INDEPENDENT CODE & PRODUCT REVIEW / 3 NOVEMBER 2026

VoiceDebrief MeetScribe Security, QA, code quality & UX

Verdict: keep this as a supervised beta. Fix the privacy and data-integrity defects before widening distribution. The architecture has useful safeguards, but passing tests currently overstate confidence in the full user journey.

Reviewed **v0.3.1**, default branch **dev**, commit **a4bbdca62a46**. The review began at v0.3.0 and incorporated both subsequent v0.3.1 commits. All 18 findings below were checked against this final snapshot. The live site was inspected at v0.3.1. No repository changes, issues, PRs, real meeting recordings, or paid API calls were made.

Area	Assessment
Security & privacy	Narrow extension permissions and safe text rendering. Deletion, mute boundaries and retention wording need correction.
QA & reliability	45 tests pass in CI, but production tab capture and real transcription remain unverified in this review. Failure and lifecycle coverage is incomplete.
Data integrity	Wrong-meeting debrief writes, invisible missing speech and heuristic removal of valid replies are actionable defects.
Code & release quality	Small modular codebase with no runtime package dependencies; good ZIP/source checks. Async state ownership and provenance need work.
UX	The v0.3.1 download and setup flow improve access. The key test/save mismatch blocks new users, and meeting navigation is not keyboard accessible.

Evidence collected

Locally: **30/30 unit tests passed**; **15 focused code-level observations reproduced**, with Chrome/media/provider/storage services stubbed where needed. CI logs confirm **30 unit + 14 extension + 1 release test** passed. Both repository ZIPs are identical, their checksum matches, and all **27 packaged files match extension/ byte for byte**.

The local browser suite could not start because the Chromium executable is absent. This is an environment limitation, not a failed application assertion. No claim is made here to have tested an actual Meet call, the user's OpenRouter account, real microphone hardware, browser restart behavior, screen readers, or a two-hour session. Focused reproductions establish code behavior, not production incidence.

Priorities: P1 = address before broader testing; P2 = fix for a dependable beta. Eight P1 findings: 01, 02, 04, 05, 07, 08, 10 and 11. No critical remote-code-execution or credential-exfiltration path was established. This is not a penetration-test certification.

Verified CI run | Reviewed repository snapshot | Live site

01 / PRIVACY BOUNDARIES

Deletion, microphone controls and retention

01 / P1 / Deleting a running meeting does not stop its capture

Reproduced application rule with an in-memory transaction adapter; UI path confirmed in source.

Delete meeting and Delete all clear stores without coordinating with the recorder or queue. The offscreen recorder still holds the meeting ID and can write the next audio segment. addSegment does not require an existing parent meeting. A reproduction left **0 meetings, 1 segment and 1 WAV**; that orphan was claimable for upload. Existing requests are not cancelled either.

Fix: Coordinate deletion through the service worker: stop capture and dispatch, fence the session with a deletion marker, wait for pending writes, then delete. Reject writes and sends for deleted meetings. Explain that a local delete cannot recall an upload already received by a provider.

Acceptance: Delete during active capture and during an in-flight send. After acknowledgement, no new local records or uploads may appear for that meeting.

extension/options.js:99 | extension/meetings.js:122 | extension/lib/db.js:63 | extension/offscreen.js:28

02 / P1 / Buffered muted audio can be admitted after unmute

Reproduced using the actual worklet and channel handler with a fake audio graph.

The worklet batches 2,048 samples; pause/mute is evaluated only when the whole batch reaches offscreen.js. If unmute occurs partway through a batch, earlier muted samples are admitted with later audio. The reproduction admitted **1,920 muted samples (120 ms at 16 kHz)**. The same boundary problem applies to Resume. Speech qualification may later discard a short segment, but the privacy gate has already accepted the samples.

Fix: Apply gating at the worklet/sample boundary. Flush or discard partial buffers on each state transition and acknowledge the effective boundary to the UI. Do not rely on message-delivery time to determine when audio was captured.

