Data Flywheels and Public AI

Nicholas Vincent

2025-08-04

Table of contents

Preface

This is a "mini-book" that discusses "public AI flywheels": software meant to enable people to opt-in to contribute data towards "public AI" causes. The goal of this book is to support efforts build a transparent, people-centric data collection ecosystem that supports the evaluation and training of public-benefit AI models. More frankly, this is way to organize some design notes, practical documentation that's out of scope for a single example projec's repo, and longer abstract writing on the topic.

This document is organized as such:

- In Section 1-4, we first we describe how this document is organized and introduce the Public AI Flywheel concept.
- In Sections 5-7, we discuss one particular implementation of a Minimum Viable Product (MVP) opt-in flywheel meant to accompany a "public AI interface" (hosted interface software that hits various endpoints for "public AI models") that uses a "serverless" app + Git backend approach
  - This MVP focuses on collecting two high-signal data types: exports of "good chats" and "fail chats." This data provides immediate value for model evaluation and, at scale, can be used for fine-tuning. Importantly, collecting a list of good and bad chats is also immediately fun, so contributors can get some value before we reach a threshold of data volume needed to construct a full benchmark or dataset. We expect key ideas discussed in this doc, and concretized in this project, to generalize to other data types.
  - We also provide details on the data retention policy for the Public AI Flywheel and the data policy for the full public AI interface pipeline: from model endpoints to OpenWebUI interface to flywheel platform.

Part I

Intro

1 Introduction

1.1 What is a data flywheel?

What is a data flywheel? Nvidia gives us this definition: "A data flywheel is a feedback loop where data collected from interactions or processes is used to continuously refine AI models." Others have also written on data flywheels (see e.g. a number of helpful blogs from Roche and Sasson, Liu and Del Balso.

In general, a "data flywheel" is a system or set of systems that capture and/or incentive data. A "flywheel" generally differs from a more general data collection or data creation system because the flywheel is embedded into some kind of application (as opposed to e.g. "standalone" data labeling tasks).

Generally, most data collection systems lean more towards using "sensors" (passive, instruments, no active "submit data" step) or forms (active, requiring somebody to click "submit"). Flywheels tend to be lean passive, but this is not necessarily a requirement.

> For some further thoughts on sensors and forms, see this post

1.2 What is a public AI data flywheel?

First, what is "public AI"? The public AI network gives us this: AI with

> "Public Access – Certain capabilities are so important for participation in public life that access to them should be universal. Public AI provides affordable access to these tools so that everyone can realize their potential." "Public Accountability – Public AI earns trust by ensuring ultimate control of development rests with the public, giving everyone a chance to participate in shaping the future." "Permanent Public Goods – Public AI is funded and operated in a way to maintain the public goods it produces permanently, enabling innovators to safely build on a firm foundation."

For more on the public AI concept, see also Mozilla's work in this space and several workshop papers and preprints (from RegML 2023 at NeurIPS, CodeML 2025 at ICML, a recent workshop on Canadian Internet Policy).

Our focus in this mini-book is building "public AI" flywheels. To summarize heavily – in trying to achieve all the principles laid out in the above work that tries to define "public AI", those building public AI flywheel will face some unique challenges in the implementation of data flywheels; they may not be able to do what private orgs can do.

In building public AI data flywheels, we are trying to create a feedback loop to improve AI. However, we likely want to start from a position of accessibility (including providing an accessible explanation of exactly what happens to any data a user creates) and accountability (so people have real agency over the shape of the data pipeline).

Of course, it's worth noting that some particular public could deliberate and make a collective decision that they prefer a more "traditional approach" to data. In this mini-book, we are taking the stance that it's best to start from a position of leaning heavily towards an opt-in approach. We start by minimizing usage and retention of data; data that is used in the flywheel to train AI should be provided via an opt-in by highly informed users.

1.3 Core Principles

We can translate the core principles of public AI to the data flywheel domain and arrive at roughly four requirements:

- Transparency for Informed Consent: Users must be fully informed about the model, its developer, and the ramifications of their contribution. A detailed FAQ and a clear consent module are required before any data is shared. To some extent, maximally informed consent will require the active expenditure of resoures to improve the public's AI literary (i.e. we need to build AI literacy focused systems and perhaps even pay people for their attention). We need systems that really do inform people. Luckily, that's something it seems like AI can help with!
- Data Rights: A public AI data flywheel must empower users with control over their data, mirroring GDPR principles and similar regulations (this is also practically important for compliance). This includes the right to access (`$Art. 15$`), rectify (`$Art. 16$`), erase (`$Art. 17$`), and port their data (`$Art. 20$`). Key exemplar: Mozilla Common Voice).
  - We note that data rights sometimes conflict with a "fully open" ethos; we will attempt to mitigate these tensions to the best extent possible!
  - We also note that public AI faces some unique challenges with cross-jurisdiction compliance; we discuss this at a high-level later on. #todo link exact section
- Balancing reputation and pseudonymity: To the extent possible, we believe it is valuable to offer people to ability to contribute to data flywheel with some kind "real account" attached, so people can earn credit and reputation. But this must be balanced with the benefits of "light-auth" or even anonymous contribution (#todo cite CDSC work on anon contributions).

  - In our MVP (discussed in the next chapter) an OpenWebUI or HuggingFace account is required to make contributions, but users can choose to remain anonymous or use a pseudonym for the public data release.

- Purpose Limitation & Licensing: Users must be able to specify their preferences for how their data is used (e.g., for public display and evaluation vs. for model training). This is captured using (new) IETF AI Use Preferences and Creative Commons Preference Signals. We will discuss below how this might extend to other preference signal proposals and/or technical approaches to gating data.

  - This is critical for answering a likely FAQ around public AI data – if you succeed in creating actually useful training data or new benchmarks, won't private labs just immediately use that data as well?

2 What data and why

> Key insights: in general, we want more records that contain high-quality signals and/or observations about the world to be available to public AI organizations for training and evaluation.

If we want to build a data flywheel, we probably need to specify what kind of data we want and why exactly want it! At its core, "data" is useful for AI (and for other things!) because it provides information about the world.

In general, its intuitive to most people that having more information will (generally) lead to better decision-making. [^1]. Although there some scenarios we might come across (or invent) where getting more information is not helpful – because we might not have "room" in our memory for more data, or some records might not help us at a certain task, or data causes our model to get worse in some sense – most people benefit from having more records of high-quality observations and signals.

So let's put these more complicated cases aside for now, and make the assumption: in expectation, acquiring more high-quality data (that is accurate, or reflects insight) is useful. Oftentimes assessing data's quality, or its truthiness, or its insightfulness, is not at all easy! With this assumption in mind, we can speak generally about the types of data we might acquire through a flywheel how it will be useful.

