# My Weird Prompts

## A Voice-to-Podcast Pipeline

Technical Architecture White Paper

Daniel Rosehill

October 2026 — Version 4.0

Pipeline V5.9 · Prompts 4.21 · DeepSeek V4.1 Flash · Chatterbox Regular on 3× NVIDIA L4

# Abstract

My Weird Prompts is a fully automated podcast. I send in a prompt — a voice memo, a few typed lines, a link, sometimes a photo — and about a quarter of an hour later a finished episode is live: a researched, scripted conversation between AI characters, voiced with cloned voices, mastered, given cover art and show notes, and published to the website and every podcast platform. Nobody edits anything in between.

The pipeline runs on Modal (serverless CPU and GPU), with Cloudflare R2 for files and the site's content catalogue, Neon PostgreSQL for metadata, Pinecone for the hosts' memory, and Vercel for the website. Every language-model stage runs on **DeepSeek V4.1 Flash**; every voice is **Chatterbox Regular**, running on the show's own GPUs.

This edition describes production as of **17 October 2026** (pipeline V5.9, prompts 4.21). It replaces the February 2026 edition (v1.0), which described pipeline V4, a two-host format, a two-pass LLM editing system and Gemini-written scripts — almost none of which survives. The figures in it are measured from the public episode record, not estimated.

> **Project links**
> Web edition (always current): myweirdprompts.com/technical
> Pipeline changelog: myweirdprompts.com/about/changelog
> Open-source reference code: github.com/My-Weird-Prompts/pipeline
> Episode dataset: huggingface.co/datasets/My-Weird-Prompts/episodes

# Contents


Abstract . . . . . 2
1 By the Numbers . . . . . 4
2 System Overview . . . . . 4
3 Intake . . . . . 5
3.1 Entry points . . . . . 5
3.2 Duplicate protection . . . . . 5
3.3 The cancel grace window . . . . . 6
4 Transcription and Routing . . . . . 6
4.1 Transcription . . . . . 6
4.2 Model routing . . . . . 6
5 Research and Planning . . . . . 7
5.1 The research agent . . . . . 7
5.2 World state . . . . . 7
5.3 Planning . . . . . 7
6 The Script . . . . . 8
6.1 One pass, the prompt verbatim . . . . . 8
6.2 The cast . . . . . 8
6.3 The rules that matter . . . . . 8
6.4 Length gates . . . . . 8
6.5 Deterministic polish . . . . . 8
6.6 Sign-off recommendations . . . . . 9
7 Speech . . . . . 9
7.1 Text normalisation . . . . . 9
7.2 Chatterbox on three L4s . . . . . 9
7.3 Making the decoder fast . . . . . 9
8 Assembly and Mastering . . . . . 10
9 Publication . . . . . 10
9.1 Publishing without a build . . . . . 11
10 Reliability . . . . . 11
10.1 The crash supervisor . . . . . 11
10.2 Credit alerts . . . . . 11
10.3 Failing open . . . . . 11
10.4 Observability . . . . . 11
11 Technology Stack . . . . . 12
12 Lessons Learned . . . . . 12
Appendix: Changes Since the Previous Edition . . . . . 13

# 1 By the Numbers

Every episode is stamped with the model that wrote its script, the pipeline and prompt versions that made it, its GPU type and worker count, its generation and speech times, and its LLM and GPU cost. That record is published daily as the episodes dataset on Hugging Face. The figures below are computed from it: 164 standard episodes (#5880–#6050) published 12–17 October 2026 on pipelines V5.4–V5.6, all scripted by DeepSeek V4.1 Flash and voiced on three L4 workers.

| Measure | 10th pct | Median | 90th pct |
|---|---|---|---|
| Run time (start to published) | 10.6 min | 13.0 min | 17.4 min |
| Speech synthesis time | 3.5 min | 4.5 min | 6.8 min |
| Episode length | 21.4 min | 26.8 min | 33.7 min |
| Script length | 3,499 words | 4,436 words | 5,700 words |
| Dialogue segments | 122 | 159 | 231 |
| LLM cost | $0.048 | $0.067 | $0.090 |
| Speech GPU cost | $0.138 | $0.181 | $0.270 |
| LLM + GPU + CPU | $0.205 | $0.258 | $0.351 |

Speech is about 70% of the cost of an episode and the language model about 26%. The previous measured figure, from 339 episodes in late September and early October, was about $0.43 per episode; the drop is almost entirely the speech-decoding work described in §7.

