Performance Test Analysis Report

1. Test Design and Profiles

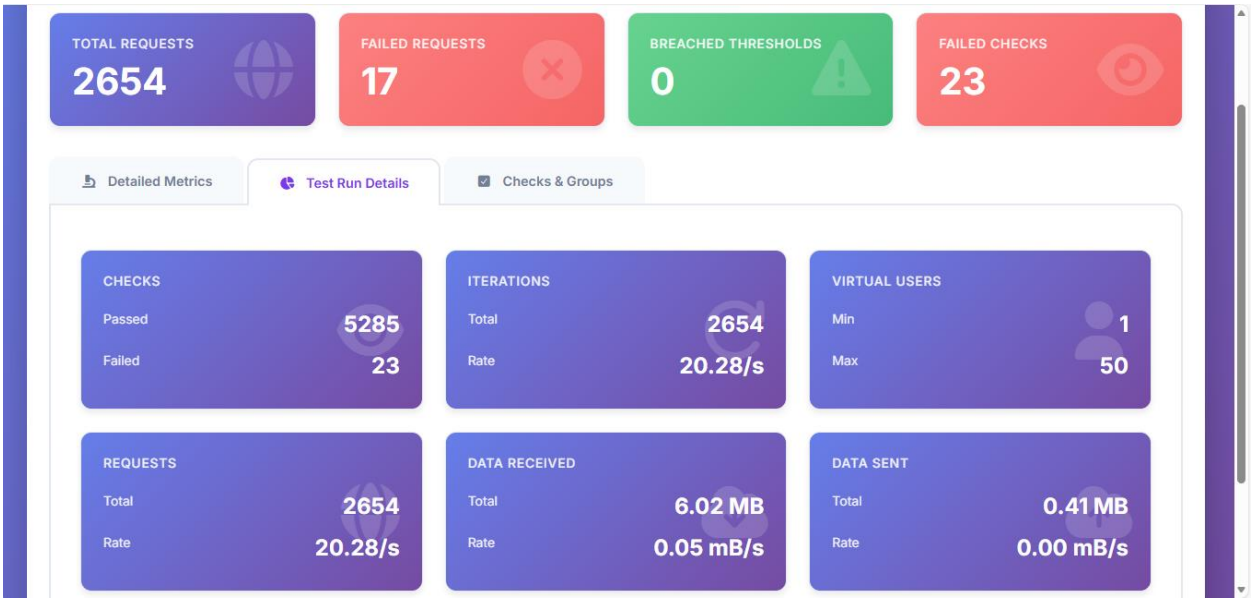
The test was designed to evaluate the /auth/login endpoint using two distinct profiles to ensure both script validity and system stability under stress:

- **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.

2. Key Performance Indicators (KPIs)

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

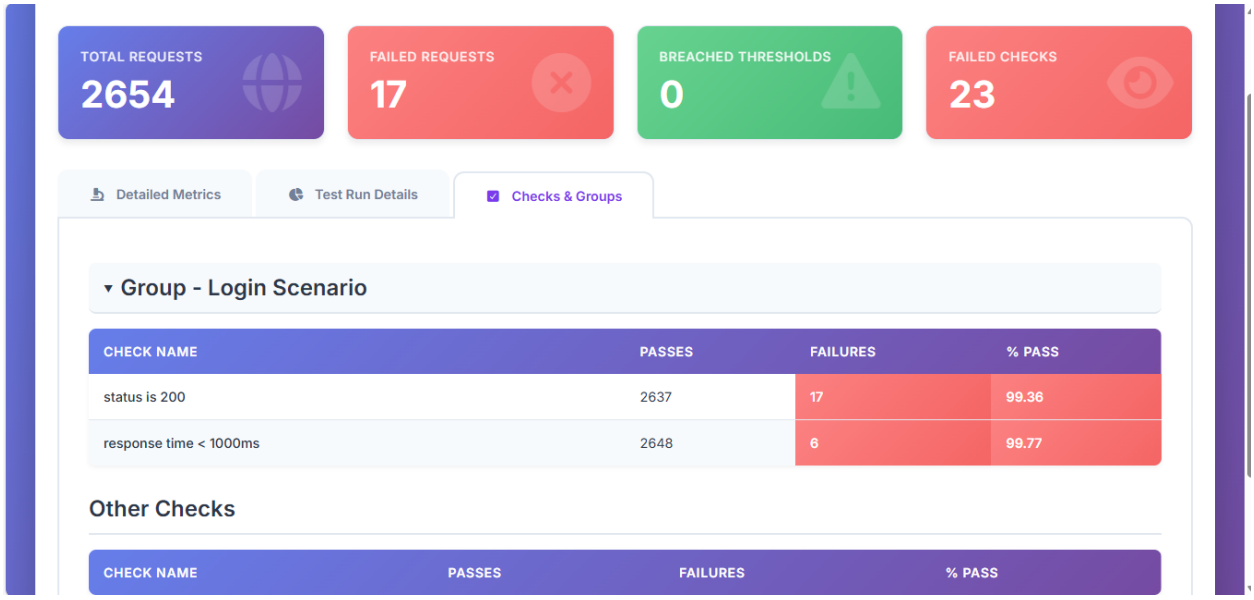
- **Throughput:** The system processed an average of **20.28 iterations per second**.
- **Average Response Time:** The mean latency was recorded at **190.93ms**.
- **95th Percentile (P95):** 95% of the requests were completed within **257.21ms**, indicating fast individual response times.
- **Error Rate:** 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. 

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.