2.1 An overly detailed accounting of all the ways we might generate LLM pre-training data

LLM pre-training data consists of (at a high level):

- Human-authored natural language: the open web (cleaned), books, encyclopedias, news, forums/Q&A, transcripts (talks, meetings, podcasts), documentation, and manuals.
  - And now, some non-human-authored natural language.
- Code: source files, perhaps with licenses and provenance, issue threads, commit messages.
- Semi-structured text: tables, markup, configs (HTML/Markdown/LaTeX/YAML/JSON) that carry schema and relationships.
- Multimodal pairs (for VLM/ASR pretraining): image+text, audio+text, video+text, and associated captions/alignment.

  - Here, the pairing is a critical characteristic that makes this data unique. This implies somebody has looked at the each item in the pair and confirmed a connection (though paired data can be produced in an automated fashion).

- Metadata about data: language, domain/topic tags, timestamps, links, authorship/attribution, *license*, and (in our case) AI preference signals.
- Quality signals: dedup scores, perplexity filters, toxicity/PII flags, heuristic or model-based ratings—used to weight or exclude.

At the lowest level, a person does something that leaves a digital trace: typing, speaking into a mic, using some kind of alternate controller, etc. They might also operate a camera or other sensing instrument that captures signals from the world.

After typing (or other input), they might use a terminal or GUI to send their inputs into some structure – by committing code, editing a wiki, responding on a forum).

Often, this person has a goal and/or a task they want to complete—ask a question, teach or correct something, build software, file a bug, summarize a meeting, translate a passage, or simply *react* (like/flag/skip).

Examples include:

- Asking a model a question and marking the response "good" or "fail", optionally with a short note about *why*.
- Corrections/edits: rewriting a wrong answer; adding a missing citation; supplying a step-by-step solution.
- Pairwise preferences: "A is better than B because …" (useful for preference learning/DPO).
- Star ratings / rubrics: numeric or categorical grades on axes like factuality, helpfulness, tone, safety.
- Tags and metadata: topic ("tax law"), language ("id-ID"), difficulty ("HS"), license (CC-BY-SA), and AI preference signals.
- Synthetic tasks: user-written prompts + *ideal* references (gold answers, test cases, counterexamples).
- Multimodal: an image with a caption; an audio clip with a transcript; a diagram with labeled parts.
- Programmatic contributions: code snippets with docstrings/tests; minimal reproductions of a bug.
- "Negative" structure: anti-patterns, jailbreak attempts, hallucination catalogs.

2.2 Some other examples of data

- Explicit numerical ratings of items

- Implicit feedback about items (clicks, dwell time, scroll/hover, skips/abandonment—when captured with consent and documented)
- Edits/diffs (how a draft changes—great signal for learning to revise)
- Flags and rationales (spam, unsafe, off-topic, with short free-text)
- Search queries + selected result (query→choice pairs; useful for intent and relevance)
- Structured forms (error taxonomy, severity, reproduction steps)
- Trace data from tools (with consent): which functions were called, which docs were opened, etc.—often better as *evaluation context* than as training text.

[^1] A bayesian might say: data is *evidence* that updates a prior into a posterior via Bayes' rule; the "goodness" of a dataset is how much information (likelihood ratio / bits of surprise) it carries about the hypotheses we actually care about. A frequentist might say: data are *samples* from some process; more (and more representative) samples tighten confidence intervals and reduce estimator variance (roughly with $1/\sqrt{n}$), so sampling design and coverage matter as much as sheer volume.

3 The Data Pipeworks: From Human Knowledge to Deployed AI (and Back Again)

This is a summary of a longer Data Leverages Newsletter post.

#todo re-add cites from the original post!

To further motivate the idea of data contribution with public AI principles, it's worth a brief discussion of what the overall "data pipeworks" of the AI industry looks like from a zoomed out view.

Key takeaways

- Modern AI can be understood as a five-stage pipeworks: (1) Knowledge & Values -> (2) Records -> (3) Datasets -> (4) Models -> (5) Deployed Systems.
- Treating AI as a cybernetic system puts feedback and control at the center. Contributors can steer outcomes by shaping data flow (more on the next chapter).
- Human factors dominate AI capabilities because they shape what gets recorded upstream. Interfaces, sensors, and incentives are therefore core AI R&D.
    - some trends may shift this – RL in real life, #todo cite experiental learning
- Properties of data create collective action problems (social dilemmas) that require markets, coalitions, and policy to fix.
- For public AI flywheels, thinking in terms of data piepworks reveals "insertion points" to add transparency, consent, rights, and preference signals so democratic inputs actually move the system.

3.1 Why a "pipeworks" view?

Most technical AI work zooms in on a clean optimization problem. But questions about who benefits, who participates, and how AI affects society live upstream and downstream of that problem. The Data Pipeworks zooms out. It describes the end-to-end flow by which human activity becomes records, then datasets, then models embedded in systems that act on the world—and thereby change the future data we can collect. That circularity is the opening for governance.

This view pairs naturally with cybernetics/control: identify system state, actuators, sensors, and feedback loops; then decide which loops to strengthen, dampen, or rou

3.2 Five stages of data

1. **Knowledge & Values (Reality Signal).** Humans (and the physical world) generate the latent "signal" AI tries to model (facts, preferences, norms). We don't presume computability; we note its existence to emphasize sampling implications.
2. **Records (Sampling Step).** Interfaces and sensors transform activity into structured records (forms, clicks, edits, uploads; camera/mic/IoT streams). Design choices here shape what becomes legible to AI. Key idea: everything either leans "sensor" or "form".
3. **Datasets (Filtering & Aggregation).** Organizations filter, label, merge, and license records under social, economic, and legal constraints. This determines coverage, bias, and what's even available to learn from.
4. **Models (Compression).** Learning compresses datasets into input–output mappings. Model choices are path-dependent on Stages 2–3; data defines the feasible hypothesis space.
5. **Deployed Systems (Actuation).** Models are embedded in products, workflows, or infrastructure, producing value and externalities. Deployment feeds back by altering incentives and future record creation.

Design note: because influence composes across stages, small, well-placed interventions upstream can dominate large downstream tweaks.