# 2 System Overview

An episode is one job. It enters through a webhook, runs as a chain of Modal functions, and ends as an audio file, a database row and a set of catalogue entries the website reads.

```
Send App / MCP server
       │  webhook (/webhook/generate*)
       ▼
Intake ── exact + near-duplicate guard ── 409 if duplicate
       │
60 s cancel grace window
       │
LangGraph: prompt_enhancement → grounding → script_writer
           → review (deterministic polish) → extract_memories
       │
Length gates → parse → metadata → cover art
       │
Text normalisation → parallel TTS (3 × L4) → assembly & mastering
       │
Publish: R2 + Neon + Pinecone + content catalogue → feeds & pings
```

| Component | Role |
|---|---|

| | |
|---|---|
| Modal | CPU orchestration, GPU speech workers, scheduled jobs. A merge to `main` that touches pipeline code redeploys it through GitHub Actions. |
| DeepSeek API | Every LLM stage, on DeepSeek V4.1 Flash. OpenRouter is a standby route. |
| OpenRouter | Transcription (Gemini 3.1 Flash Lite, Whisper large-v3 fallback) and embeddings. |
| Cloudflare R2 | Audio, transcripts, images, voice conditionals, and the content catalogue the website reads. |
| Neon PostgreSQL | Jobs, episodes, characters, episode types, production records. |
| Pinecone | Past-episode retrieval and the hosts' long-term memory. |
| Vercel | The Astro website and the Send App. |

The canonical voice-prompt path is its own stage; fourteen other episode types (panels, debates, interviews, reader questions and so on) run through a single generic dispatcher that reads each type's routing from a database row and calls registered handlers.

# 3 Intake

## 3.1 Entry points

Episodes are dispatched from two places: the **Send App** (a small web app behind sign-in, used to record or type a prompt and attach files or images) and an **MCP server** that lets an AI agent dispatch from a coding session. Both call the same webhooks. Listeners can send prompts through a form on the website or by email; those reach me and do not dispatch anything on their own.

The Send App offers three lengths: **short** (15 minutes), **standard** (25) and **extended** (30). Text, PDF and CSV attachments are extracted to text; links are fetched in full as primary sources. Up to five images can accompany a prompt — downscaled to 1,568 px, HEIC converted, anything unreadable skipped — and only the script writer sees them.

## 3.2 Duplicate protection

An **exact guard** normalises the prompt (Unicode, quotes, whitespace, case), fingerprints it, and refuses it with HTTP 409 before anything is spawned if the same prompt is already queued, running or recently finished.

The **near-duplicate guard** (16 October 2026) catches the same question reworded. It exists because episodes #6026 and #6027 came from one question rewritten and resent four and a half minutes apart.

The obvious design — block anything whose embedding is too similar — failed on the numbers. Across 766 jobs from 30 days, the rewrite scored a cosine similarity of 0.964 with its original; parts one and two of a listener's series, which are different episodes that should both exist, scored 0.952. The median for two unrelated prompts was 0.637. A threshold cannot separate 0.964 from 0.952. So similarity is only a shortlist:

- each new prompt is embedded (Gemini Embedding 2, 768 dimensions) and compared with up to 40 prompts from the last six hours;
- at most two candidates scoring 0.86 or higher go to DeepSeek V4.1 Flash, which reads both prompts and returns a verdict;
- only a verdict of `duplicate` blocks, and the response offers a “send anyway” override;
- any error or timeout lets the prompt through.

The first version of the verdict prompt blocked a listener’s explicit follow-up question. After that rule was fixed it labelled all 14 test pairs correctly on two consecutive runs.

