Workload Test Analysis

Project: Dummy JSON API Testing

Prepared by: Nada Abdelrahim

Date: 21-October-2025

Contents

1. Overview

This workload model is designed to simulate realistic user interaction patterns with the Dummy JSON API endpoints under moderate concurrent load. The configuration represents expected user traffic and response performance under typical usage conditions.

2. Workload Configuration

Virtual Users	5 → 20
Ramp-up Period	2 minutes
Test Duration	5 minutes
Think Time	1–3 seconds (random delay between requests)
Product Selection	Random product ID extracted dynamically
Environment	https://dummyjson.com

3. Justification of Workload Model

The selected workload model mirrors realistic user behavior for an e-commerce-style API environment. Each component of the model has been carefully chosen to show realistic interaction timing.

- ***Virtual Users (5 → 20)***

  Starting with 5 users and ramping up to 20 allows observation of performance scalability. This range represents typical concurrent usage levels for small-scale applications.

- ***Ramp-up Period (2 minutes)***

  Gradually increasing user load helps identify how the API performs as concurrent users steadily grow. This ensures the system has time to allocate resources efficiently.

- ***Duration (5 minutes)***

A 5-minute steady-state duration is sufficient for validation, allowing consistent measurement of average response times, throughput, and error rates.

- ***Think Time (1–3)***

Inserting random think time simulates realistic user pauses between requests, reducing the risk of overloading the system unrealistically and reflecting real-world browsing or interaction delays.

- ***Random Product Selection***

By dynamically extracting and using random product IDs, each virtual user interacts with different product data. This avoids caching effects and ensures the test captures genuine backend processing variability.

4. Conclusion

The configured workload model accurately represents expected user behavior patterns. It balances concurrency, pacing, and variability, providing meaningful insights into API performance and scalability. The combination of moderate concurrency, realistic think time, and randomization ensures that the system is tested under conditions close to real-world usage without introducing artificial load.