Acceptance: Place a distinctive tone entirely inside mute/pause intervals, including just before unmute/resume. Assert it is absent from every persisted WAV.

extension/worklet.js:3 | extension/offscreen.js:43 | extension/offscreen.js:132

03 / P2 / Privacy settings overstate their guarantees

Confirmed from source and current official OpenRouter/Chrome documentation; account policy was not inspected.

The UI says zero-data-retention providers only, but current OpenRouter documentation says **zdr:true follows account/guardrail Safety Retention settings**; self-serve defaults can allow Safety Retention. Locally, the audio retention option **immediately** is only serviced by a six-hour alarm. Failed audio is kept indefinitely, and Delete all keeps the key/settings. These boundaries deserve explicit wording.

Fix: Explain and link the account-level ZDR-only choice and policy exceptions. Apply zero-day retention at successful settlement and when changed. Reconcile alarms on worker startup, especially for supported Chrome versions before 150. Label the delete scope precisely.

Acceptance: Verify the effective provider-policy configuration; assert zero-day audio removal in the settlement path, and retention recovery after restart.

extension/options.html:35 | extension/options.html:54 | extension/service-worker.js:127 | extension/lib/db.js:159

02 / LIFECYCLE

Stop, recovery and state ownership

04 / P1 / Stopping offline can block the next meeting indefinitely

Reproduced queue behavior with a controlled clock/provider; full stop path traced in source.

Stop waits for an untilEmpty drain before reporting finished. Offline errors leave segments pending and retry forever, so the offscreen document and service-worker session remain draining. Start only accepts idle. Audio capture stops, but the user cannot start another recording until the queue clears. An indefinitely pending fetch can also hold Stop because the abort signal is not passed to fetch.

Fix: Separate recorder state from transcription state. Once streams stop and writes finish, release the capture session and hand queued work to a resumable scheduler. Add request deadlines and cancellation, and show stopped with uploads pending.

Acceptance: Go offline with pending speech, press Stop, and immediately start a second meeting. The first must remain recoverable and must not monopolize recording.

extension/offscreen.js:104 | extension/lib/queue.js:98 | extension/lib/openrouter.js:32 | extension/service-worker.js:32

05 / P1 / Fixing a rejected key does not resume live transcription

Reproduced drain exit and coordinator no-op with stubs; options path confirmed in source.

A key/credit error exits the live drain loop. Saving a new key requeues held segments and asks the service worker to drain, but kickDrain immediately returns while an offscreen document exists. There is no message to restart that document's completed drain. The UI says held segments will be sent even though live capture can keep accumulating unsent audio.

Fix: Give queue ownership an explicit resume operation. Keep a cancellable supervisor alive or restart the offscreen drain on settings recovery. Treat key/credit state globally so one request failure does not leave inconsistent runners.

Acceptance: Reject the key during capture, save a valid replacement without stopping, and assert pending segments resume while capture continues.

extension/lib/queue.js:103 | extension/options.js:46 | extension/service-worker.js:118 | extension/offscreen.js:99

06 / P2 / Failed startup and missing-recorder stops strand session state

Reproduced coordinator behavior with rejected Chrome API calls.

start writes starting before ensureOffscreen/sendMessage, but its cleanup covers only an unsuccessful reply, not a thrown call. A rejected createDocument leaves starting; a second start returns already starting. Stop then changes state to draining and swallows the missing-receiver error, with no finished event to recover it. Pause/mic handlers also update coordinator state without consistently checking success.

Fix: Serialize session transitions, wrap startup in rollback/finally, and validate session identity on every command/acknowledgement. Reconcile recorder existence when a message fails. Catch the asynchronous stop task and persist an actionable failure state.

Acceptance: Inject createDocument, sendMessage and persistence failures; lose the offscreen context; double-click Start/Stop. Every path must reach a truthful, recoverable state.

extension/service-worker.js:32 | extension/service-worker.js:55 | extension/service-worker.js:77 | extension/offscreen.js:131

03 / TRANSCRIPT INTEGRITY

Missing speech and unreliable inference

07 / P1 / Storage failures silently discard speech

Reproduced the segment writer rejecting with QuotaExceededError; UI/error path confirmed in source.

