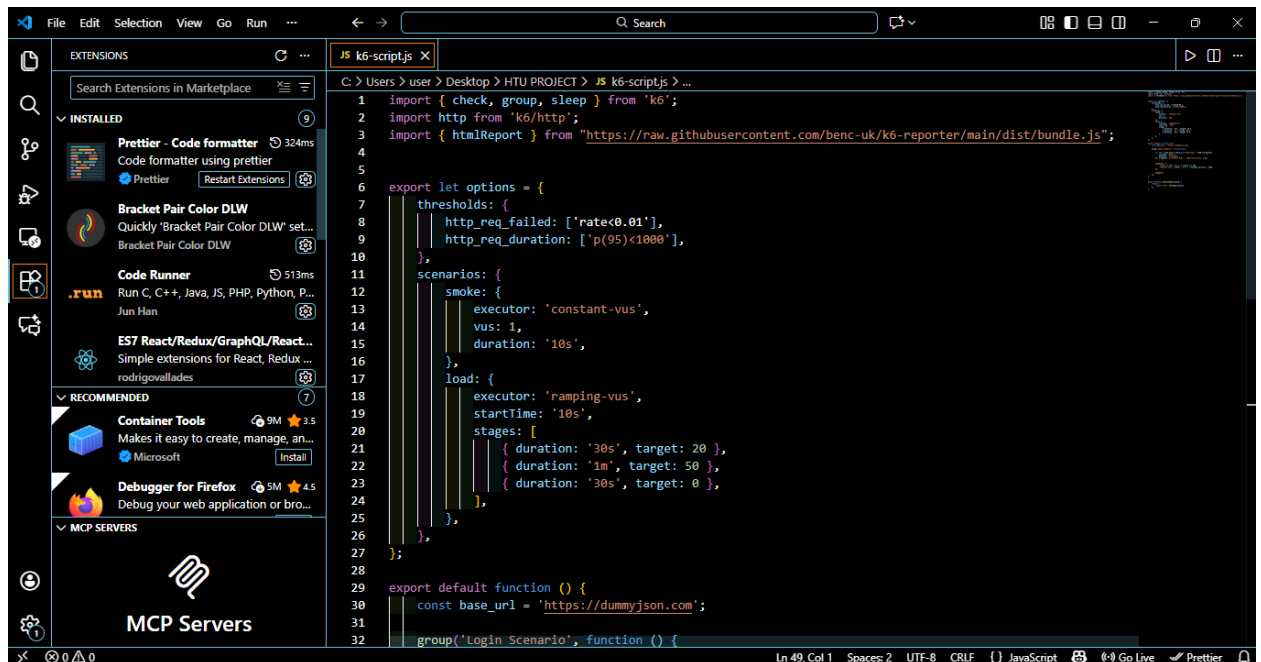


# Performance Test Analysis Report

## 1. Test Design and Profiles

The test was designed to evaluate the /auth/login endpoint using two distinct profiles to ensure both script validity and system stability under stress:

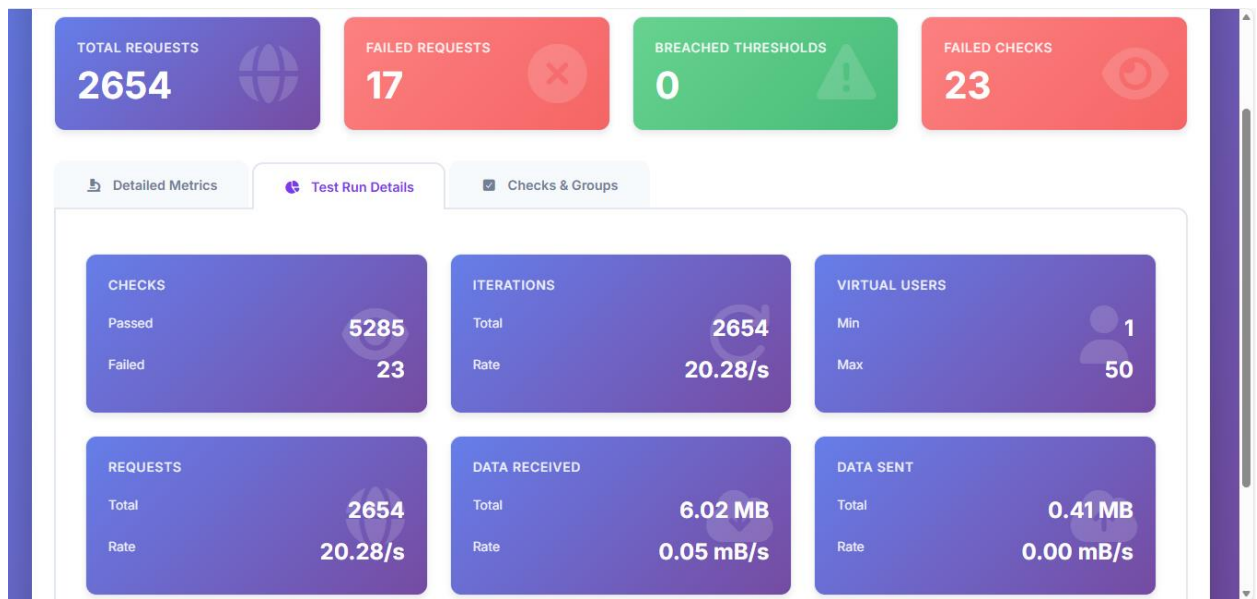
- **Smoke Test:** Conducted with **1 Virtual User (VU)** for **10 seconds** to verify the API's functional baseline.
- **Load Test:** A **Ramping-up strategy** was implemented, starting after the smoke test (at 10s mark). The load increased from **1 to 50 concurrent users** over a **2-minute duration** to observe the system's breaking point.

A screenshot of the Visual Studio Code editor interface. The left sidebar shows the 'EXTENSIONS' view with a list of installed and recommended extensions, including Prettier, Bracket Pair Color DLW, Code Runner, and Container Tools. The main editor area displays a JavaScript file named 'k6-script.js'. The script defines a performance test configuration with thresholds, scenarios, and a load profile. The scenarios section includes a 'smoke' test with 1 VU for 10s and a 'load' test with a ramping-up strategy from 1 to 50 VUs over 2 minutes. The load test is divided into three stages: 30s at 20 VUs, 1 minute at 50 VUs, and 30s at 0 VUs. The script also includes a default function to set the base URL to 'https://dummyjson.com' and a group for the 'Login Scenario'.

## 2. Key Performance Indicators (KPIs)

Based on the execution results, the following metrics were captured:

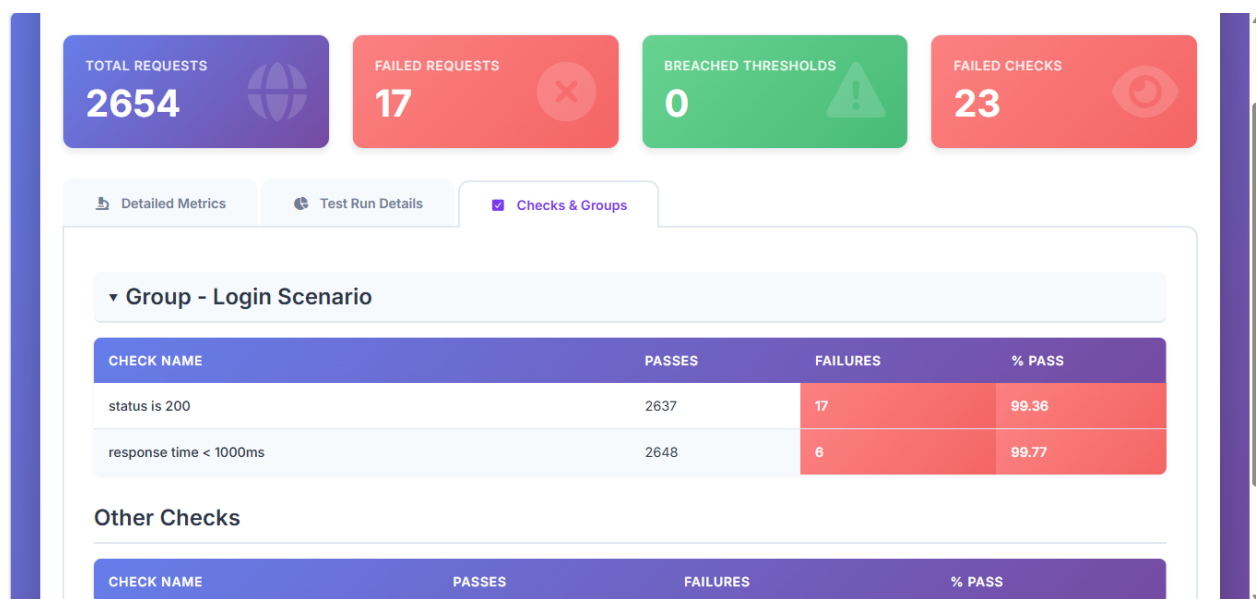
- **Throughput:** The system processed an average of **20.28 iterations per second**.
- **Average Response Time:** The mean latency was recorded at **190.93ms**.
- **95th Percentile (P95):** 95% of the requests were completed within **257.21ms**, indicating fast individual response times.
- **Error Rate:** A total of **23 checks failed** out of 2654 iterations, resulting in a failure count that appeared specifically during high-concurrency stages.



### 3. Assertions and Thresholds

We applied two primary assertions to measure success:

- **Response Time (< 1000ms): Passed 100%.** Every request was handled well within the defined time limit.
- **Status Code (200 OK): Failed 23 times.** As the load reached **50 VUs**, the server began returning non-200 status codes.



### 4. Analysis: Bottlenecks & Likely Causes

- **Identified Bottleneck:** The system suffers from a **Reliability Bottleneck** rather than a Latency Bottleneck. The server remains fast, but it becomes unstable and starts dropping requests once it exceeds a specific throughput threshold.

- **Likely Causes:**

1. **API Rate Limiting:** The target server (dummyjson.com) likely employs a rate-limiting policy that triggers when receiving more than **20 requests/second** from a single source, leading to the **23 failed checks**.
2. **Concurrency Saturation:** The backend authentication service may have a limited pool of worker threads available for token generation, causing it to reject new connections during the peak of the 50 VU stage.

3.

Trends & Times						
	AVG	MIN	MED	MAX	P(90)	P(95)
group_duration	1195.39	1157.05	1178.10	8367.33	1205.42	1260.16
http_req_blocked	3.38	0.00	0.00	7192.45	0.00	0.00
http_req_connecting	0.51	0.00	0.00	1014.42	0.00	0.00
http_req_duration	190.93	156.39	176.87	1252.18	199.09	257.21
http_req_receiving	1.96	0.00	0.15	47.82	6.88	10.33
http_req_sending	0.53	0.00	0.55	12.49	1.02	1.10
http_req_tls_handshaking	2.85	0.00	0.00	6167.24	0.00	0.00
http_req_waiting	188.43	155.19	174.87	1251.33	196.82	252.89
iteration_duration	1195.44	1157.24	1178.18	8367.33	1205.43	1260.16
% Rates						

## 5. Practical Improvement Recommendations

1. **Horizontal Scaling:** Deploy the authentication service across multiple nodes using a Load Balancer to distribute the 50+ concurrent user load more effectively.
2. **Retry Mechanism:** Implement an "Exponential Backoff" retry strategy in the client code to handle transient 429 (Too Many Requests) errors.
3. **Rate Limit Optimization:** Adjust the web server configuration (e.g., Nginx) to increase the allowed burst rate for peak traffic periods.
4. **Database Connection Pooling:** Optimize how the server manages concurrent database hits during the login process to prevent status code failures under pressure.