3.3 Why this matters for governance and alignment

- Human factors are primary. The distributions the AI field is optimizing over are created, not discovered. Interfaces, defaults, prompts, consent flows, and incentives shape the topology of AI work.
- Social dilemmas are inevitable. Contributing high-quality records to a shared system is a collective action problem (free-riding, failure to reach critical mass). Today's "dictator solution" (opaque scraping) collapses when people gain data agency.
- Data leverage (next chapter) is the steering wheel. Individuals and groups can alter records, licenses, and access. This allows people to steer model behavior by modulating data flow rather than model internals.
- Pluralism becomes measurable. Tracing contributions lets us quantify relative weight of individuals and communities, enabling pluralistic governance and new not

3.4 Where to place the levers (for public AI flywheels)

- Stage 1 to 2 (Knowledge to Records): invest in interfaces and sensors with informed consent; design contribution prompts and micro-tasks; support pseudonymity and reputation choices. Aim to raise signal quality and widen participation.
- Stage 2 to 3 (Records to Datasets): attach licenses and AI preference signals per record; validate, de-duplicate, and redact PII; publish partitioned releases. Make rights legible and keep high-trust, high-reuse bundles.
- Stage 3 to 4 (Datasets to Models): enable data markets and coalitions, attribution, and sampling weights; build evaluation sets tied to provenance. Align training with community intent and enable bargaining.
- Stage 4 to 5 (Models to Systems): publish transparent deployment notes, opt-outs, and model cards tied to data buckets. Surface externalities and set expectations for use.
- Stage 5 to 2 (Feedback loop): close the loop with flywheel UX. Leaderboards, grants, bounties, governance hooks (votes, preferences) to sustain contributions and invite further steering.

Our MVP ("Serverless + Git" flywheel) primarily operates across 2-3-5: it turns opt-in records into licensed, preference-tagged datasets, publishes gated and public releases, and feeds visible outcomes back to contributors.

3.5 Implications for research and practice

Building flywheels are part of broader agenda to enable a data pipeworks. More in the next chapter on how data contribution through flywheels (including licensed or user-restricted contribution) interplays with data protection, data strikes, markets, etc.

3.6 A compact mental model

- Sensors and interfaces decide what counts.
- Filters and markets decide what persists.
- Compression decides what generalizes.
- Deployment decides what changes next.
- Governance decides who gets to steer.

Public AI flywheels turn that loop into a participatory control system: contributors see consequences, express preferences, and are (hopefully) rewarded for adding high-signal records.

4 Flywheels and Markets

> Key insight: Beyond improving public AI systems, getting public AI data flywheels right can make it easier for people to use data flow as a lever for governance and alignment.

Based on our previous chapter about "what data and why", we can arrive at a very obvious argument for a data flywheel: the flyhweel will produce data, and that data will make AI better!

But this isn't the only benefit of getting building flywheels in a "public AI" manner. Doing so can also enhance the amount of agency that people have over data flow, and "making with data" possible such that the public has more power to govern and align AI systems.

In short, AI is somewhat unique relative to other technologies, because of its data dependence. Data comes from people. The fact that this powerful technology has a dependency on people from around the world means that AI has a natural "governance lever".

Setting up a public AI data flywheel is thus important not only to make AI better; it also has the potential to solve some (but not all!) of the thorny governance and alignment challenges that AI poses.

You can read about data leverage via a newsletter or even via a dissertation. For a short summary, check out Plural AI Data Alignment.

It's worth pulling out two distinct ways that flywheel can interact with AI and governance

It's worth pulling out two distinct ways that a flywheel can interact with AI and governance:

- A thin flywheel, with no attempt to capture creator intent, licensing, or data rights, simply increases available data. It may make public models a bit better and keep public labs competitive at the margins, but it does not change the bargaining relationship between contributors and model builders.
- A governed, opt-in flywheel captures provenance, per-item license and AI-use preferences, contributor identity or pseudonym, and release discipline. This converts contributions into addressable units that can be assembled, priced, withheld, or targeted—opening the door to markets and, if necessary, strikes.

4.1 How an opt-in flywheel enables markets

An opt-in flywheel can create the prerequisites for functioning data markets without turning the project into "just a marketplace."

1. Units and provenance Each contribution is a unit with provenance, license, usage preferences, minimal schema, and (optionally) reputation of a contributor or collective. That makes it legible enough to transact on.
2. Permissioning and packaging By establishing practices around setting use requirements (e.g., "eval-only", "train-allowed-if-model-is-open"), it is possible to either (1) provide nicely packaged releases (monthly, by topic/language/domain) of data (data user is paying for premium like Wikimedia Enterprise) or (2) collectively move from sending data to an open repo to a gated repo managed by a market agent.
3. Discovery and price signals Leaderboards, scarcity tags (rare language/domain), and quality scores effectively begin to generate price signals. A bounty board ("need 5k labeled failures in X") converts demand into targeted supply. In this sense, bounty boards for OSS also enable market dynamics around outputs that are eventually made public with license restrictions.
4. Clearing and settlement Because contributions are opt-in and addressable, enterprise-style payouts can be routed to individuals or collectives, with rules like "70% to contributors pro-rata by weight; 30% to the commons."
5. Collectives as market actors Co-ops/unions/DAOs can represent contributors, negotiate bundle terms, run audits, and set default preferences. The flywheel provides the shared ledger and release cadence that markets need.

Note: a key idea here is that it's possible to enable market activity while keeping the data open-but-gated-and-restricted or by using the flywheel as a stepping stone towards a more "property-like" market.

4.2 How an opt-in flywheel enables strikes (or credible refusals)

"Strike" here means a coordinated, temporary withdrawal or constraint on new high-signal contributions or releases—not sabotage, and not retroactive deletion of already-licensed copies.

What makes strikes possible:

- Voluntariness is preserved. Because contribution is opt-in, non-participation is a legitimate default.
- Release control. A waiting-room, processing, release pipeline provides a natural "valve" for cadence changes or strikes.

- Shared visibility. Everyone sees dependence on fresh contributions (e.g., evaluation drift). Visibility creates leverage.

What a strike can look like:

- Quality freeze. Contributors keep using systems but withhold labeled "good/fail" chats or corrections for a period.
- Selective embargo. A community with scarce data (language/domain) pauses releases or flips new records to "evaluation-only."
- Preference shift. New contributions change AI-use preferences to deny training unless a stated condition is met (funding, governance, attribution).
- Rate limit. Collectives cap monthly volume to force negotiations on price or terms.

What a strike cannot do (and shouldn't promise):

- Undo past licenses. Items released under irrevocable terms (e.g., CC0, CC-BY) remain available.
- Prevent copying entirely. Public releases can be mirrored; anti-scraping reduces risk but does not eliminate it.
- Guarantee compliance outside the ecosystem. Preference signals work when counterparties agree to honor them or when law/policy backs them.

Net effect: strikes are most effective where freshness and scarcity matter (evaluations, rapidly changing domains, rare languages), and least effective where substitutes or synthetic data can plausibly fill gaps.