The segment write chain catches a database failure and appends a string to S.errors, then resolves. No segment or gap is persisted. Those errors are sent in level messages, but the panel reads only meter values. Capture continues and the meeting can finish as complete. The user therefore loses speech without the promised visible gap; an output-device failure also goes through the ignored error channel.

Fix: Treat persistence failure as a critical recorder fault: pause/stop safely, surface a persistent warning, retain a bounded retry buffer if feasible, and record the affected time span. Completion must distinguish incomplete capture from complete transcription.

Acceptance: Fail a segment transaction and verify an immediate visible failure, stopped or explicitly degraded capture, and a durable missing-audio interval after reload.

extension/offscreen.js:28 | extension/offscreen.js:100 | extension/sidepanel.js:119

08 / P1 / Empty transcription is hidden as successful completion

Reproduced through the real parser, line builder and transcript assembler.

A provider response {text:""} is accepted. The queue marks the segment done with empty:true; assemble omits every done segment lacking lines. The reproduction produced **zero lines and zero timeline items** for an existing audio segment. It cannot be manually retried through retryFailed, and its audio becomes eligible for successful-audio retention. An empty result is not proof that the captured speech was silence.

Fix: Use an explicit no-transcript/review-needed state, retain the audio and expose playback/retry. Distinguish detected silence from speech that the provider returned empty. Include the missing span in debrief input.

Acceptance: Return empty text for non-silent audio. The UI, exports and debrief input must all preserve a visible, retryable gap.

extension/lib/openrouter.js:70 | extension/lib/queue.js:44 | extension/lib/transcript.js:24 | extension/lib/transcript.js:57

09 / P2 / The echo heuristic removes legitimate replies from debriefs

Reproduced with an ordinary question-and-answer pair.

Word-set overlap within a broad time window is enough to mark probable-echo. "Can you send the report today?" followed by "I will send the report today" is flagged. transcriptForPrompt then removes the reply entirely. The transcript still contains the line, but the debrief loses the speaker's commitment. This is particularly damaging for decisions and action ownership.

Fix: Keep probable echoes as uncertain evidence unless acoustic evidence or explicit user confirmation supports exclusion. Use timing, duration and audio correlation, not word containment alone. Preserve source-line IDs in generated claims.

Acceptance: Test genuine repetition, confirmations, short answers, interruptions and acoustic echo. A legitimate commitment must remain available to the debrief.

extension/lib/transcript.js:36 | extension/lib/transcript.js:49 | extension/lib/debrief.js:7

04 / USER JOURNEYS

Async identity, onboarding and accessibility

10 / P1 / A debrief can be saved into the wrong meeting

Reproduced with actual meetings.js, a fake DOM/database and a delayed chat response.

writeDebrief reads the selected meeting's transcript, awaits chat, then writes through the mutable global current. The meeting list remains clickable. Generate for meeting A, select B before the response arrives, and A's debrief is saved on B. This can contaminate an unrelated export or overwrite B's earlier debrief. The reproduction saved "Summary of A" to **B**.

Fix: Capture an immutable meetingId and transcript revision before any await. Pass that ID through reads and writes, and keep per-meeting request state. Check that the target still exists before committing; a navigation must not change the destination.

Acceptance: Delay chat, switch meetings, delete the original, and generate from two tabs. Results must only attach to their originating meeting/revision.

extension/meetings.js:28 | extension/meetings.js:92

11 / P1 / The published quick start tests the key without saving it

Confirmed live v0.3.1 instructions and source behavior.

The homepage and README tell a new user to paste a key and press Test the key. That handler validates the input but never stores it. Only Save writes apiKey. After a successful test, the panel still disables Start because stored settings have no key. The longer installation guide includes Save, so the two journeys disagree.

Fix: Prefer a single Save and test action with explicit success/error state. Alternatively make the quick start say Save, then Test, and have the test result say whether the tested key is saved. Keep recovery behavior consistent with finding 05.

