Design v1.1 — Pipeline-Aware Mapping

Project: AI-Enhanced Music Mashup Studio
Evolved from: Design v1 + Requirements v1.3
Principle: *If the pipeline cannot see it, it cannot evaluate it.*

---

1. Requirements-to-Components Mapping

Req ID	Requirement Summary	Component(s)	AI Involved?	Trust Risk
F1	Display BPM & key for selected tracks	TrackHeader, TrackMetadata, SpotifyService	No	Low
F2	Scrub forward/backward during playback	WaveformDisplay, AudioEngine	No	Low
F3	Simultaneous playback of ≥2 tracks	AudioEngine, TrackCard	No	Med
F4	Pause/resume/restart tracks independently	AudioEngine, TrackCard	No	Low
F5	EQ/volume changes isolated to selected track	BasicControls, CollapsibleEQ, AudioEngine	No	Med
F6	Runs entirely in browser, no hardware	App.js, AudioEngine (Web Audio API)	No	Med
F7	Save/load full project state	SaveButton, LoadButton, FirebaseService	No	High
F8	Persist config without audio files	FirebaseService, Firestore schema	No	Med
F9	Custom name/label for saved mashups	PlaylistNameEditor, FirebaseService	No	Low
F10	Recommend tracks by BPM + key	AIRecommendationEngine, SpotifyService	Yes	High

F11	Show reasoning behind AI recommendations	SuggestionCard, AIContext, AIRecommendationEngine	Yes	High
F12	AI must not auto-replace user tracks	AIPanel, AddButton, App.js	Yes	High
F13	Full manual function if AI fails	AIPanel (graceful degradation), MainWorkspace	Yes	High
F14	Allow disabling AI pitch/tempo adjustments	PitchControl, SpeedControl, AudioAdjustments	Yes	Med
F15	Near-real-time audio update on control change	AudioEngine, BasicControls, AudioAdjustments	No	Med
F16	Fetch Spotify metadata within 5s	SpotifyService, TrackSearchModal	No	Med
F17	Limit recommendations to 5–10	SuggestionList, AIRecommendationEngine	Yes	Low
F18	Visual feedback for audio changes	WaveformDisplay, VolumeSlider	No	Low
F19	Disclose AI bias to users	AIContext, SuggestionCard	Yes	Med
NF1	AI suggestions are optional, not authoritative	AIPanel, AddButton	Yes	Med
NF2	Warn on extreme pitch/tempo shifts	PitchControl, SpeedControl	No	Low
NF3	No third-party data sharing beyond Spotify	FirebaseService, SpotifyService	No	High
NF4	Prioritize responsiveness over audio fidelity	AudioEngine	No	Med
NF5	Notify user on API/network failures	SpotifyService, FirebaseService, error UI	No	Low
NF6	No copyright compliance guarantee	N/A — disclosure only	No	Med

⚠ **Requirements that cannot be satisfied without AI:** F10, F11, F12 (AI guard), F13 (AI fallback path), F17, F19, NF1

⚠ **Requirements likely to fail silently:** F12 (no UI block = silent bypass), F11 (explanation could be hardcoded), NF3 (data flow invisible to user), F13 (AI failure may not surface)

---

2. AI Entry Points

Component	AI Activity	Observable Signal	Risk
AIRecommendationEngine	Scores BPM/key/genre compatibility; ranks tracks	Compatibility score (0–100%) logged per suggestion	Wrong weights → misleading scores silently
SpotifyService.getRecommendations()	Calls Spotify's recommendation endpoint (external ML)	HTTP response code + returned track IDs logged	Spotify model bias; no introspection available
SuggestionCard	Renders AI-generated explanation text	Explanation string visible in UI + logged payload	Hardcoded/template text could misrepresent actual scoring
AIContext	Adapts context message to playlist state	Context string rendered in UI	Static fallback indistinguishable from dynamic reasoning
AIPanel	Triggers full recommendation pipeline	API call timestamp + result count logged	Silent empty result if AI fails and fallback not shown
AIRecommendationEngine.explainRecommendation()	Generates human-readable reason for match	Explanation string logged alongside score factors	Explanation may not match actual scoring factors used