4.3 Preconditions the flywheel must provide

- Clear, per-item license and AI-use preference capture and propagation
- Auditable provenance and contributor linkage (including pseudonyms)
- Release discipline (cadence, versioning, checksums) and the ability to pause cadence
- A shared metrics surface showing demand, scarcity, and impact
- Collective controls: settings a group can change together (defaults, embargo toggles)
- Payment rails or grant accounting tied to releases, not pageviews

4.4 Limits and failure modes to plan for

- Irrevocability vs. agency. If the goal is strike leverage, discourage CC0 for scarce data; offer eval-only or time-boxed terms.
- Leakage. Plan for mirrors; design incentives that make compliance more attractive than defection.

- Substitution. Some domains can be synthetically approximated; focus markets/strikes where real-world records are hard to fake.
- Coordination overhead. Without lightweight collective tools, only well-organized groups can act.
- Legal and policy constraints. Collective bargaining around data may raise jurisdiction-specific questions; stay within applicable law.

4.5 A practical path from MVP to markets (and optional strikes)

- v0: Opt-in contributions with per-item license and AI-preference signals; public/gated releases.
- v1: Contributor dashboards, scarcity/impact badges, and a bounty board.
- v2: Collective accounts that can set defaults, pool payouts, and flip embargo/quality-freeze toggles.
- v3: Formal procurement: RFPs for datasets/evals with escrowed funds and delivery criteria. This might mean some collectives STOP pushing to the public repo.
- v4: Policy hooks: standardized terms that institutions can adopt; preference signals integrated in provider contracts.
- v5: Federation: multiple flywheels interoperate on releases and preferences, raising both market depth and strike credibility.

5 Rationale (and other options for flywheel variants)

5.1 Purpose of this section

This section gives more context about th many ways we might built flywheels, and lays out alternative governance paths and a future work (in particular, a focus on futures that involve healthy data markets, data intermediaries, federated learning, etc.)

We also discuss why we think an approach that includes a minimal retention frontend + opt-in flyhweel platform can serve as a pragmatic bridge to more advanced approaches. For instance, we can use the patterns and concepts used here to move towards independently governed data co-ops, eventual **federated learning, etc.

5.2 More on all the other approaches we could've taken

In choosing our architecture, we consider the following questions:

5.2.1 Where does the "final" data live?

- **Centralized Database:** Traditional server-controlled storage (PostgreSQL, MongoDB, etc.)
- **Public Repository:** Version-controlled platforms (GitHub, GitLab, Hugging Face Hub)
- **Totally Local:** Federated model where data stays on user devices
- Other options
  - **Distributed Network:** Peer-to-peer systems (IPFS, BitTorrent)
  - **Something hybrid?:** Metadata centralized, actual data distributed

5.2.2 When is the user prompted to contribute?

- **Proactive:** User initiates contribution unprompted (e.g., "Share this chat" button)
- **Reactive:** System prompts based on signals (e.g., after thumbs down or trigger word, ask "What went wrong?")
- **Passive:** Automatic collection with prior consent (e.g., telemetry, browser extension)
- **Scheduled:** Regular prompts (e.g., weekly "best conversations" review)
- **Task-Based:** Specific requests for data types (e.g., "Help us improve math responses")

5.2.3 What information object is created?

- **Simple Signal:** Binary feedback (/), star ratings, or flags
- **Annotated Conversation:** Full chat with user corrections, ratings, or notes
- **Preference Pair:** A/B comparisons between responses
- **Examples:** User-created prompts and ideal responses
- **Structured Feedback:** Form-based input (error type, severity, correction)
- **Multimodal Bundle:** Text + images + voice + metadata
- **More advanced structured data** …

5.2.4 When is the data processed?

- **Pre-submission:** Client-side processing before data leaves user's device
- **On-submission:** Real-time processing during the contribution flow
- **Post-submission:** Batch processing after data is received
- **Pre-publication:** Review and processing before making data public
- **On-demand:** Processing happens when data is accessed/downloaded

(In practice, there may be some processing at various steps, but it is important to clarify this to users)

5.2.5 How is the data accessed?

- **Direct Download:** Raw access to complete dataset (with rate limits)
- **API Access:** Programmatic access with authentication and quotas
- **Static Site:** Read-only web interface with anti-scraping measures
- **Gated Access:** Application/approval process for researchers
- **Hybrid Access:** Public samples + gated full access, or public metadata + restricted content
- **Streaming Access:** Real-time feeds for continuous model training

5.2.6 How much friction is acceptable?

- **Zero-Friction:** One-click actions with no interruption
- **Low-Friction:** Modal popup or inline form
- **Medium-Friction:** Redirect to separate interface
- **High-Friction:** Multi-step process, account creation, or technical skills required

5.3 Some Categories of Architectural Models

5.3.1 Standard "PrivateCo" Web App

An obvious option is to simply build a hosted "standard" "PrivateCo" /start-up style web app. In fact, in some contexts it may make sense to skip building an opt-in flyhweel and simply use the data generated by users directly for training, eval, etc. While one could argue that the Terms of Service for many existing tech products do make these products "opt in" in some sense, there are also serious downsides to the status quo (see e.g. Fiesler, Lampe, and Bruckman 2016.)

While perhaps some users might prefer even prefer a start-up style model, we believe this would not be a good starting place for a public AI interface. We also believe it's important to communicate to users how the public AI interface differs from e.g. using ChatGPT, Gemini, or AI overviews via search.

This approach doesn't constrain how we answer the above questions, although it's highly likely that data lives in centralized database and company can use telemetry/surveillance and keep friction low.

- **Example Stack:** Django/Rails + PostgreSQL, Next.js + MongoDB

If we use a PrivateCo model with focus on telemetry, our flywheel might look like:

5.3.1.1 Direct Telemetry

- **Where data lives:** Centralized analytics database
- **When prompted:** Passive (continuous collection)
- **Information object:** Simple signals with context IDs
- **When processed:** On-submission (real-time pipeline)
- **How accessed:** Aggregated dashboards only (no raw access)
- **Friction level:** Zero
- **Pros:** Massive scale, unbiased sampling, real-time insights
- **Cons:** Limited richness, privacy concerns, no corrections
- **Example Stack:** ClickHouse/BigQuery + streaming pipeline

5.3.2 Git/Wiki Platform

Another option to build a "very active flywheel" (that arguably stretches the definition because friction will be very high) is to just deploy a server for git or wiki style peer production.

