Voice-Driven Real-Time Podcast Editing, Assembly, and In-Band Command Macros

Provisional Patent Application - One-Page Summary

Inventor	Benjamin Scott Gerhardt
Residence	Glendale, CA
Working Names	Podcast-Pro-Plus (PPP), NoSweatCast
Prepared	September 21, 2025
Drawings	Yes (FIG. 1-9)

Cover-Sheet Quick Answers

Title	Voice-Driven Real-Time Podcast Editing and Assembly System
Inventor	Benjamin Scott Gerhardt
Residence	Glendale, CA
Applicant	Individual (same as inventor)
Correspondence	121 W Lexington Dr. Suite 236 — Glendale, CA 91203
Entity	Micro (if eligible) or Small
Government Interest	No
Drawings	Yes

Overview

Systems and methods for in-band, spoken command macros during or immediately after recording that drive non-destructive, transcript-aligned edits (cut/mute, fades, SFX and music level changes) and ad-break insertion, with a time-bounded rollback ('Flubber') and cloud-orchestrated assembly/publishing. UI emphasizes a simplified, senior-friendly workflow.

Provisional Patent Cover Sheet

Title: Voice-Driven Real-Time Podcast Editing and Assembly System
Inventor: Benjamin Scott Gerhardt
Residence: Glendale, CA
Applicant: Individual (same as inventor)
Correspondence Address: 121 W Lexington Dr. Suite 236
Glendale, CA 91203
Entity: Micro (if eligible) or Small
Government interest: No
Drawings: Yes

Provisional Patent Application Draft - v2

Voice-Driven Real-Time Podcast Editing, Assembly, and In-Band Command Macros

Inventor: Benjamin Scott Gerhardt (Glendale, CA) Working names: Podcast-Pro-Plus (PPP), NoSweatCast Prepared: September 21, 2025

This is a provisional-ready technical specification describing systems and methods for in-band, spoken command macros that drive non-destructive, transcript-aligned audio edits (e.g., cut/rollback, SFX and music level changes, commercial break insertion) with cloud orchestration (Celery or Google Cloud Run Jobs/Tasks) and a senior-friendly UI.

1. Field of the Invention

Computer-implemented audio production and publishing. More specifically, detecting spoken, in-band edit and control commands during capture or shortly thereafter; aligning those commands to token-timed transcripts; executing non-destructive timeline operations (cuts, fades, SFX inserts, music ducking, ad-break insertion) with rollback/abort semantics; and assembling/publishing the resulting episode in a cloud pipeline.

2. Background

Traditional podcast editing requires manual timeline work after recording. Notes spoken during a session are rarely actionable. Prior systems support text-based editing from transcripts and some filler-word cleanup; however, they generally lack a formal, in-band command grammar that is spoken naturally during performance, automatically recognized, executed, and scrubbed from program audio, supports a time-bounded rollback/abort, and extends to mix automation and ad-break macros tuned for podcast workflows.

3. Summary of the Invention

The invention provides:

A spoken command language embedded in normal speech (wake words and phrases such as "Intern, cut that," "Flubber," "Music down three dB," "Insert sting 'Intro A'," "Insert commercial break").

Normalization and matching on tokenized transcripts (word-level timestamps), with resilient alignment to audio timecodes.

Non-destructive edit application over a mutable representation (word list + edit log), including trims, fades, SFX/music automation, and ad-marker creation with slot-placement rules.

A rollback/abort mechanism (e.g., "Flubber" within a time window) that atomically reverts the last macro's multi-track changes and transcript markers.

A cloud pipeline that executes these macros idempotently using Celery or Google Cloud Run Jobs/Tasks (with Pub/Sub fan-out and Cloud Storage artifacts).

A senior-friendly UI with a simplified recording flow, progress status ("While You Work"), and one-click assembly/publishing.

4. Brief Description of the Drawings (proposed)

FIG. 1 - System block diagram: UI, storage, ASR, Command Detector, Edit Engine, Mix/SFX Engine, Ad-Break Engine, Assembler, Publisher.

FIG. 2 - Session dataflow: Record -> Upload -> Transcribe -> Detect Commands -> Apply Macros -> Render -> Publish.