Acceptance: Follow only the homepage instructions from an empty profile. The user must reach an enabled Start button without guessing that an extra Save is required.

index.md:15 | README.md:11 | extension/options.js:46 | extension/options.js:55 | extension/sidepanel.js:35

12 / P2 / Meeting selection and status feedback are not accessible

Confirmed in markup/event handlers; no screen-reader audit was performed.

History entries are li elements with click listeners, without links/buttons, keyboard handling or a tab stop. Keyboard users cannot select another meeting. The key input, output-device select and theme select lack explicit labels; key/error/recording changes lack live status semantics. Replacing the meeting list and transcript every three seconds can destroy focus on playback controls and text selection.

Fix: Use native links or buttons for meeting navigation, explicit labels, named playback controls and restrained live status regions. Update changed rows in place; preserve focus, selection and scroll position. Check focus visibility and contrast in every theme.

Acceptance: Complete setup, select a past meeting, play a line, retry and export using only the keyboard. Verify status announcements and focus survival across refreshes.

extension/meetings.js:21 | extension/meetings.js:66 | extension/meetings.js:128 | extension/options.html:11

Billing, retries and response validation

13 / P2 / The lease cannot guarantee a segment is billed only once

Reproduced with real claim/settle rules, a controlled clock and delayed provider promises.

A sending lease expires after 90 seconds, with no heartbeat or per-request deadline. A second worker can reclaim a still-running request. In the reproduction the same audio caused **two provider calls**, while only one $0.10 result remained in the ledger. A browser/context crash after provider acceptance can cause the same uncertainty. Chrome also documents worker termination when a fetch response takes over 30 seconds.

Fix: Use renewable leases and unique attempt/fencing tokens, bound requests, and persist attempt-level billing/provenance. Reconcile uncertain outcomes and use provider idempotency only if supported. Describe the queue as at-least-once unless end-to-end deduplication is proven.

Acceptance: Exercise requests exceeding the lease, death after send/before settle, lost responses and manual retries. Show uncertain spend rather than asserting exactly-once billing.

extension/lib/db.js:83 | extension/lib/db.js:102 | extension/lib/openrouter.js:32 | extension/lib/queue.js:44

14 / P2 / Every HTTP 400 disables verbose timestamps for that model

Reproduced with an unrelated unsupported-language provider error.

classify correctly distinguishes response_format errors, but the queue ignores that distinction and treats any 400 while verbose as verbose_json rejection. An invalid language or another malformed parameter permanently sets noVerbose for the model and resends once. Later valid requests lose timestamps, with no reset control or explanation.

Fix: Fallback only on the classified format-capability error. Preserve unrelated validation errors, avoid poisoning the capability cache, and allow re-probing a model/provider after a suitable interval.

Acceptance: Test separate 400 responses for unsupported format, invalid language, model parameters and policy restrictions. Only the explicit format error may update noVerbose.

extension/lib/openrouter.js:9 | extension/lib/queue.js:52

15 / P2 / Model timestamps are accepted outside the audio segment

Reproduced through actual response parsing and line assembly.

The parser checks only that start/end are finite numbers. It accepts negative, inverted and far-out timestamps; line assembly treats them as model timing. A one-second audio segment accepted a line from **-9 seconds to 10,000 seconds**. Invalid times can reorder the transcript, distort echo detection and provide misleading debrief evidence. The displayed clock clamps negatives, hiding part of the defect.

Fix: Validate 0 <= start <= end within the measured WAV duration, with a documented tolerance. Reject or flag invalid timing and fall back to the segment boundary. Preserve the raw result separately for diagnosis.

Acceptance: Feed negative, reversed, excessively large and partially invalid timestamps. No invalid time may become an unqualified source reference.

extension/lib/openrouter.js:73 | extension/lib/transcript.js:22 | extension/lib/transcript.js:71

06 / AUDIO AND PROVENANCE

Timing accuracy and traceability

16 / P2 / Fallback resampling drifts; Stop drops the worklet tail

Reproduced with real resampler/worklet and synthetic 128-frame input.

resample floors the output count independently for each 2,048-sample batch and resets interpolation phase. At 48 kHz, an hour becomes **3,596.484375 seconds**: 3.515625 seconds short. This only affects the fallback when the 16 kHz context is unavailable. Separately, the worklet has no flush protocol: 1,920 queued samples (120 ms) were still unposted at Stop; the maximum tail is just under 128 ms at 16 kHz.

