	107ms	1s	1s	3s
iteration_duration	3s	7s	3s	1s	4s	4s	6s

Counters

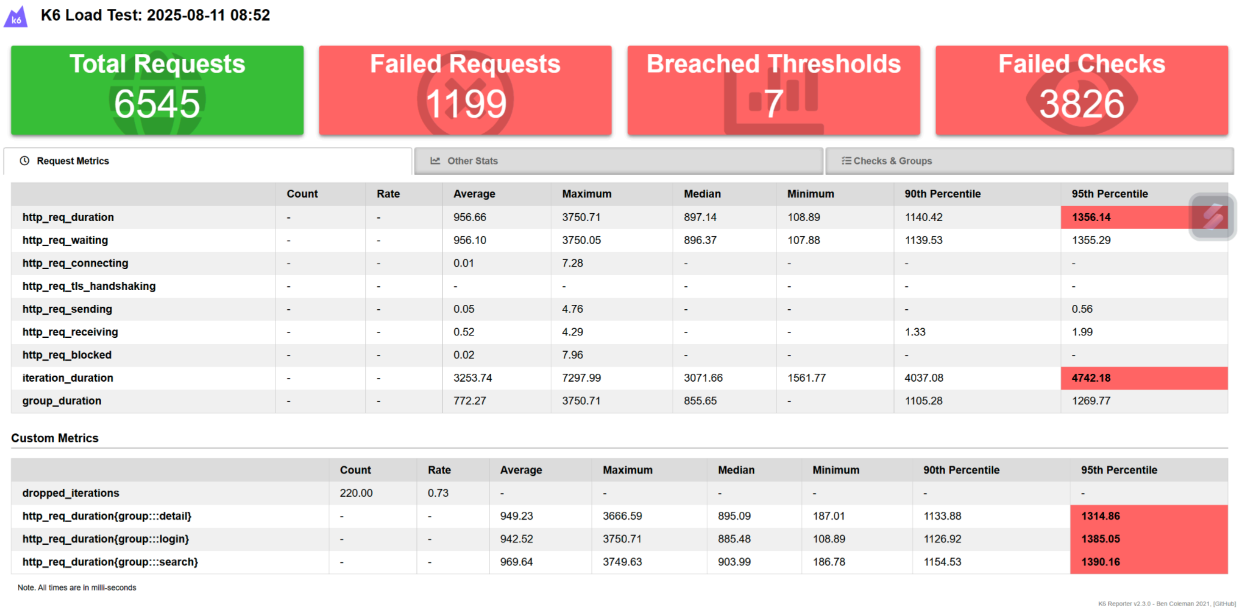
metric	count	rate
data_received	9.18 MB	30.3 kB/s

Rates

metric	rate
checks	0.71/s

Gauges

metric	value
vus	9



	Count	Rate	Average	Maximum	Median	Minimum	90th Percentile	95th Percentile
http_req_duration	-	-	956.66	3750.71	897.14	108.89	1140.42	**1356.14**
http_req_waiting	-	-	956.10	3750.05	896.37	107.88	1139.53	1355.29
http_req_connecting	-	-	0.01	7.28	-	-	-	-
http_req_tls_handshaking	-	-	-	-	-	-	-	-
http_req_sending	-	-	0.05	4.76	-	-	-	0.56
http_req_receiving	-	-	0.52	4.29	-	-	1.33	1.99
http_req_blocked	-	-	0.02	7.96	-	-	-	-
iteration_duration	-	-	3253.74	7297.99	3071.66	1561.77	4037.08	**4742.18**
group_duration	-	-	772.27	3750.71	855.65	-	1105.28	1269.77

Custom Metrics

	Count	Rate	Average	Maximum	Median	Minimum	90th Percentile	95th Percentile
dropped_iterations	220.00	0.73	-	-	-	-	-	-
http_req_duration{group:::detail}	-	-	949.23	3666.59	895.09	187.01	1133.88	**1314.86**
http_req_duration{group:::login}	-	-	942.52	3750.71	885.48	108.89	1126.92	**1385.05**
http_req_duration{group:::search}	-	-	969.64	3749.63	903.99	186.78	1154.53	**1390.16**

Note. All times are in milli-seconds

K6 Reporter v2.3.0 - Ben Coleman 2021, [GitHub]