3. Components-to-Pipeline Stages Mapping

Component	Pipeline Stage(s)	Why This Stage	Observable Behavior
App.js, AuthProvider	Build, Deploy	Root bundle; auth wiring must compile and be present at deploy	Build completes without error; app loads at deployed URL; Firebase auth redirect works
SpotifyService	Build, Test, Analysis	API call logic must compile, pass unit tests, and be lint-checked for correct error handling	Tests pass for API helper functions; ESLint reports no issues; HTTP response codes handled in coverage report
FirebaseService	Build, Test, Analysis, Deploy	CRUD must compile, be unit tested, lint-check for correct async patterns, and deploy with valid config	Tests pass for read/write helpers; no lint errors on async/await usage; Firestore operations succeed at deployment URL
AudioEngine	Build, Test	Web Audio API integration must compile, be unit tested for audio chain correctness and parameter changes	Build succeeds with Web API references; tests pass for audio node wiring, gain changes and playback control
AIRecommendationEngine	Build, Test, Analysis	Scoring logic must compile, be unit tested in isolation, and lint-checked for type safety on score calculations	Tests confirm score output in range $[0, 100]$; coverage report shows branch coverage on BPM/key sub-scores; ESLint passes on scoring functions
TrackCard, TrackSettings	Build, Test	UI rendering must compile and pass unit	Build includes components without error; tests pass for

		tests for control state correctness	settings, state persistence and render output
WaveformDisplay	Build, Test	Waveform rendering must compile and pass unit tests for data transformation from Spotify audio analysis	Build succeeds; tests pass for waveform data parsing and canvas/SVG render logic
AISidebar / AIPanel	Build, Test	AI display must compile, be tested for recommendation rendering, with fallback UI if recommendation empty/not found	Tests pass for suggestion card rendering from mock data; fallback message shown when AI returns empty array
SuggestionCard	Build, Test	UI component must compile and pass unit tests for correct score and explanation rendering	Build succeeds; tests confirm score is displayed and explanation string is non-empty
TrackSearchModal	Build, Test, Analysis	Search modal must compile and unit tests must verify Spotify query construction along with track-add callback fires	Build succeeds; tests pass for search input handling and add-track callback with correct track ID
SaveButton / LoadButton	Build, Test, Deploy	Persistence logic must compile, round-trip serialization must pass unit tests, and save/load must function at deployed URL	Tests pass for data integrity (track IDs, EQ, pitch, volume match after round-trip); saved state fully restored at deployed URL
Header	Build, Deploy	Navigation must be compiled and rendered at deployed URL (can include default browser)	Build succeeds; header visible at deployed URL with correct layout and no z-index breaks

---

4. Candidate Trust Gates

Gate G1 — AI Score Sanity Check

- **Location:** Test stage (unit) + Analysis stage (lint)
- **Checks:** analyzeCompatibility() returns a number in $[0, 100]$; never NaN or null; BPM sub-score and key sub-score are each present
- **Evidence Collected:** Score value, sub-scores (BPM, key, genre), input track IDs, timestamp
- **Initial Threshold:** score $\in [0, 100]$; sub-scores sum to $\approx 100\%$ weight [PLACEHOLDER — needs calibration during implementation]
- Pipeline enforces gate with:
    - **Test:** Unit tests assert score output is a number in [0, 100] for valid inputs; assert NaN/null returns are caught; assert BPM and key sub-scores are both present in output object; branch coverage confirms all scoring paths exercised
    - **Analysis:** ESLint verifies type safety on score arithmetic (no implicit string-to-number coercion); catches unhandled edge cases in scoring functions
- **Failure Action:** Test stage fails the pipeline if assertions break; at runtime, substitute score = 0 and do not surface recommendation
- **Linked to:** F10, F11, NF1; AIRecommendationEngine; AI Entry Point row 1

Gate G2 — Recommendation Count Boundary

- **Location:** Test stage (unit)
- **Checks:** generateRecommendations() returns between 3 and 10 suggestions; empty result triggers fallback UI
- **Evidence Collected:** Result count, Spotify API response code, engine input (seed tracks)
- **Initial Threshold:** count $\in [3, 10]$ [ARBITRARY — 3 minimum assumed for UX; adjust after user testing]

- Pipeline enforces gate with:
    - **Test:** Unit tests mock Spotify API responses with varying result counts $(0, 3, 10, 15)$ and assert: count of 0 triggers fallback UI render; count within $[3, 10]$ renders suggestion list; count > 10 is truncated to 10; coverage report confirms fallback branch is exercised
- **Failure Action:** If count $= 0$, show "No suggestions available" UI (not silent); if count > 10, truncate and log; test stage fails pipeline if fallback path is not covered
- **Linked to:** F13, F17; AIPanel, SuggestionList; AI Entry Point rows 4–5