Fix: Use a stateful resampler carrying phase and residual samples, with appropriate filtering. Flush and acknowledge the worklet before flushing the segmenter and closing audio contexts.

Acceptance: Account for every frame over long 44.1/48 kHz streams and arbitrary stop points. Distinguish intentional mute/silence removal from unexplained missing samples.

extension/lib/wav.js:18 | extension/worklet.js:3 | extension/offscreen.js:43 | extension/offscreen.js:108

17 / P2 / The shipped build identifies an older source commit

Verified in source and the repository ZIP; all packaged files match each other.

The v0.3.1 build stamp reports **18f602d**, the v0.3.0 commit, while the reviewed release is **a4bbdca**. package.mjs stamps the pre-commit HEAD; CI deploys the committed ZIP rather than stamping the release checkout. The new release test verifies only that a commit-shaped string exists. A user following that SHA obtains different source from the installed extension.

Fix: Build/stamp the release artifact from the immutable release checkout in CI and publish an artifact-to-commit manifest. If generated artifacts remain in git, distinguish source-tree identity from packaging-commit identity and handle dirty trees explicitly.

Acceptance: Check the embedded revision against the intended release source, and make a clean rebuild reproducible with an explicit timestamp. A hex-shaped string is not enough.

extension/build.js:2 | admin/build/package.mjs:25 | tests/release.test.mjs:33 | .github/workflows/deploy-pages.yml:147

18 / P2 / Debriefs can become stale without warning or verifiable citations

Confirmed from generation, persistence and rendering paths.

Write the debrief is available while capture/queue processing continues. It reads a transcript snapshot but stores no source revision or line IDs. Later transcription/retries do not mark it stale. Requested timestamps are ordinary generated text, not validated citations or links to transcript/audio. A gap warning in the prompt helps, but does not show that the displayed debrief predates the final transcript.

Fix: Store transcript revision/hash, source line IDs and generation inputs. Label summaries partial/stale after transcript changes. Validate cited IDs, make them navigable, and separate trusted instructions from untrusted meeting content in the prompt. No prompt-injection exploit was tested.

Acceptance: Generate with pending segments, settle/retry them, and verify a stale indicator. Every cited point must navigate to an existing source line and audio interval.

extension/meetings.js:92 | extension/meetings.js:76 | extension/lib/debrief.js:13 | extension/lib/prompts.js:8

07 / WIDER ASSESSMENT

What is sound, and what should improve next

Security controls worth preserving

The MV3 manifest requests a narrow API set and one host, openrouter.ai. There are no content scripts, web-accessible resources or external-message handlers in this snapshot. Runtime networking is centralized; the API key is in an Authorization header and is omitted from normal exports. UI rendering uses textContent instead of injecting model HTML. CSP excludes remote scripts. Audio and segment metadata are persisted in one transaction before sending. These reduce real attack surface; absence of sender checks alone is not evidence of a remotely exploitable message endpoint here.

The release pipeline pins actions to SHAs, separates permissions by job and gates deployment on validation/tests. The ZIP/source equality gate is useful and passed independent comparison. The new public test-report template explicitly tells users to remove private meeting information. The repository has no runtime npm dependency chain.

Additional code and release hardening

Observation	Recommended change
Global configuration can change queued work	Segments begin with model:null and use current model/language/ZDR at dispatch. Snapshot the policy/model per meeting or segment; show changes and retries accurately.
Spend is incomplete	costOf covers successful transcription segments; it omits uncertain/duplicate attempts, and repeated debrief generation overwrites earlier debrief cost. Keep an attempt ledger and show transcription, debrief and unknown totals.
History work grows with retained data	claimNext/countByStatus scan all segments; echo assembly compares mic lines with tab lines; UI polls and rebuilds whole lists. Index due work, bound queries, render incrementally and measure multi-hour/many-meeting performance.
Idle drain listeners accumulate	sleep adds an abort listener each time but does not remove it when the timer resolves. Clean up listeners on both completion paths.
Release branch is unprotected	GitHub returned protected:false for dev and an empty repository ruleset list. Require review and checks for changes to release code/workflows; inspect any organization-level controls separately.
Local storage is not an encrypted vault	Keys are in chrome.storage.local; audio/text are in IndexedDB without application encryption. State this clearly, add an optional session-only key, and avoid claiming resistance to a compromised browser/OS.