### 3.3 The cancel grace window

Every job waits 60 seconds before it spends anything, checking every five seconds whether it has been cancelled. Before this, a dispatch was effectively irreversible: every place a job could be cancelled came after research and the script had already been paid for. The window costs one idle CPU minute per episode.

## 4 Transcription and Routing

### 4.1 Transcription

Voice memos are transcribed by **Gemini 3.1 Flash Lite** via OpenRouter in a single pass that also strips fillers (um, uh, like, you know, sort of). If that fails, the fallback is **Whisper large-v3**, also via OpenRouter. Both paths then run a deterministic correction pass for words the models habitually mishear.

Whisper used to run on OpenAI’s own API. In October 2026 that account ran out of credit and took down every episode that started from audio. On OpenRouter, Whisper large-v3 punctuated best of the three models tried on my recordings and was the cheapest. The first deploy of the switch broke anyway: the code asked for plain text, and OpenRouter’s speech endpoint only accepts JSON.

### 4.2 Model routing

Every LLM stage runs on `deepseek-flash`, which is **DeepSeek V4.1 Flash**, called on DeepSeek’s own API. Model choice is hard-coded in the routing file; changing it is a code change and a redeploy, not an environment variable.

In September 2026 DeepSeek announced that requests for its Pro model were being served by V4.1 Flash, at the Flash price, until the next Pro model ships. The show credits its model on air, so asking for Pro made the credit untrue. The pipeline now asks for Flash by name.

The script stage sends no temperature, so DeepSeek’s default of 1.0 applies, as DeepSeek recommends for this model; thinking mode is off. Utility stages run cooler: intake 0.2, research 0.3, the crash supervisor 0. The script call requests up to 10,000 output tokens for episodes up to 40 minutes.

Running DeepSeek or a small open model on our own GPUs was costed and rejected: V4.1 Flash needs eight H200s to fit, and the break-even point is about 4,500 episodes a month against roughly 640 actually made.

# 5 Research and Planning

## 5.1 The research agent

Grounding is a tool-using research agent on DeepSeek V4.1 Flash: up to five rounds, a five-minute timeout. Its tools are grouped into **toolkits**:

| Toolkit | Tools |
|---|---|
| Core (always on) | Web search (Exa, hour-level freshness), breaking news, open news (GDELT), URL fetch, a full-text document reader (arXiv HTML/PDF, any PDF, web pages), Wikipedia |
| Technology | arXiv, Hugging Face, GitHub search, whole-repository reader, technical discussion search |
| Medical | PubMed |
| Science | arXiv |
| Judaism | Sefaria |

Toolkits are preloaded when keywords in the research brief call for them, and the agent can load another mid-run. Each tool has a per-run cap (six calls each for GitHub, Hugging Face, arXiv and Sefaria; five full-document reads; three Wikipedia lookups; four Exa searches and six page fetches); a call over the cap is answered "budget spent" and logged. For technical topics the agent is steered to the primary source — the docs, repository, specification or paper — before forums.

If the agent fails, a simpler search loop runs, then a single deterministic search. Research under 2,000 characters counts as a failure, and a thin first answer is sent back once for more.

## 5.2 World state

The hosts are told the date and today's Jerusalem weather (conditions, high, low, chance of rain), from the keyless Open-Meteo service, cached for three hours. They mention it only when it is relevant. This replaced a block of six news headlines that the hosts kept reaching for whether or not they had anything to do with the episode.

## 5.3 Planning

A planning call picks the episode's target length and word range. The standard tiers are:

| Length | Target words | Hard floor before speech |
|---|---|---|
| 15 min | 2,000–2,600 | 1,800 |
| 25 min | 3,700–4,300 | 3,000 |
| 30 min | 4,600–5,300 | 3,800 |

When the length is not requested, the planner's choice is clamped to 20–30 minutes. Grounding also pulls similar past episodes from Pinecone, the hosts' memories, and lore entries above a similarity threshold.

# 6 The Script

## 6.1 One pass, the prompt verbatim

The script is written in one call. The writer sees my prompt exactly as I sent it — earlier versions handed it a tidied summary, and the episodes drifted from what I had asked — plus any images, the research brief, the plan, character cards for Corn and Herman and their relationship, notes for Hilbert, lore and memories.