Gate G3 — Explanation Presence Check

- **Location:** Test stage (unit) + Analysis stage (lint)
- **Checks:** explainRecommendation() returns non-empty string; string references at least one of: BPM, key, genre
- **Evidence Collected:** Explanation string, triggering score factors, presence of required keywords
- **Initial Threshold:** String length > 10 chars; contains at least one of ["BPM", "key", "$genre$"] [WEAK CHECK — improve with validation]
- Pipeline enforces gate with:
    - **Test:** Unit tests assert explanation string is non-empty for every valid recommendation; assert at least one keyword ("BPM", "key", "genre") is present in output; test with edge-case inputs (identical BPM, unknown genre) to confirm explanation still generates
    - **Analysis:** ESLint validates that explainRecommendation() always returns a string type, not undefined or null
- **Failure Action:** Test stage fails pipeline if explanation is empty or missing keywords; at runtime, display generic fallback text flagged as "generic" in UI
- **Linked to:** F11, F19; SuggestionCard, AIContext; AI Entry Point row 3

Gate G4 — AI Non-Replacement Guard

- **Location:** Test stage (unit) + Analysis stage (lint)

- **Checks:** Track stack state does not change unless user explicitly clicks "+ Add to Mix"; no automatic track insertion/reorder by AI code paths
- **Evidence Collected:** Track state before/after AI panel interaction; action origin (user click vs. code)
- **Initial Threshold:** Track array unchanged after recommendation fetch unless add action logged [PLACEHOLDER — requires event-sourced audit log]
- Pipeline enforces gate with:
    - **Test:** Unit tests render AIPanel with mock recommendations, call generateRecommendations(), then assert track state array is unchanged (same length, same IDs, same order); separate test simulates user clicking "+ Add to Mix" and asserts track is appended only then
    - **Analysis:** ESLint can be configured to flag if AI module files directly import track-state setter functions (e.g., setTracks, dispatch({type: ADD_TRACK})), enforcing that only user-action handlers modify track state
- **Failure Action:** Test stage fails pipeline if track state mutates without simulated user click; flag for immediate human review
- **Linked to:** F12, NF1; AIPanel, App.js; AI Entry Point row 4

Gate G5 — Spotify API Failure Visibility

- **Location:** Test stage (unit)
- **Checks:** Any Spotify API call returning non-2xx triggers visible user notification; silent fallback without notification is a failure
- **Evidence Collected:** HTTP status code, endpoint called, error message, user-facing notification fired (boolean)
- **Initial Threshold:** 100% of non-2xx responses must produce a user-visible error OR explicit silent-failure log entry [ARBITRARY — 100% may be relaxed for transient 429s]
- Pipeline enforces gate with:
    - **Test:** Unit tests mock SpotifyService responses with 401, 403, 429, and 500 status codes; assert that error notification component renders for each; assert that silent empty-state (no error shown, no data shown) does not

occur; coverage report confirms all error-handling branches in SpotifyService are exercised

- **Failure Action:** Test stage fails pipeline if any non-2xx mock produces no error UI; at runtime, log error, show user notification, do not silently degrade
- **Linked to:** NF5; SpotifyService, AIPanel, FirebaseService; AI Entry Point row 2

Gate G6 — Audio Degradation Warning Gate

- **Location:** Test stage (unit)
- **Checks:** When pitch $> \pm 3$ semitones OR speed outside $[0.85, 1.15]$, warning UI fires before audio applies
- **Evidence Collected:** Pitch/speed value at time of warning; whether user acknowledged or dismissed
- **Initial Threshold:** Thresholds as stated in NF2 [PLACEHOLDER — $\pm 15\%$ tempo $= \pm 0.15$ multiplier, mapped to $[0.85, 1.15]$]
- Pipeline enforces gate with:
    - **Test:** Unit tests set pitch to ± 4 semitones and assert warning component renders; set speed to 0.80 and 1.20 and assert warning renders; set pitch to ± 2 and speed to 1.0 and assert warning does NOT render; coverage confirms both the warning and no-warning branches are exercised
- **Failure Action:** Test stage fails pipeline if warning not rendered when threshold exceeded; at runtime, log if warning not shown despite threshold exceeded
- **Linked to:** F14, NF2; PitchControl, SpeedControl; no direct AI entry point (human-triggered)

5. Failure Visibility Analysis

Component	Silent Failure Scenario	Pipeline Detects?	Instrumentation Needed
AIRecommendationEngine	scoreByKey() returns 0 for all tracks due to key format mismatch (e.g., "Am" vs. "A minor"); all scores artificially low; user sees bad suggestions with no error	Partially — G1 unit tests catch this only if test inputs include varied key formats ("Am", "A minor", "am", "a"). If test data uses a single format, mismatch goes undetected.	Add unit test cases with all Spotify key format variants (short and long form); assert key normalization function handles each; assert score > 0 for known-compatible key pairs. G1 provides the assertion framework; test data coverage is the gap.
AIPanel	Spotify getRecommendations() returns 200 but empty array; SuggestionList renders nothing; no error shown	Yes — G2 unit tests mock an empty array response and assert fallback UI ("No suggestions available") renders instead of silent empty state. Coverage report confirms fallback branch is exercised.	G2 covers this path. Additionally, log empty-result events at runtime with seed track IDs for debugging why Spotify returned no results (valid API response but unhelpful content).
Suggestion Card	explainRecommendation() returns template string not tied to actual score factors; user shown misleading reasoning	Partially — G3 checks keyword presence ("BPM", "key", "genre") and non-empty length but cannot verify that the explanation semantically matches the actual sub-scores used.	G3 provides baseline string validation. To strengthen: unit tests should pass score sub-factors as input and assert the explanation string contains the specific values (e.g., if BPM diff = 2, explanation should include "2" or "similar