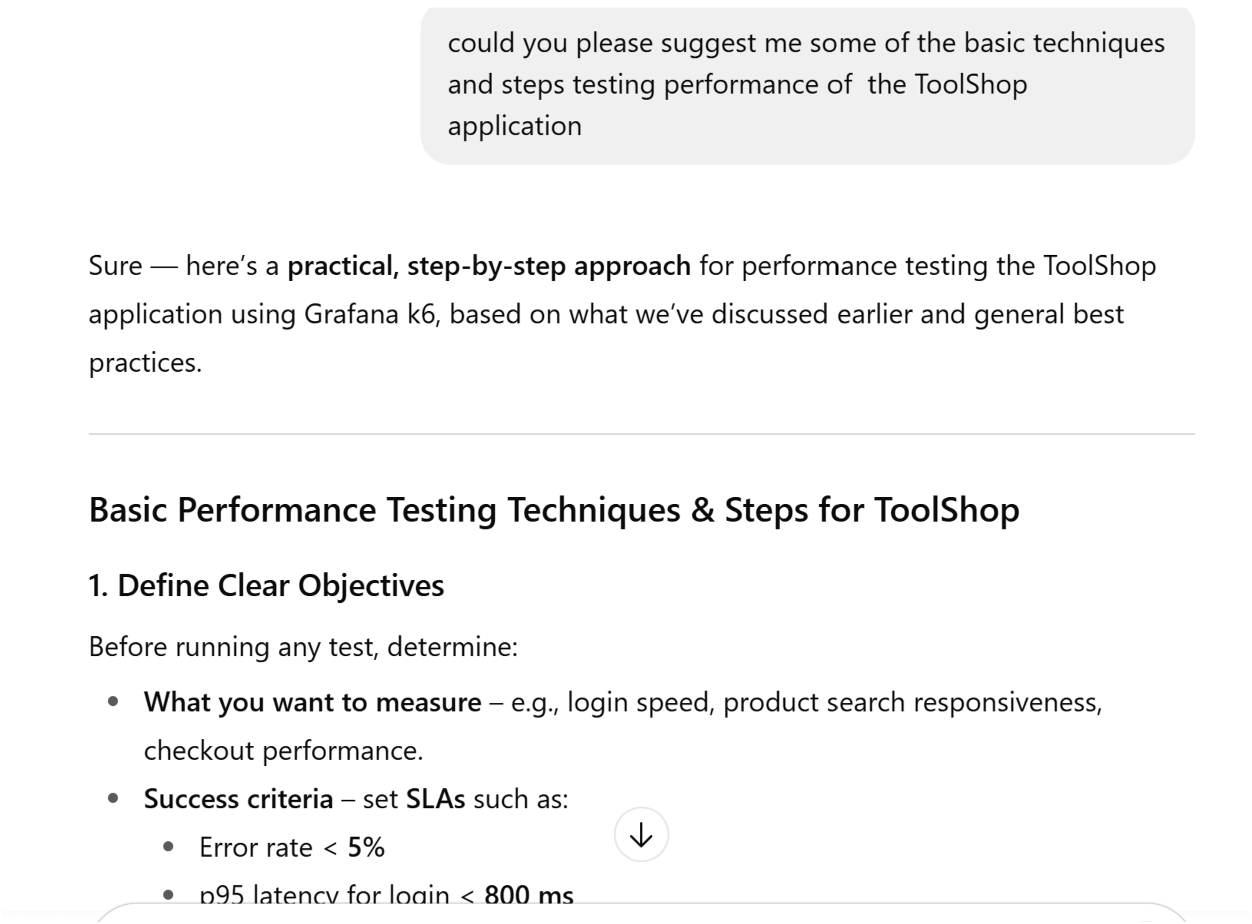
2.6 Results analysis

- The performance test results were evaluated against the predefined Service Level Agreements (SLAs) to determine whether the system meets expected performance under different load conditions. The analysis focused on two main aspects: **error rate** and **latency**.
- To have a deep understanding about the performance of the application on each test type, please follow this link: https://grafana.com/docs/k6/latest/using-k6/thresholds/

3. AI Promoting

- AI models: ChatGPT Plus
- Prompt: could you please suggest me some of the basic techniques and steps testing performance of the ToolShop application

- Results:



could you please suggest me some of the basic techniques and steps testing performance of the ToolShop application

Sure — here's a **practical, step-by-step approach** for performance testing the ToolShop application using Grafana k6, based on what we've discussed earlier and general best practices.

Basic Performance Testing Techniques & Steps for ToolShop

1. Define Clear Objectives

Before running any test, determine:

- **What you want to measure** – e.g., login speed, product search responsiveness, checkout performance.
- **Success criteria** – set **SLAs** such as:
  - Error rate < **5%**
  - p95 latency for login < **800 ms**

2. Identify Key Scenarios

Focus on realistic user flows in ToolShop:

- **Login** (JWT token authentication)
- **Product Search** (search by name from CSV)
- **Product Detail** (fetch product details by ID)
- **Checkout** (if implemented)

These flows should mimic actual user behavior and be scripted in k6.

---

3. Choose Appropriate Testing Types

Different test types help answer different questions:

1. **Load Testing** –

   Simulates expected daily usage (e.g., 30 VUs for 1–2 min steady).

   - Purpose: Verify system handles normal traffic without performance degradation.

2. **Stress Testing** –

   Gradually increases load beyond normal to find the **breaking point**.

   - Purpose: Determine max capacity before errors/slowdowns.

3. **Spike Testing** –

---

4. Prepare Test Data

- `login.csv` – multiple valid user accounts for authentication.
- `products.csv` – product names to search for.
- Make sure test data covers typical and edge cases.