FIG. 3 - Token model with start/end timestamps and normalization pipeline.

FIG. 4 - "Intern" macro flow: token match -> region compute -> non-destructive CUT/MUTE ->

render.

FIG. 5 - "Flubber" safety flow: time-bounded ABORT/rollback of prior macro(s).

FIG. 6 - SFX/Music mix timing: token-aligned inserts, auto-duck, crossfades, level-match.

FIG. 7 - Commercial Break macro: spoken trigger -> ad marker(s) -> policy (min spacing, max count, loudness) -> placeholder insert (static now; dynamic later).

FIG. 8 - Pipeline variants: (a) Celery; (b) Cloud Run Jobs/Tasks + Pub/Sub + GCS; deterministic artifacts.

FIG. 9 - UI states: Quick Tools, While-You-Work, Assemble, Publish.

5. Detailed Description (exemplary) 5.1 Core Data Structures

Transcript token: { idx: int, word: str, start: float, end: float, speaker?: str } (seconds).

Mutable word list: token sequence supporting virtual removals/replacements.

Edit operation (examples): CUT(time_range), MUTE(time_range), INSERT_SFX(id, at_time), MUSIC_DELTA_DB(delta_db, duration), FADE(shape, duration), INSERT_AD_MARKER(slot_id, at_time), INSERT_MEDIA(clip_id, at_time).

Edit log: append-only, with op type, params, token/time context, and macro provenance.

5.2 Command Normalization & Matching

Lowercasing, punctuation stripping, regex tokenization; synonym mapping for common ASR variants; sliding-window fuzzy matching; matched token indices map to absolute timecodes via token.start/token.end. Confidence gating avoids false positives.

5.3 Intern Macros (structured edits)

Examples: "Intern, cut that," "remove last five seconds," "mark chapter Intro," "add shownotes: ...". Execution translates to edit ops over time ranges computed from token boundaries and/or relative windows; optionally preserve or scrub the spoken command region. Warnings when confidence is low.

5.4 Flubber (safety/rollback)

Abort window: if "Flubber" is spoken within W seconds (e.g., 15s) of a prior macro, trigger ABORT and rollback transcript markers and media changes atomically. Rollback-restart: a single "Flubber" can trim a configurable look-back region to re-speak a line; cleanup heuristics handle restarts/fillers.

5.5 SFX / Music Mix Macros

Spoken phrases: "music up/down [N] dB (for [duration])," "insert sting [name]," "fade under." Level-match to a target dBFS/LUFS; crossfades for artifact-free transitions; speech-first priority if overlapping narration exists.

5.6 Commercial Break Macro (in-band ad-break insertion)

Spoken trigger examples: "insert commercial," "insert break," a branded wake phrase. Operation:

Token-align trigger and compute a canonical insert point (nearest inter-sentence boundary).

Create an Ad Marker with {slot_id, at_time, max_pod_length, min_gap, pod_type}.

Apply ad-pod rules: respect min spacing, max per episode/segment, and loudness/peak constraints.

Insert placeholder media (today: static host-read or sponsor clip). Future: dynamic ad insertion at render/publish time.

Scrub the spoken trigger and heal room tone. Safety: "Flubber" inside the window reverts the ad marker and removes the placeholder insert. UI: ad markers visible on the timeline with drag-adjust; slots export to chapter/ad JSON.

5.7 Cloud Pipeline & Orchestration

Variant A (Celery): tasks for transcribe -> detect -> apply -> render -> publish; artifact keys ensure idempotency. Variant B (Cloud Run Jobs/Tasks): stateless jobs per stage; Pub/Sub fan-out; GCS artifacts; deterministic ordering via edit-log sequence numbers.

5.8 Senior-Friendly UI/UX

Single-screen Quick Tools recorder; big transport controls; mic test; autosave; While-You-Work status with manual notes; one-click Assemble/Publish with sensible defaults; brand A/B toggle.

5.9 Best Mode (illustrative, non-limiting)

ASR with word-level timestamps; 48 kHz internal audio; export MP3 192-256 kbps at ~-16 LUFS; abort window W ~ 15 s; token normalization via regex + synonym maps; default ad-pod policy (min gap 6 min; max 3 breaks per 60 min; LUFS match on inserts).