## 6.2 The cast

**Corn** (a sloth) and **Herman Poppleberry** (a donkey) host. **Hilbert Flumingtop**, the producer, gets a turn about 80% of the way through every episode: around 650 words at standard length, a story that leaves the topic behind rather than mirroring it, mostly set in the present, where ridiculous outranks useful. **Larry** interrupts each standard episode in its first half with an unbooked advertisement for a product that should not exist; the shape of the ad is drawn from seven and the hosts' reaction from nine. Other guest bits are parked.

## 6.3 The rules that matter

The script prompt went through seventeen versions in ten days in October 2026. The rules with the most effect:

- **Funny by default.** "Daniel wants every episode to be funny wherever the material allows." Grief, illness and other people's hardship are exempt; when the register is unclear, lean ridiculous.
- **Corn's quips get room.** One flat sentence, often under ten words, is one of the main reasons people listen; a short Corn turn is a reaction, not a recap.
- **Research is material, not the agenda.** The hosts think first; findings support. The weight of evidence follows the topic, my own experience counts as testimony, and when nothing has been studied they say what they think.
- **At most one callback** to a past episode, and only to one research actually supplied — checked in code afterwards.
- **No production talk.** The hosts never mention the research pass, and the script never gives the prompt address: a recorded clip does that (§8).

## 6.4 Length gates

A script under 90% of the planned minimum gets one expansion pass; one over 105% of the maximum gets one trim. Either is rejected if it overshoots (over 1.8× or under 0.6× the original). A script still below the hard floor for its length (table above), or with fewer than ten dialogue segments, never reaches the GPUs.

## 6.5 Deterministic polish

The pipeline once had an LLM review pass meant to fact-check and tighten every script. An audit found it was off in production, so none of the instructions written into it had ever run. It stays off. Its predecessor, an external fact-checker, once returned a 169-word "corrected" script and wiped out an episode. The rule since then: anything that must hold goes in the script prompt or in code.

The code is a deterministic polish pass. It removes filler praise and sycophantic openers, echoed hand-offs, references to the research pass and AI-recursion riffs; caps over-frequent phrases, conjunction openers and negation fragments; rotates overused phrases through synonyms seeded by the script; and strips host cues that summon Hilbert before his first line.

Since 15 October it also cuts **repeat-then-extend echoes** — a host repeating the end of the previous turn before adding to it. Measured on the previous 60 episodes, 138 such echoes had survived the old pass; the new one removes 124.

### 6.6 Sign-off recommendations

The hosts close by recommending up to three earlier episodes with a similarity of at least 0.6. Episode numbers and titles are re-read from the database rather than trusted from the model. "Hub" episodes that turn up in more than a quarter of recent retrievals are dropped — one early episode had been recommended in 12 of 19 runs. The picks are shown on the episode page as Recommended Listening.

After the script, a final node extracts new memories for the hosts into Pinecone, where a new memory supersedes an old one above 0.75 similarity.

## 7 Speech

### 7.1 Text normalisation

Before synthesis, numbers and symbols are rewritten in spoken form: technical tokens by rules, everything else by NVIDIA NeMo's text normaliser, touching only sentences that contain digits, symbols or domains. Across 60 recent scripts (9,897 turns) it changed 625 turns, left no unspeakable tokens, and cost about 0.3 seconds an episode on a CPU. Small language models were tried for the same job and rejected: a 1.5-billion-parameter model left digits in 55 of 80 hard sentences and dropped words in 17; the 0.5-billion one turned "A10G is a dollar" into "A ten thousand dollars is A one hundred dollars".

### 7.2 Chatterbox on three L4s

Speech is **Chatterbox Regular** (pinned version, weights baked into the image, exact build recorded on each episode). Text is cut into chunks of up to 250 characters at sentence ends and divided into three contiguous blocks, one per **NVIDIA L4** worker. Each worker loads precomputed voice conditionals for every character, retries a failed segment once, skips segments it has already made (so a restarted job resumes), and encodes audio on a CPU thread pool so the GPU never waits for it. If more than 20% of segments fail, the episode fails. Containers shut down two seconds after their batch.

