Video Player Engineering Checklist — One-Page Reference

The player is the only piece of your platform the viewer touches. Settle these decisions before you ship. Engineering guidance; player-library behaviour changes per version — confirm against the current docs.

1 · MEDIA PIPELINE (the foundation)

- ☐ **Pick the engine, don't write one** — web: Shaka / hls.js / dash.js; iOS/tvOS: AVPlayer; Android/Fire TV: Media3.
- ☐ **MSE feeds the player** — segments append to a SourceBuffer; the media element plays from it (W3C MSE).
- ☐ **Match codecs to the device** — never select a rendition the device's decoder cannot play.
- ☐ **Evict watched video** — keep a window, not the whole title, or hit QuotaExceededError on TVs.

2 · ADAPTIVE BITRATE (ABR)

- ☐ **Choose an ABR family** — throughput, buffer-based (BOLA), or hybrid (most production players in 2026).
- ☐ **Start conservative, then climb** — a fast first frame beats a pristine first 3 seconds.
- ☐ **Damp the switching** — visible up/down churn looks worse than a steady, slightly lower level.
- ☐ **Respect device limits** — cap quality at what the screen and decoder can actually use.

3 · BUFFER + RECOVERY

- ☐ **Set the buffering goal** — ~20-30 s for VOD, less for live; it is how many seconds of bad network you hide.
- ☐ **Retry with backoff** — most CDN edge errors are transient; ask again before giving up.
- ☐ **Rendition fallback + gap jump** — re-fetch at another quality; skip a tiny missing-timeline hole.
- ☐ **Fatal is last** — reload from the manifest before any error screen. Never fatal on the first 404.

4 · DRM + MEASUREMENT

- ☐ **Drive DRM via EME** — Widevine / PlayReady / FairPlay through one EME path (W3C EME).
- ☐ **Encrypt once, licence many** — cbcs Common Encryption (ISO/IEC 23001-7) serves all three DRMs.
- ☐ **Budget the licence round-trip** — DRM adds latency before the first frame; measure it.
- ☐ **Beacon the four QoE numbers** — startup time, rebuffering, average bitrate, errors.

THE TEST — A MEDIA ELEMENT IS NOT A PLAYER

The HTML5 video tag (and a phone's built-in view) plays one fixed-quality file and, on Apple devices, native HLS — and nothing else. Manifest parsing, ABR switching, buffer tuning, segment recovery, multi-DRM via EME, and QoE beacons are all added by a player engine layered on top. Almost no OTT service writes that engine from scratch: web teams wrap Shaka, hls.js, or dash.js (or licence a commercial one), and mobile teams use AVPlayer and ExoPlayer/Media3. The real engineering is integration, per-device testing, and tuning — the buffering goal, the ABR family, and the recovery ladder — not the core switching maths, which lives in the Video Streaming section. The test of a finished player: on a flaky mobile connection it starts in under two seconds, recovers silently from a dropped segment, and reports every stall it could not hide. Re-verify player-library behaviour (Shaka bufferingGoal, dash.js ABR modes, Media3 naming) against the current docs before launch.