6. Alternative Embodiments

Live vs near-real-time detection; on-device detection; video podcasts with synchronized cuts and lower-thirds; locale-specific grammars; speaker-aware commands.

7. Advantages

Captures editor intent at performance time; atomic rollback via a simple safety word; podcast-native ad-break macros with spacing/loudness governance.

8. Example Scenarios

(1) "Intern, cut that... Flubber" -> cut executed then rolled back. (2) "Music down three dB for four seconds... insert sting 'news-bump'." -> mix automation + SFX aligned to tokens. (3) "Insert commercial break." -> marker placed at sentence boundary; placeholder ad inserted; trigger scrubbed.

9. Claims (informal, for provisional)

A method comprising: receiving an audio signal; generating a token-timed transcript; detecting a spoken command embedded in the audio; mapping the command to a token-aligned time range; and non-destructively applying an edit operation to produce an edited recording.

The method of claim 1 wherein the spoken command is a macro selected from: cut, mute, fade, insert ad break, insert sound effect, and change background-music level.

The method of claim 1 wherein detecting comprises token normalization and fuzzy phrase matching over sliding windows.

The method of claim 2 wherein insert ad break creates an ad marker subject to pod rules (minimum spacing, maximum count, loudness constraints) and inserts placeholder media to be confirmed or replaced at render time.

The method of claim 1 further comprising scrubbing the spoken command from program audio and healing room tone.

The method of claim 1 further comprising a safety command that, when detected within a time window of a prior edit, atomically rolls back the prior edit's media and transcript changes.

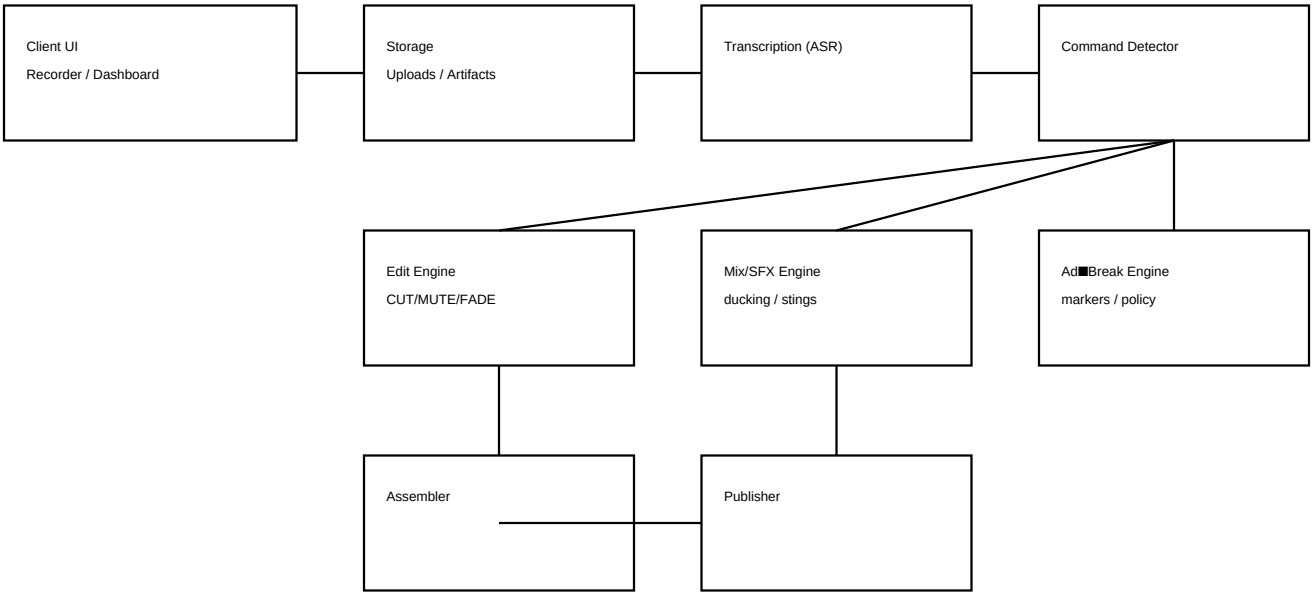
The method of claim 1 wherein overlapping macros are resolved by priority rules favoring speech intelligibility.