### 7.3 Making the decoder fast

Two changes in October 2026 cut speech cost by roughly two thirds:

1. A one-line patch turns off a flag that requested attention weights on every decoding step. Only the multilingual model reads them, and requesting them forced a slower attention path. The output is token-identical and about 10% faster.
2. The one-token decoding step is captured as a **CUDA graph** with a static key–value cache, and the backbone runs in fp16.

| GPU (list price) | Stock RTF | CUDA graph fp16 RTF | Cost per audio-hour |
|---|---|---|---|
| L4 (~$0.80/h) | 0.957 | 0.346 | $0.77 → $0.28 |
| T4 (~$0.59/h) | 0.84 | 0.52 | $0.50 → $0.31 |
| A10G (~$1.10/h) | 0.61 | 0.31 | — |

RTF is seconds of compute per second of audio, measured on 16 real lines. In fp16, tokens drift late in three or four of 16 turns, so it shipped as a listening test. In production the measured GPU cost is a median of $0.39 per audio-hour, because the recorded cost bills all three workers for the whole parallel stage.

An experimental route renders Hebrew code-switching: a small classifier finds the Hebrew spans, one LLM call writes them in Hebrew script with vowel points, and Chatterbox Multilingual renders and stitches the parts. It is opt-in.

# 8 Assembly and Mastering

The episode is assembled in this order:

1. intro music;
2. the AI disclaimer, read by a clone of my own voice;
3. my original voice prompt, EQ'd and trimmed (audio prompts only);
4. the dialogue, including Larry's ad and a pre-recorded spoof mid-roll spliced in near the halfway point;
5. Hilbert's "send a prompt" clip — the address and the form, recorded once, so the model never has to say a URL;
6. the credits, naming the script model and the speech engine;
7. outro music.

The dialogue bus gets a room-tone bed, per-character tonal adjustment, EQ, compression and de-essing, and since 17 October a pitch-preserving slowdown to 0.97×. The whole episode is loudness-normalised once to −16 LUFS (true peak −2 dB) and encoded as 128 kbps mono AAC at 48 kHz, with chapters and waveform peaks.

# 9 Publication

Publication uploads the audio, transcript and images to R2 in parallel, writes the episode row to Neon, and upserts its embedding to Pinecone. Since 15 October a failed database insert fails the job instead of reporting success over an episode no one can find; the files go to recovery storage.

## 9.1 Publishing without a build

The website used to show a new episode only after a rebuild. Since 14 October the pipeline instead asks the site to rewrite its **content catalogue**: the site re-reads the database and writes a manifest, an episode list and the episode's own file to a private R2 bucket, then purges its cache. The episode page, homepage, episode list, RSS feed, per-episode JSON and the radio feed read from that catalogue, so they update within a minute or two. Other pages still come from a half-hourly build.

After that: backups to two further object stores, a Telegram announcement once the page is live, and feed pings (WebSub, Podcast Index). The prompt text itself is never published anywhere; the public dataset records only its length, the sender's role and the intake channel.

# 10 Reliability

## 10.1 The crash supervisor

Until October every crash was retried a fixed number of times with a fixed wait — sensible until the cause is something that won't fix itself, like a bad input or an empty credit balance, and two more full runs are burned before the job gives up. Now a small supervisor, DeepSeek V4.1 Flash at temperature zero with 300 output tokens, reads the error and answers **retry** (with a delay clamped to 15 seconds–5 minutes) or **abort**. The retry cap (two) is enforced in code, not by the model, and if the supervisor itself fails the old fixed rule applies.

## 10.2 Credit alerts

On 14 August 2026 the DeepSeek balance hit zero and every episode stopped. Twice a day a housekeeping job now checks the DeepSeek and OpenRouter balances and emails if either is under $5, or if DeepSeek reports the account unavailable — a flag that goes false before the balance does. The alert repeats every run until someone tops up.