UX recommendations beyond the defects

Keep the v0.3.1 download CTA, five-step flow, channel labels and separate-microphone warning. Make setup show saved key, device readiness and a short audio check. Keep Stop/mute visible while scrolling. Distinguish recording, stopped/upload pending, failed and complete-with-gaps. The first screenshot repeats synthetic transcript lines; use a clearly labelled realistic example for comprehension. Reduce the ten-item navigation for first-time testers, with engineering history under a developer section. Explicitly say TXT excludes the debrief, and qualify prices by selected model, mic channel and retries. Replace absolutes such as "Nothing is lost quietly" and "It does not guess" until the behavior supports them.

08 / QA PLAN

The next tests should target real boundaries

The existing tests are useful for WAV construction, segmenting synthetic audio, fixtures, queue basics and page loading. Their key limitation is that **test:fake routes tab capture through getUserMedia**. They do not exercise the actual side-panel Start gesture, activeTab grant, stream-ID redemption, real Meet playback or real OpenRouter responses. The release test loads the ZIP and checks provenance text; it does not record a call.

Test group	Required scenarios and pass condition
Real capture / device matrix	Actual Meet tab on Windows, macOS, Linux/ChromeOS as supported; headphones/speakers/Bluetooth; two devices; no grant, denied grant, unplug/replug. Start captures the intended tab and preserves playback.
Privacy and deletion	Separate Meet mute vs extension mute; pause/resume at every batch boundary; delete active meeting/all; clear key during sending. Verify persisted WAV samples and outbound dispatch, not only UI flags.
Failure recovery	Offline -> Stop -> new meeting -> reconnect; 401/402 -> valid key during capture; 429 Retry-After, 5xx, slow/hung responses, missing offscreen context, quota failures. State and data stay truthful.
Crash / worker lifecycle	Terminate worker/recorder after upload but before settlement; restart browser; reload extension; close tab during starting; sleep/wake. Recover durable work without unexplained gaps or false completion.
Transcript / debrief integrity	Empty responses; invalid times; reply falsely flagged as echo; switch/delete meeting during chat; generate before queue settles; spoken prompt-injection attempts. Preserve source identity and show uncertainty.
Speech quality / model evaluation	Consented multilingual recordings with human references; short yes/no answers, names/numbers, accents, quiet voices, overlap, background noise. Measure WER, missed speech, timing, hallucinations and action correctness.
Long session / scale	Two-hour calls plus many retained meetings; measure memory, CPU, dropped frames, queue latency, UI responsiveness and final sample count. Repeat with 44.1/48 kHz fallback paths.
Accessibility / installation	Fresh user follows only homepage; update existing unpacked install; keyboard-only and screen-reader flows; narrow panel/200% zoom; all five themes and status changes.

Release acceptance

Close all P1 findings with regression coverage, then complete a real-call smoke test using a capped key and non-sensitive test speech. Freeze the tested SHA/artifact checksum. Keep test evidence for exact Chrome versions and audio devices. Record sanitized real provider fixtures. Obtain separate human transcripts for quality evaluation; the current synthetic waveforms and canned replies cannot establish transcription accuracy. Do not use a numeric quality score until those measurements exist.