Now, we likely constrain our answers to the above questions:

- **Where data lives:** Public repository
- **When prompted:** Proactive (user initiates)
- **Information object:** Markdown-formatted conversations
- **When processed:** Pre-submission (user does it) + CI/CD validation
- **How accessed:** Direct download via Git + web interface
- **Friction level:** High (technical knowledge required)
- **Pros:** Maximum transparency, built-in versioning, low cost
- **Cons:** Excludes non-technical users, limited data types
- **Example Stack:** some combo of MediaWiki, GitHub, GitLab, HuggingFace + CI/CD validation

5.3.3 Serverless + Git Platform

- **Where data lives:** Public repository
- **When prompted:** Proactive or reactive
- **Information object:** Structured data files (JSON/YAML)
- **When processed:** On-submission via serverless function
- **How accessed:** Git access + static site generation
- **Friction level:** Low (automated complexity)
- **Pros:** Transparency + usability, serverless scaling
- **Cons:** Cold starts, API rate limits, complex error handling
- **Example Stack:** Vercel/Netlify + GitHub API + Hugging Face Hub

5.3.4 Federated Learning Model

One radically different approach might involve using federated learning.

- **Where data lives:** User devices (distributed)
- **When prompted:** Passive with consent
- **Information object:** Model gradients or aggregated statistics
- **When processed:** Pre-submission (on-device)
- **How accessed:** Only aggregated model updates available
- **Friction level:** Zero after setup
- **Pros:** Maximum privacy, no data transfer, infinite scale
- **Cons:** Complex implementation, limited debugging, device requirements

- **Example Stack:** Flower/TFF + edge deployment

5.3.5 Browser Extension

We could implement a flywheel that relies on users downloading a browser extension! This only reflects a data ingestion choice: can be used with various backend choices above.

- **Where data lives:** Centralized or distributed
- **When prompted:** Proactive or passive
- **Information object:** DOM captures, interaction logs, selections
- **When processed:** Depends on backend
- **How accessed:** Depends on storage choice
- **Friction level:** Very low after installation

5.3.6 P2P Network Model

- **Where data lives:** Distributed across peer nodes
- **When prompted:** Passive (background sharing)
- **Information object:** Torrent-style data chunks
- **When processed:** Pre-submission by contributor + network validation
- **How accessed:** P2P client required for full access
- **Friction level:** Medium (client installation)
- **Pros:** No infrastructure costs, censorship resistant
- **Cons:** Availability issues, complex coordination
- **Example Stack:** libp2p + BitTorrent protocol + DHT

5.4 Scenario Walkthroughs: A Practical Comparison

Here, we walk through two common scenarios and describe what happens (in one sentence) for each of the architectures described above.

#todo: these could be made crisper to highlight the key differences better (But also be honest about where there are similarities)

5.4.1 Scenario A: User marks a chat as "Good" – when does processing happen?

- **Web App:** Redirects to platform, PII scrubbed on submission, available via API after review
- **Git/Wiki:** User removes PII manually, creates PR, instantly visible on merge
- **Telemetry:** Signal sent, processed in real-time, only visible in aggregates

- **Hybrid:** Signal sent immediately, full chat processed if shared
- **Serverless+Git:** Modal appears, serverless function strips PII, PR created automatically
- **Federated:** Local processing only, contributes to next model update
- **Extension:** Captures state, removes PII client-side, sends to chosen backend
- **P2P:** Processes locally, shares with peers who validate before propagating

5.4.2 Scenario B: User corrects a factual error

- **Web App:** Editor interface, toxicity check on submission, published after human review
- **Git/Wiki:** User edits markdown, CI/CD checks format, visible immediately on merge
- **Telemetry:** Only captures "error" signal, no correction possible
- **Hybrid:** Error signal triggers correction UI, correction queued for review
- **Serverless+Git:** Inline correction, automated PII/toxicity checks, PR needs approval
- **Federated:** Correction processed locally, differential privacy applied
- **Extension:** Highlights error, pre-processes correction, sends to backend
- **P2P:** Broadcasts correction, network consensus before acceptance

5.4.3 Scenario C: Accessing the contributed data

- **Web App:** Researchers apply for API key, public sees samples on static site
- **Git/Wiki:** Anyone can clone repo, but rate-limited through CDN
- **Telemetry:** Only aggregated statistics available via public dashboard
- **Hybrid:** Public can see signals dashboard, researchers apply for conversation access
- **Serverless+Git:** Public (or gated) repo with all data, static site with search/filter
- **Federated:** No direct data access, only model checkpoints released
- **Extension:** Depends on backend choice, typically follows that model
- **P2P:** Must run client to access network, can specify data sharing preferences

5.5 Frontier approaches data cooperatives, federated learning, and more

In many cases, users may want to have data governed by community organizations (e.g., organized by domain/region/language) that hold rights and decide release cadence, licensing defaults, and benefit policies.

We note that because our implementation is built on top of open-source software, communities can easily choose to deploy their own OpenWebUI instance and their own data flywheel and effectively operate entirely parallel, self-governed instances. If they also choose to share opt-in data via similar licensing and preference signal approaches, such datasets could be easily

merged – but with fine-grained adjustments to precise details (e.g., slight modifications on retention, access, release cadence, content moderation, adn so on.) Of course, data co-ops may choose to use quite different technical stacks. This approach is just one among many.

It may be possible to also move from an opt-in data flywheel approach to a federated learning-first approach. Here, model training occurs across user or institutional nodes; only gradients/updates (with privacy tech) are centralized. The dataset remains partitioned or local; central custodian minimized. This approach would:

- Reduces central data custody and breach surface
- Aligns with data-residency and institutional constraints
- Enables "learning from data that can't leave"

But has some major downsides / existing barriers:

- Harder reproducibility and data auditability
- Complex privacy stack (secure aggregation, DP, client attestation)
- Benchmarking must be redesigned (federated eval)

This is a bigger leap, but we believe it's important to begin to think about how the implementation of the Public AI Data Flywheels might support communities wishing to transition towards an FL approach.

One rough sketch might look like: * Build the MVP defined in Chapter 2 * Ship license + AI-preference metadata (MVP). * Maintain gated HF releases and public leaderboards/full data access. * Publish provider-payload transparency and link to provider terms (no guarantees). * Process deletions via HF mechanisms when possible; keep our mirrors in sync. * Phase 1 — Co-op pilots * Charter one or two community co-ops; define bylaws, scope, and release cadence. * Spin up many instances of interface + flywheel combos (can fork software directly, or use similar approaches) * Establish a concrete sharing / merging plan * And beyond! * Once several independent data communities, are operated, it might be possible to move from lightweight sharing and merging to more serious federation with technical guarantees. Perhaps this might start with federated evaluation and then move to federated training. Much more to do here, out of scope for this document.