			BPM"), not just generic keywords.
FirebaseService	savePlaylist() resolves successfully but writes partial data (segment array truncated by Firestore limit); load returns incomplete state	Partially — SaveButton/LoadButton unit tests perform round-trip serialization (save then load), but only if tests include playlists with maximum track/segment counts.	Add unit test case with maximum payload: 5 tracks, each with multiple segments and full settings (EQ, effects, pitch, speed). Assert saved field count equals loaded field count.
SpotifyService	Token refresh fails silently; subsequent API calls return 401 but error swallowed in catch block; app appears to work but fetches stale/no data	Yes — G5 unit tests elicit 401 responses and assert error notification component renders. Coverage report confirms all catch blocks in SpotifyService surface errors to the UI.	G5 covers this path. Additionally, Analysis stage (ESLint) can flag empty catch blocks (catch(e) {}) in SpotifyService via the no-empty rule, preventing silent error swallowing from entering the codebase.

---

6. Gap Identification

Components with no clear pipeline stage:

- DragHandle.js — pure UI, no test coverage specified
- AddSegmentButton.js — no unit test for segment-add state mutation or downstream Firestore write path

AI entry points with no trust gate:

- SpotifyService.getRecommendations() (external ML model) — no gate on *what Spotify returns*, only on downstream count (G2)
- AIContext.js context message adaptation — no gate on whether context message is accurate to playlist state

Trust gates with no evidence source:

- G4 (Non-Replacement Guard) requires event-sourced audit log that does not currently exist in design

Requirements with no component mapping:

- NF6 (copyright disclaimer) — disclosure only; no component surfaces this in the UI; needs a DisclaimerBanner or equivalent
- F19 (bias disclosure) — partially covered by AIContext but no dedicated component or logged event

Pipeline stages with no trust gates:

- **Deploy stage** — no gate before pushing to staging/production (e.g., smoke test, auth sanity check).
- **Build stage** — no gate on bundle size or Web Audio API compatibility check.
- **Analysis stage** — indirectly participates in G1, G3, and G4 safety checks, but doesn't have its own trust gate (e.g., no lint rule threshold for maximum warnings).

---

7. Red Flags Assessment

RF1 — AI explanation text is cosmetic, not derived
explainRecommendation() could return template strings ("Great harmonic match!") disconnected from actual scoring. Gate G3 is a weak mitigaton.
Remediation: Log score sub-factors alongside explanation string; add test asserting explanation content correlates with top-weighted factor.

RF2 — No audit trail for AI non-replacement (F12/NF1)
There is no event log proving that track state changes originate from user actions, not AI code paths. Gate G4 cannot function without this.
Remediation: Implement an action-origin field on all track state mutations (values: "user" | "ai_suggestion" | "system"). G4 unit tests can then assert that after generateRecommendations() is called, no mutations with origin "ai_suggestion" exist without a preceding user click event.

RF3 — Spotify external ML is a black box with no trust gate
getRecommendations() delegates to Spotify's own model. The design has no gate on what that model returns — biased, irrelevant, or low-quality results pass through silently.
Remediation: Add a post-fetch validation step in AIRecommendationEngine that re-scores

Spotify results using analyzeCompatibility() and filters out suggestions below a minimum threshold before display.

RF4 — Firebase save "success" is not a correctness signal
savePlaylist() resolving does not guarantee data integrity (partial writes, schema drift). This is a silent failure risk for F7/F8.
Remediation: Write a round-trip unit test (save -> load -> diff) in the Test stage that uses a maximum-size fixture (5 tracks, multiple segments, full settings). Assert that every field saved equals every field loaded. Consider adding a fieldCount metadata property written alongside the playlist document that is validated on load.

RF5 — Deployment stage has no trust gate
There is no defined smoke test or go/no-go check before production deployment. Any broken auth, blank AI panel, or missing Spotify credentials would be caught only by users.
Remediation: Add a post-deploy verification step to deploy-production job. After deployment completes, use a curl that: (1) fetches the deployed URL and asserts HTTP 200, (2) confirms the HTML response contains the root app element. A full test (auth flow, Spotify token exchange, AI panel render) is desirable but may exceed scope.