Lightweight Performance & Resilience Test Plan

Goal

The goal is to get a quick sense of how stable and responsive the API is without putting any real load on it. We'll do a small-scale benchmark (something like 1 request per second for a minute) just to check response times and reliability. This approach stays well within public API limits and avoids triggering any rate limiting or ToS issues.

Approach

- Run a **micro-benchmark** with a single virtual user sending about **1 request per second for 60 seconds**.
- Stick to **read-only (GET)** endpoints only.
- Measure latency, success rate, and error trends.
- If the API doesn't allow load testing, just measure response times and errors from normal automated tests and use those numbers as your baseline.

What I'd Monitor

- **Success rate** – How many responses come back as 2xx (ideally ≥ 99%).
- **Latency (p95)** – The 95th percentile response time should ideally stay under 800ms.
- **Error budget** – The small percentage of time the API can fail before it's considered a problem (something like 0.5% of requests).
- **Stability** – No timeouts or large spikes in latency over time.

If error rates climb or response times start creeping up, that's a sign the service might be degrading under even light usage.

If Load Testing Isn't Allowed

- Skip the benchmark entirely and just record response times and status codes from your regular Playwright or Postman tests.
- Track those results over time to see if performance is trending up or down.
- The focus shifts from *stress* testing to *observing* performance drift and error rates.

What “Good” Looks Like

- 95% of requests finish under 800ms.
- Less than 5% of requests fail.
- No noticeable timeouts or throttling.
- The API remains stable across multiple runs.

Sample script in tests/parabank directory

npx artillery run performance.yml