Part II

A Worked Example: MVP Flywheel

6 The Serverless + Git MVP

This section describes a particular minimal viable product for an independent data flywheel aimed at collecting data that can be shared publicly, but with licensing, usage preferences, and anti-scraping. The idea is to produce data that is useful to public AI labs.

6.1 Overview

Our initial MVP of the flywheel is a "Serverless + Git Platform" approach. It is meant to be a robust and scalable starting point for the data flywheel. It strictly separates the "write" (contribution) and "read" (display) components of the system.

The overall goal is to enable opt-in contributions of data (prompt, output, good/bad, optional metadata) with relatively open licenses (per-item Creative Commons license: default is CC0, CC-BY, CC-BY-SA), state-of-the-art (caveat: also experimental / untested) preference signals (using IETF aipref draft spec + CC Preference Signals draft spec) and enforcement. The data is distributed via:

- Hugging Face (gated) — full bundles by "license/usage bucket"; access via request.
- Public site — leaderboards + full dataset access; Cloudflare WAF/bot controls to discourage bulk scraping.

For full details, see the separate repo and its readme: #todo fill me in

For a short summary of the technical approach, see below bullet points:

- Frontend: A Next.js application hosted on Vercel, providing a simple, static interface for contributions.
  - No major lock-in here. Can easily be swapped for a lightweight static site, other modern web tech, etc.
- Authentication: User identity is managed via Auth.js (Next-Auth), using Hugging Face (HF) as the exclusive OAuth provider. This allows for clear attribution of contributions to a user's public HF username (if they choose to).
- Data Storage: The single source of truth is a Hugging Face Dataset repository, which functions as a "Git-as-a-database."

  - Starting with HF as it is a platform with specific focus on AI datasets and open culture

- Waiting room approach: The implemented workflow follows a two-stage "waiting room" pattern to ensure data quality and safety:
- How contribution works:
  - A user logs in with their Hugging Face account.
  - The interface accepts contributions in three ways: by uploading an OpenWebUI JSON export file, via URL params, or by pasting plain text.
  - The frontend parser automatically detects the OpenWebUI format, extracts metadata (like `model` and `tags`), and prepends it as YAML frontmatter to the chat content.
  - Users choose a label for the chat type ("Good Chat" / "Fail Chat") and attach (1) a Creative Commons license and (2) an IETF aipref and/or Creative Commons Preference Signal to signal preferences around AI use of the contribution.
  - A pseudo-anonymity system allows users to contribute publicly with their HF username, as "anonymous," or with a custom pseudonym.
  - An informed consent checkbox, linked to Terms of Service and FAQ pages, is required for all submissions.
  - Upon submission, a serverless function writes the contribution not to the final dataset, but as a new, single JSON file in a `_waiting_room/` directory within the Hugging Face repository. This operation is fast and avoids write conflicts.

- How processing of the "waiting room" works:
  - A separate, asynchronous script is run on a regular schedule (e.g., as a daily GitHub Action).
  - This script fetches all pending files from the `_waiting_room`.
  - It validates each contribution and includes a placeholder for future content moderation and PII checks.
  - Validated contributions are batched and appended to a final, organized dataset file (e.g., `data/2025-08.jsonl`). Contributions are further bucketed by license/prefs.
  - To ensure atomicity, all file additions (to the final dataset) and deletions (from the waiting room) are performed in a single commit to the Hugging Face Hub. #todo when scaling, need to consider race conditions around the processing!

6.2 Advantages of a Serverless + Git approach

A serverless + Git stack keeps the "write path" lightweight for contributors and cheap to operate. Functions spin up on demand and idle to zero, so we can avoid paying for boxes that sit around; the trade-off is cold starts, which are well-documented and can be mitigated with provisioned concurrency when needed.

On the "read path," a static site on a global CDN gives instant distribution and low operational overhead. Pages (e.g., from Cloudflare, something like GitHub Pages, or similar) can read directly from the Git "source of truth" and serve assets from edge locations by default, which is exactly what we want for a browsable leaderboard and dataset browser.

Additionally, using the Hub (Git-backed) as the source of truth buys us a public audit trail and first-class versioning semantics. HF's dataset repos are literally Git + LFS, with revision pinning via commit/tag/branch; storage is backed by object storage and scales. That maps cleanly to our "waiting room → release buckets" workflow and makes it easy to diff changes over time. (relevant HF docs: datasets, storage)

Moderation and PII handling are naturally centralized in the processing step. Because we trigger the write as a small file into a staging path and move it during a scheduled job, we can run filters, de-dup, and attach license/pref metadata before publication without asking contributors to learn tooling.

Basic safety and access controls are pragmatic at the edge. Cloudflare's WAF/Bot Management and newer "AI bot" controls give us a reasonable anti-scraping posture for public downloads and pages, even if nothing on the open web is truly copy-proof. Recent product updates explicitly target AI crawlers, with default blocking and challenge flows we can enable. (Cloudflare Docs, WIRED, Business Insider)

Finally, the "preferences and licenses" story fits the stack. Dataset cards and metadata natively expose a `license` field and other tags (Hub UI and YAML), and CC licenses give clear obligations (e.g., BY, BY-SA) we can enforce in packaging and docs. That lets us partition releases by license and publish compatibility notes in a way downstream users can actually follow. (Hugging Face, Hugging Face)

6.3 Disadvantages (and what to watch)

Trust centralizes in the processor. Contributors have to believe the middle layer won't silently drop or reshape submissions. If/when we introduce event-driven ingestion (queues/streams), we must design for retries and duplicates.

UX isn't perfectly "instant." There's an inherent gap between a user pressing "share" and seeing their item on the public site, because we run validation and batching. That's a conscious choice, but we should set expectations and likely provide some kind status expectations.

Operationally, serverless isn't "no ops," it's "different ops." Cold starts exist (especially on sporadic paths), API limits are real on the platforms we hit (GitHub has documented ceilings; HF also rate-limits writes/reads even if specifics vary by endpoint), and "pay per use" can surprise us at scale without cost guardrails (see e.g. GitHub Docs, GitHub Docs).

There's also platform coupling to be consider. Using specific hosted CI/CD, serverless runtimes, and hub APIs creates a degree of vendor lock-in; this is a known trade-off in serverless

architectures and something we can blunt with portable formats (JSONL), documented exports, and "boring" interfaces (Git). (CNCF)

Compliance remains non-zero. Deletions across mirrored artifacts (Hub revisions, static site snapshots, downstream forks) take a clear policy and a repeatable playbook. For licenses, we're responsible for honoring CC obligations in our packaging and comms (e.g., keeping BY attribution fields intact; not mixing BY-SA content into incompatible bundles). CC's legalcode and guidance make those obligations explicit; our tooling should, too. (Creative Commons)

Net: Serverless + Git gives us a pragmatic bridge—fast contributor UX, public versioning, cheap distribution—while we invest in moderation, idempotent processors, and clear license lanes. If we communicate the review gap, publish the processor's rules, and keep escape hatches (export scripts, mirrors), the trade-offs are acceptable for an MVP and refine-able over time.

