STREAMAMG TEST STRATEGY

1. Test Vision & Quality Goals

StreamAMG's aims to deliver uninterrupted, broadcast-grade video at any scale.

My strategy targets four non-negotiables for every release:

Goal Success Metric Typical SLA

API correctness 100 % of content-management endpoints return valid schema & codes < 1 % contract failures per day

Streaming performance 95-percentile manifest & segment latency < 500 ms during live peaks

Device reach Pass rate across target browser device matrix > 98 %

Live-event stability Playback recovers from disruptions within 3 seconds Zero player hard-stalls

Assumptions For above decisions:

- Content management API follows REST conventions (e.g. GET /videos/:id, POST /videos, etc.)
- Streaming uses HLS (and/or DASH) manifests and segmented media file system.

2. Test-Pyramid Tailored to AMG OTT Systems

Manual/UAT for exploratory testing on real devices like Android and IOS with tools like postman and BrowserStack right from initial development stages.

Playwright E2E (Web + Native) for desktop, mobile, smart-TV testing

Playwright API / Integration testing to cover the different systems.

Why Playwright?

- One tool covers testing of browser, API, and mobile-web testing using the same language TypeScript runner.
- Built-in device emulation & project matrix means we can spin up Chrome, Safari, Firefox, WebKit and dozens of mobile presets from one config.
- Native APIRequestContext lets us hit REST endpoints without leaving the framework.

3. Automation Framework demo

Sample of my framework has been attached with a readme file to explain my decisions.

4. Performance Testing with JMeter

1. Demo site used (webflow.com) to demonstrate how I can use the tool to carry out testing on the actual

StreamAMG applications

2. Why JMeter? Easy to set up and run test both on UI and API. Little training needed even for less experienced QA

5. Multi-Device Compatibility

Tier Method Coverage

Desktop browsers Playwright projects (Chromium, WebKit, Firefox) Windows 10 + macOS

Mobile web Playwright device emulation + network throttling iPhone 13 and above, Pixel 7

Native/Smart-TV BrowserStack App Live & Automate with Playwright runner iOS, Android, Samsung Tizen, LG webOS

Tool Suggestion: BrowserStack provides real device & smart-TV grids, integrates directly with Playwright via browserstack/playwright CLI.

6. Live-Streaming Test Approach

Risk Automated Check

Manifest continuity Poll EXT-X-MEDIA-SEQUENCE; ensure +1 increment, no gaps.

Latency drift Compare server EXT-X-PROGRAM-DATE-TIME to system clock; assert live edge <= 10 s behind.

Network disruption In E2E test, carry out test on the playbacks for 5 s, then resume; expect playback recovery within 3 s.

DVR window After 5 min live, seek -60 s, play, then seekToLive().

Each scenario lives in tests/e2e/live.spec.ts and runs in Desktop-Chrome + iPhone-13 projects.

7. CI/CD Pipeline (GitHub Actions)

Stage Trigger Jobs

API Smoke push Playwright API quick suite

E2E Headless push Desktop Chrome

Cross-Browser nightly Full Playwright matrix

Performance nightly / pre-release Run JMeter, publish HTML report

Device Lab nightly BrowserStack desktop & smart-TV grid

8. Reporting & Observability

- Playwright HTML & trace viewer stored as build artifacts.

- JMeter HTML dashboard exported; 7-day latency trend in Grafana.
- Flakiness bot comments the historical pass-rate per test (Playwright test-retry stats).
- Slack alerts: red build, high latency, or manifest-gap detection.

9. Risk-Based Coverage vs Speed

Change Type Required Suites Build Time

Player UI tweak API + 1 E2E smoke ~3 min

API contract change Full API + E2E smoke ~6 min

Pre-match release All suites incl. cross-browser, JMeter ~25 min (nightly)

Fast feedback on every commit; exhaustive coverage nightly or before high-traffic live events.

10. Security Testing

Using zap tool and Burp suit run a security scan test before every release to targer and pinpoint all potential risk.

Read Me file has been attached with a quick break down of the security scan with Owasp Zap to detect common baseline security issues I have done on my own personal website with a scan report.

Outcome

Using Playwright-centric strategy gives StreamAMG one language, one test runner for API, web, mobile-web and smart-TV flows, augmented by JMeter for scale and BrowserStack for real-device reality-checks.

It addresses the platform's hardest problems mass-concurrency live events and device fragmentation, while keeping CI/CD feedback loops tight and maintainable.