09 / DELIVERY PLAN

A practical order for the next patches

#	Work package	Exit condition
1	Data ownership and deletion	Immutable meeting/request identity; deletion fencing; source IDs on debriefs; no orphan capture writes or wrong-meeting results.
2	Recorder / queue lifecycle	Independent recording and upload states; bounded requests; live key recovery; startup rollback; idempotent stop; truthful completion.
3	Audio integrity and privacy	Worklet gating/flush; stateful resampling; critical storage-failure handling; visible empty-result gaps; immediate retention behavior.
4	Provider and transcript correctness	Constrained fallback; timestamp validation; uncertain echo handling; attempt-level cost/provenance; stale summary indicators.
5	Onboarding and accessibility	Save-and-test key flow; explicit labels; keyboard meeting navigation; stable focus; clear status and privacy wording.
6	Release evidence	Stamp the immutable source revision; keep ZIP/source equality; enforce branch checks; real-call and hardware/lifecycle validation before broader distribution.

Architecture recommendation

A framework rewrite is unnecessary. Keep the small modules, but give the service worker one serialized session controller. Commands should carry a meeting/session generation ID; effects should report success/failure before state changes. Keep capture state independent from background transcription jobs. Persist an event or status record for every meaningful gap, retry, deletion and revision. Let the UI subscribe to those states instead of reconstructing all truth by polling.

Security hardening that should remain proportionate

Add message schemas and sender/context checks as defense in depth, especially before any future content script or externally_connectable support. Do not describe the current internal message handler as an unauthenticated website API. Keep the one-host policy and safe text renderer. For Markdown exports, define whether transcript/model text is literal evidence or renderable Markdown; escaping untrusted HTML/images avoids surprises in downstream viewers.

What the review deliberately does not conclude

It does not certify legal consent compliance, prove provider deletion, validate a specific OpenRouter account's routing configuration, establish real-call word accuracy, or establish accessibility conformance. No exploit against a live external service was attempted. No production data was used. Missing capabilities such as speaker diarization, transcript editing, search and Web Store distribution are roadmap scope, not undisclosed defects, because the product names them as absent.

Review snapshot and artifact

Source: **a4bbdca62a46f9ea67b7c5d040b3d8bf31b40ad8**
Version: **0.3.1**
Repository ZIP: **44,526 bytes; 27 entries**
SHA-256: 2eca0eb202b4c916f1a6432162cac4a49f6ecc9882a9713c061ed626af8cf1e2
The repository ZIPs match the reviewed tree. Deployed ZIP bytes were not independently downloaded.

Reproduction results and primary sources

Focused checks ran against the reviewed source. Pure functions were imported directly. UI/coordinator/worklet checks used fake DOM/Chrome/media objects; queue checks used a controlled clock, provider promises and an in-memory transaction adapter. They establish application logic and are not a substitute for browser integration tests.

Check	Observed result
Empty provider result	0 lines; 0 timeline entries for a done audio segment.
Echo false positive	A genuine "I will send the report today" reply was omitted from debrief input.
Timestamp bounds	Accepted -9,000 ms start and 10,000,000 ms end on a 1-second segment.
48 kHz resampling	3,600 seconds became 3,596.484375 seconds.
Worklet stop tail	1,920 samples / 120 ms remained unposted.
Mute boundary	1,920 muted samples were accepted after unmute.
Storage rejection	Write promise resolved after QuotaExceededError; error only in S.errors.
Debrief identity	Transcript A produced a debrief stored on meeting B.
Startup failure	Second start said already starting; Stop left draining.
Key-recovery kick	0 worker drains while an offscreen document existed.
Unrelated HTTP 400	Unsupported-language error set noVerbose.model = true.
Delete while producing	0 meetings; 1 orphan segment/WAV, claimable for send.
Expired lease	2 provider calls; only 1 successful cost stored.
Offline final drain	Still unresolved with pending audio after repeated retry cycles.
Held drain recovery	Key fixed and segment requeued; completed drain did not restart.

Primary references, checked 3 November 2026

1. GitHub CI run for a4bbdca: 45 passing tests; source links appear beside every finding.
2. Chrome extension service-worker lifecycle - termination, persistence and recovery boundaries.
3. Chrome alarms reference - persistence and startup reconciliation, including pre-150 behavior.
4. Chrome tabCapture reference - stream-ID restrictions and capture requirements.
5. OpenRouter transcription endpoint tutorial - payload, timestamp formats and data-policy controls.
6. OpenRouter Zero Data Retention documentation - account/guardrail Safety Retention choice and policy exceptions.