6.4 Other reading:

- https://arxiv.org/abs/2109.02846
- https://datascience.codata.org/articles/10.5334/dsj-2021-012

7 Data Flow and Data Retention

Last updated: 2025-08-03

This document describes one possible data retention policy for a public AI data flywheel connected to a public AI chat frontend (or other frontend). It describes the data that is collected, when it's produced, how long we keep it, how contributor license & preference signals work (beta), and how our distribution channels operate.

This chapter is written mainly as a piece of reference material to highlight the flow and retention considerations that arise in building or deploying a product like the Flywheel MVP.

To contextualize the data retention policy, it may be useful to first review a bullet point, plain language description of all the ways that users create data in their use of the Public AI Chat Frontend, across 4 distinct websites:

- User visits the "landing page" {{landing_page_link}}
  - user can enter a query as guest
    - created: a chat object
    - chat is sent to model provider
    - chat is stored, associated with generic guest account
- user can proceed to "the main app" {{app_link}}
  - user enters name, email, password. Handled by OWUI auth code.
  - use can enter queries. Creates a chat object associated with the user.
    - see OWUI chat schema for all possible fields
    - key fields: free text entry, feedback data
    - chats are stored in the OWUI database so that users can browser their chat history, but are NOT used for training or eval.
    - some standard data is stored for basic admin/engineering needs (request volume, errors, etc.), with short retention
- in current dev version, user can also share to openwebui.com if they wish to (requires an account with the openwebui community platform)
- in launch version, user can share to flywheel, on a separate https://optinflywheel.com (#todo exact url)
  - on sharing, a public chat object is created. This object will live in a gated HF repo and a public static site

7.1 Glossary of Defined Terms (for this Chapter)

"{The Public AI Chat Frontend}" or "Frontend" means the hosted interface described in this document that allows users to issue prompts to third-party model endpoints.

"Open WebUI Instance" or "OWUI" means the hosted Open WebUI application at {{app_link}} (or successor URLs) that provides optional accounts and chat history.

"Opt-in Data Flywheel" or "Flywheel" means the separate contribution and distribution platform at https://optinflywheel.com (or successor URLs) through which users may opt in to contribute data for public evaluation and research use.

"Provider(s)" or "Model Endpoint(s)" means third-party model services (e.g., national labs, commercial providers) that receive user prompts and return model outputs. The Frontend is a gateway to these services and does not control their retention, training, or use practices.

"Model Gateway" means the service layer that forwards requests from the Frontend to Providers and records a Request Envelope (metadata such as request ID, model ID, token counts, latency, and status).

"Request Envelope" means non-content request metadata retained for reliability, capacity, and SLO monitoring.

"Session Telemetry" means minimal first-party analytics collected on page load/navigation (e.g., timestamp, pseudonymous session ID, coarse locale, feature flags).

"Security Logs" means IP address and User-Agent records used for rate-limiting and abuse detection.

"Chat Object" means prompt(s), tool calls (if any), and model output(s) associated with a session or account within the Open WebUI Instance.

"Contribution" means any data a user intentionally submits to the Flywheel (e.g., prompt/output pairs, tags, corrections), along with per-item License and AI Preference Signal selections.

"AI Preference Signal" means an AI-use preference value the contributor attaches to a Contribution (e.g., IETF AI Preferences draft values and/or Creative Commons preference signals), intended to be conveyed downstream.

"License" means the Creative Commons license selected by the contributor for a Contribution (supported in the MVP: CC0-1.0, CC-BY-4.0, CC-BY-SA-4.0).

"License Bucket(s)" means the partitioning of Contributions into separate release artifacts by License (e.g., v1.0-cc0, v1.0-cc-by, v1.0-cc-by-sa).

"Waiting Room" means the Flywheel's gated staging directory (e.g., _waiting_room/ in a Hugging Face repository) where Contributions are first written prior to validation and release.

"Release" means a published version of the dataset (and associated notes/checksums) assembled from validated Contributions, partitioned by License.

"Gated Repository" means the Hugging Face dataset repository that requires an access request and acceptance of dataset-specific terms.

"Static Site" means the public site that hosts leaderboards and provides full dataset access, with anti-scraping controls.

"Anonymized Contributor ID" or "Pseudonym" means a non-identifying handle published with a Contribution when a contributor elects anonymity or a pseudonym instead of a Hugging Face username.

"Personal Data" means information that identifies or can reasonably be linked to an individual; "Sensitive Personal Data" means Personal Data that is sensitive by law or policy (e.g., health, financial, precise location, government identifiers).

"We/Us/Our" means [ENTITY NAME], the operator of the Frontend, the Open WebUI Instance, and the Flywheel.

"You/Your" means the individual using the Frontend and/or contributing to the Flywheel, or the entity on whose behalf the individual acts.

7.2 What data is produced & when

7.2.1 Open WebUI (no account required, but optional and recommended)

Your OpenWebUI account and all associated data are stored and managed entirely by your OpenWebUI instance. In the case of the public AI MVP, both the flyhweel and frontend will be managed by the same organization, but in theory you could use the flywheel using an entirely separate OWUI instance (a community instance, your own, etc.)

The Flywheel App has no access to or control over the data stored within OpenWebUI. This includes:

- **Your OpenWebUI Account:** Your login credentials, email, and user settings.
- **Your Full Chat History:** All conversations you have within OpenWebUI are stored on that platform's server.
- **Server Logs & Analytics:** Any logs or analytics generated by the OpenWebUI platform itself.

To understand how OpenWebUI collects, uses, and retains your data, you must consult the specific Terms of Service and Privacy Policy provided by the administrator of your OpenWebUI instance.