## 10.3 Failing open

Most auxiliary steps fail open: the near-duplicate guard, primary transcription (to Whisper), research (down its fallback chain), weather, planning (to a default word target), images, memory, cover art, the catalogue write, backups and the mid-roll. A top-level guard marks any crash as failed so no job is left running forever.

## 10.4 Observability

Every run records a trace of its stages, including the research agent's tool log, and every published episode stores its production record. That record is what §1 is computed from.

# 11 Technology Stack

| Layer | Technology |
|---|---|
| Compute | Modal — CPU orchestration, NVIDIA L4 speech workers, scheduled jobs |
| Language model | DeepSeek V4.1 Flash on the DeepSeek API (OpenRouter standby) |
| Transcription | Gemini 3.1 Flash Lite; Whisper large-v3 fallback (both via OpenRouter) |
| Speech | Chatterbox Regular (Multilingual, experimental); NVIDIA NeMo text processing |
| Orchestration | LangGraph for the script chain; a generic dispatcher for episode types |
| Research | Exa, GDELT, Wikipedia, arXiv, Hugging Face, GitHub, PubMed, Sefaria, Open-Meteo |
| Memory & retrieval | Pinecone; Gemini Embedding 2 |
| Images | fal.ai Flux Schnell (covers), Flux 2 (social images) |
| Storage | Cloudflare R2 (files and content catalogue); Wasabi and Backblaze B2 backups |
| Database | Neon PostgreSQL |
| Web | Astro on Vercel; Send App on Vercel |
| Data | Hugging Face datasets (episodes, transcripts), synced daily |
| Deployment | GitHub Actions: a merge to `main` redeploys the pipeline |
| Audio | FFmpeg |

# 12 Lessons Learned

- **Rules belong where they run.** An LLM review pass that is switched off enforces nothing. Anything that must hold goes in the script prompt or in deterministic code.
- **Name the model you actually get.** When a provider quietly substitutes a model, the on-air credit has to follow.
- **A similarity threshold can't spot a reworded prompt.** A rewrite and a legitimate sequel sat a hundredth apart. A cheap shortlist plus a model's judgement can; failing open keeps the cost of a mistake at one extra episode.
- **Small models make bad number normalisers.** A grammar-based normaliser is faster, deterministic and correct; the small LLMs invented numbers.
- **Profile before buying bigger GPUs.** The largest speed-up in speech came from a stray debugging flag and a CUDA graph, not from more expensive hardware.
- **Make mistakes free.** A 60-second grace window and a duplicate guard cost almost nothing and remove the two commonest wasted runs.
- **Watch the balance, not the error.** Two outages in two months came from empty accounts, not code.
- **Chatterbox Regular over Turbo** (January 2026): about 95% fewer speech hallucinations. Parallel workers and cached voice conditionals took speech from 36 minutes to about 10, and the decoder work above has since taken it to under five.

# Appendix: Changes Since the Previous Edition

| Date (2026) | Change |
|---|---|
| 20 Sep | Every stage on DeepSeek V4.1 Flash; the credit names the model actually served. |
| 9 Oct | Per-episode slate retired; phrase rotation by synonym; sign-off recommendations. |
| 10 Oct | Attention-flag patch and CUDA-graph fp16 decode; speech workers back on L4. |
| 11–13 Oct | Research treated as material; Corn's quips; hosts think first. |
| 14 Oct | R2 content catalogue: publishing without a site build. |
| 15 Oct | Funny by default; Larry's ad every episode; echo cuts; NeMo number normalisation; Whisper via OpenRouter; crash supervisor; failed insert fails the job. |
| 16 Oct | Near-duplicate guard; Hilbert's story leaves the topic. |
| 17 Oct | Hilbert's prompt-address clip; disclaimer in Daniel's cloned voice; 0.97× dialogue tempo; length tiers 15/25/30; research toolkits with Sefaria; world state reduced to date and weather. |

The full history, with the episode where each change first appeared, is in the pipeline changelog. Reference code from this version is at github.com/My-Weird-Prompts/pipeline.