The method of claim 1 executed in a cloud pipeline selected from Celery workers or serverless job orchestration using Google Cloud Run Jobs/Tasks and cloud storage artifacts.

A system comprising: an interface, a transcription engine, a command detector, an edit engine, a mix/SFX engine, an ad-break engine, and an assembler, configured to perform any of claims 1-8.

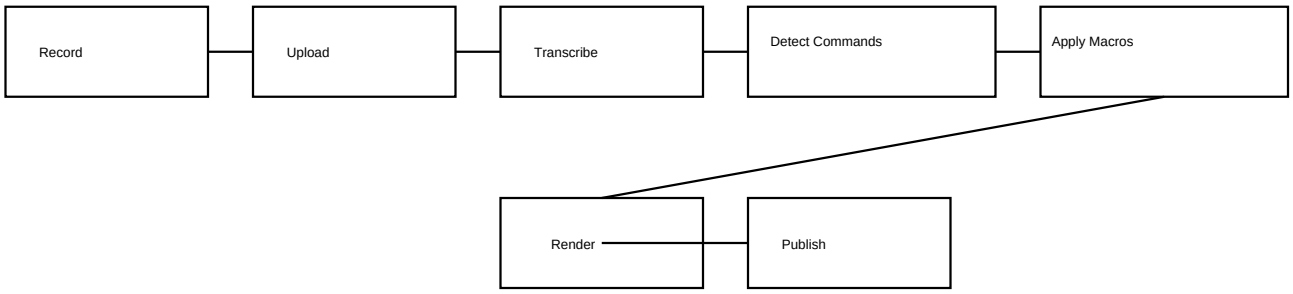
A non-transitory computer-readable medium storing instructions to perform any of claims 1-8.

FIG. 1 — System Block Diagram



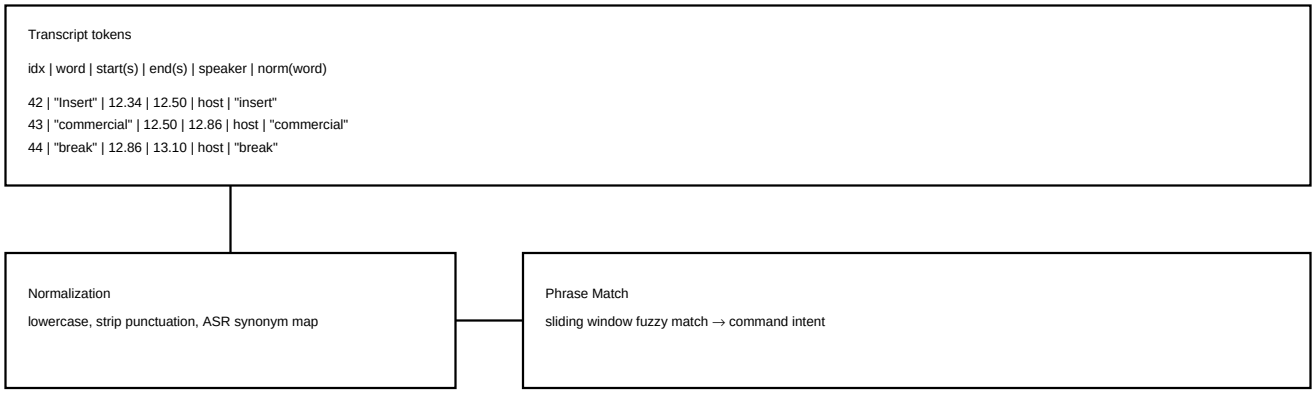
Generated 2025-09-19 22:41

FIG. 2 — Session Dataflow



Generated 2025-09-19 22:41

FIG. 3 — Token Model Normalization



Generated 2025-09-19 22:41

FIG. 4 — Intern Macro (Cut/Mark)