---

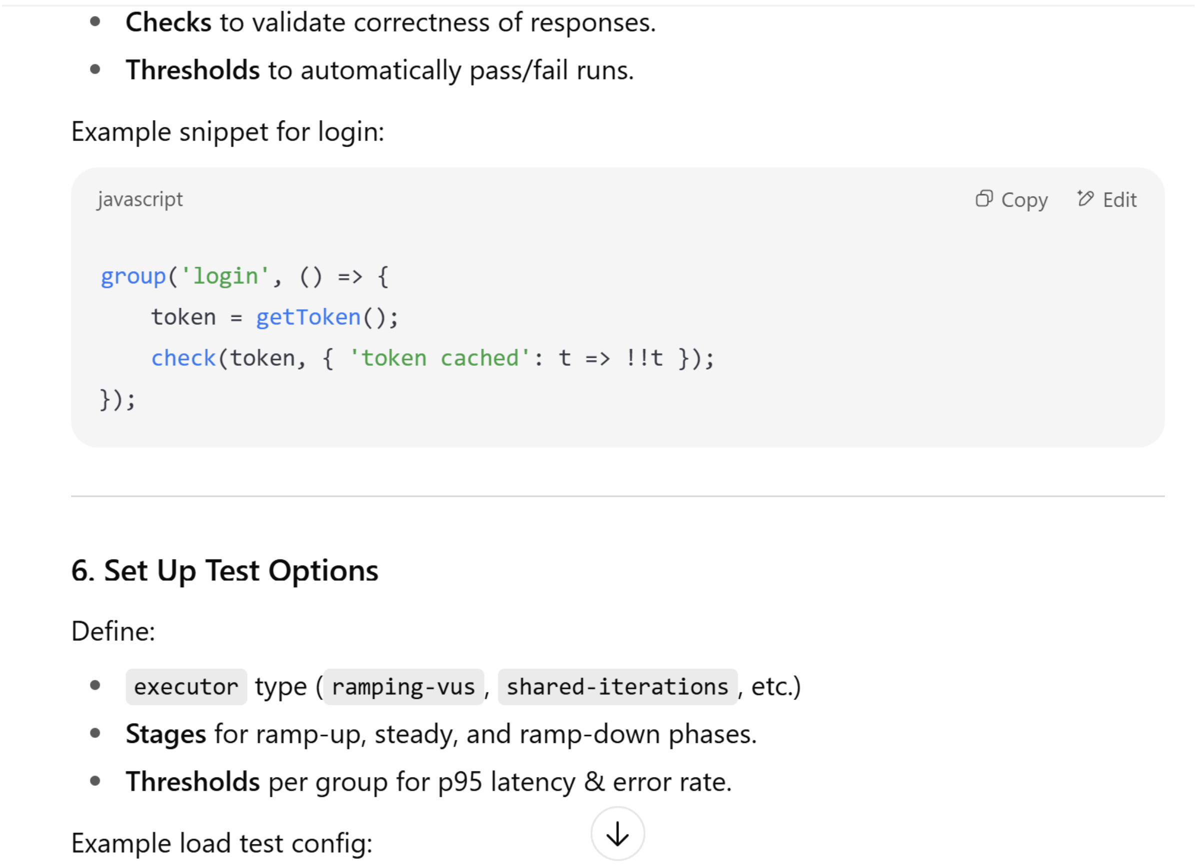
5. Script Your Tests

In **k6**, use:

- `SharedArray` to preload CSV data.
- **Groups** (`group('login', ...)`) for measuring each action separately.
- **Checks** to validate correctness of responses.
- **Thresholds** to automatically pass/fail runs.

Example snippet for login:

```javascript
group('login', () => {
    token = getToken();
    check(token, { 'token cached': t => !!t });
```



- **Checks** to validate correctness of responses.
- **Thresholds** to automatically pass/fail runs.

Example snippet for login:

```javascript
group('login', () => {
    token = getToken();
    check(token, { 'token cached': t => !!t });
});
```

6. Set Up Test Options

Define:

- `executor` type (`ramping-vus`, `shared-iterations`, etc.)
- **Stages** for ramp-up, steady, and ramp-down phases.
- **Thresholds** per group for p95 latency & error rate.

Example load test config:

4. Youtube link

The Youtube link demonstrating how to apply performance testing: https://youtu.be/Mi61qbh2D74

5. Self-assessment

Criteria	Description	Max Points	Self Assessment
Load testing	Missing any of the following “report”, “script”, or “video” results in 0 points	3.0	3.0

	Report: 1.0 TestCases, BugReport: 0.5 Script, Data: 0.5 Video: 1.0		
Stress testing	Missing any of the following “report”, “script”, or “video” results in 0 points Report: 1.0 TestCases, BugReport: 0.5 Script, Data: 0.5 Video: 1.0	3.0	3.0
Spike testing	Missing any of the following “report”, “script”, or “video” results in 0 points Report: 1.0 TestCases, BugReport: 0.5 Script, Data: 0.5 Video: 1.0	3.0	3.0
Use of AI Tools	Prompt transparency, critical validation, added value	1.0	1.0

Total		**10.0**	**10.0**