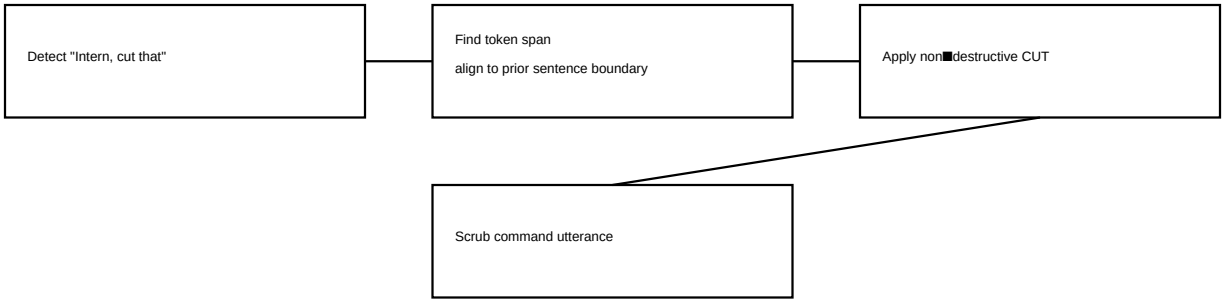


FIG. 5 — Flubber Safety Rollback

FIG. 6 — SFX/Music Mix Macros

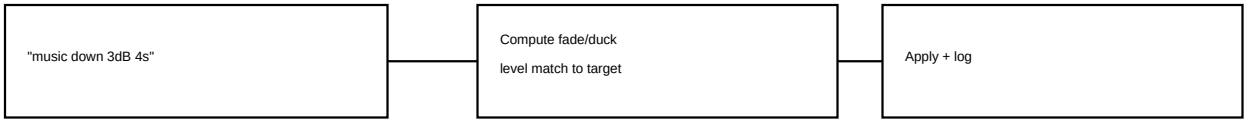


FIG. 7 — In■Band Commercial Break Macro

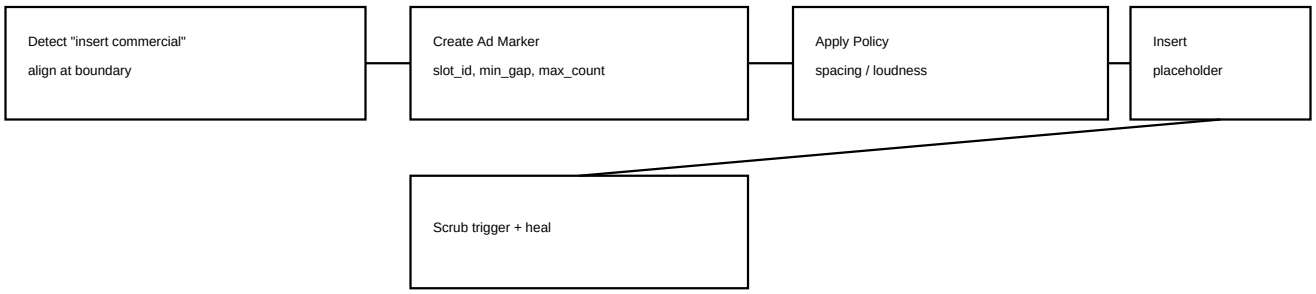


FIG. 8 — Pipeline Variants

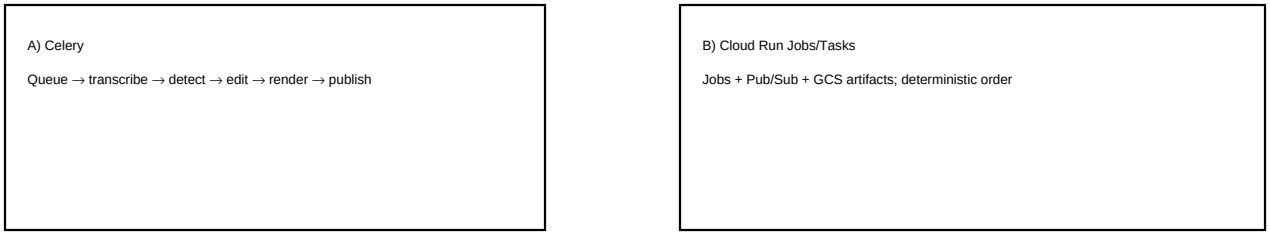


FIG. 9 — UI States