This data includes:

- Session telemetry (minimal) #todo: Double check telemetry and provide a sample paylod to show interested users;
  - Produced when: Page load / navigation
  - Includes: Timestamp, pseudonymous session ID, coarse locale, feature flags
  - Stored in: First-party analytics
  - Access: Eng/Analytics (aggregated)
  - Default retention: Raw 30 days; aggregates 13 months
- IP & User-Agent (security) #todo double check this
  - Produced when: Each request
  - Includes: IP, User-Agent for rate-limit/abuse detection
  - Stored in: Security log store
  - Access: SRE/Security
  - Default retention: 7 days, then delete/aggregate
- Query content
  - Produced when: On submit
  - Includes: Prompt + output
  - Stored in: interface database
  - Access: server admin only
  - Default retention: user can delete at any time; deleted along after 30 days of user inactivity. #todo, discuss this
- Error logs (sanitized) #todo double check this, provide example? least important to provide an example here.
  - Produced when: On error
  - Includes: Stack trace, request ID (no prompt/output bodies)
  - Stored in: Log store
  - Access: Eng (least privilege)
  - Default retention: 30 days
- Profile and settings #todo double check this
  - Produced when: Sign-up
  - Includes: Email/OAuth ID, display name
  - Stored in: User DB
  - Access: Support/Eng
  - Retention: Kept until the user deletes it. If the account is inactive for 30 days, we delete the account and associated records.

7.2.2 Data Sent from OpenWebUI to Model Gateway (to providers)

- Request envelope
  - Produced when: Each request
  - Includes: Request ID, model ID, token counts, latency, status
  - Stored in: Metrics DB
  - Access: Eng/SRE
  - Retention: 90 days (aggregates 13 months)

  Provider transparency: We do not guarantee provider behavior. We clearly display the exact payload we forward (headers + body summary) and link to the provider's own terms and policies where available. Users should review provider terms before use.

7.2.3 Data Sent From OpenWebUI to the Flywheel

Within the OpenWebUI interface, you may have the option to explicitly **opt-in** and share a specific chat with this public dataset project. When you choose to do this, OpenWebUI sends a copy of that single chat to our `/api/contributions/ingest` endpoint.

Once that data is received by our application, it is handled according to the policies described in this document. The data packet we receive includes:

- The chat content and metadata you selected.
- Your OpenWebUI username (for display purposes if you choose "public").
- A unique, non-identifying `provider_user_id` (e.g., `user-123`). We use this to create a `contributor_hash` to group your anonymous contributions, but we never store the raw ID itself in the final dataset.

This looks like:

- Contribution payload
  - Produced when: On explicit opt-in
  - Includes: Prompt, model output, optional tags
  - Stored in: Gated staging (effectively public) → license-bucketed release (also effectively public)
  - Access: Public
  - Retention: Indefinite
- HuggingFace username OR Anonymized contributor ID
  - Produced when: On ingest

  - Includes: user provides their huggingface username or selects a pseudonym (can leave blank)
  - Stored in: Dataset metadata
  - Access: Public
  - Retention: Indefinite

- Release artifacts

  - Produced when: On release
  - Includes: JSONL/TSV, checksums, notes
  - Stored in: HF (gated) and Static Site (public)
  - Access: Public (per channel)
  - Retention: Indefinite

When you sign in directly to this Contribution App's web interface, you are authenticated through your Hugging Face account using the standard OAuth protocol.

- **What we use:** We request the `openid`, `profile`, and `email` scopes from Hugging Face. The application uses your **public Hugging Face username** (also called a "handle") to identify you. This username is stored in a secure, encrypted session cookie in your browser.
- **What we don't store:** Your email address and Hugging Face password are never stored or logged by our application. The session cookie is deleted when you sign out or it expires.
- **Who can access it:** Only the application's backend can read your session cookie to verify you are logged in when you submit a contribution.
- **Retention:** This data is ephemeral and only lasts for the duration of your active session.

7.3 Server & API Data

To ensure security, stability, and prevent abuse of the Contribution App, we handle technical data related to your requests.

7.3.1 Rate Limiting

To prevent spam and ensure the service is available for everyone, we limit the number of requests a single user can make.

- **What we use:** The `middleware.ts` file shows that we use your **IP address** to enforce a rate limit (e.g., 100 requests per hour). For ingestions from OpenWebUI, we rate-limit based on their API key.

- **Who can access it:** Your IP address is sent to **Upstash Redis**, our rate-limiting service provider.
- **Retention:** Upstash retains a record of your IP address for the duration of the rate-limiting window, which is **1 hour**.

7.3.2 Server Logs

Like most web applications, our hosting platform (**Vercel**) automatically generates server logs for every request.

- **What they contain:** These logs may include your **IP address**, user-agent string (browser information), the requested URL, response status code, and other standard request headers. If an error occurs, the log might contain details about the error to help us debug.
- **Who can access it:** Project Maintainers can access these logs through the Vercel dashboard for debugging and monitoring purposes.
- **Retention:** Log retention is determined by Vercel's platform policies, which is typically between 1 and 30 days.

7.4 Event timeline (how data flows)

1. User prompts a model → payload sent to model provider; query and response are saved if user made an optional OpenWebUI account.
2. If user wishes to select a chat to share via opt-in, they can do so. They never have to. Licensing and preference signals are set at opt-in time.
3. Opt-in chat goes to "waiting room"
4. Processing script collects items from waiting room, applies some checks (PII, content moderation), moves them into release buckets.
5. A separate processing script takes data from the gated HF repo and builds the static site with leaderboard and data dump
6. Post-release: approved removals are honored in future releases and our mirrors; prior downloads may persist.

7.5 More on the flywheel

This is the core data lifecycle for every chat you contribute. Your submission goes through several automated stages before it becomes a permanent part of the public dataset.

7.5.1 Phase 1: Submission

Whether you use the web form or contribute via OpenWebUI, you provide the data for a single conversation.

7.5.2 Phase 2: The Waiting Room (Temporary)

Immediately after submission, your contribution is packaged into a single JSON file and uploaded to a private `_waiting_room` directory in our Hugging Face dataset repository.

- **What's in the file:** This file contains the raw content and metadata from your submission. To uniquely and privately identify contributors, we generate a `contributor_hash`:
  - For web submissions, this is a SHA256 hash of your Hugging Face username combined with a secret salt: `sha256(salt + hf_username)`.
  - For OWUI submissions, this is a hash of the provider and your provider user ID: `sha256(salt + "openwebui:user-123")`.
- **Who can access it:** Only Project Maintainers with access to the repository can see these files.
- **Retention:** These files are **temporary**. They exist only until the processing script runs, typically within a few minutes or hours, after which they are deleted.

7.5.3 Phase 3: Processing & PII Redaction

A script (`process_pending.ts`) periodically runs to process every file in the waiting room. Its primary job is to validate the data and **automatically redact Personally Identifiable Information (PII)** from the chat content. Our redaction script looks for and replaces the following patterns:

- Email addresses
- IP addresses (IPv4 & IPv6)
- Social Security Numbers (SSN)
- IBAN bank account numbers
- Ethereum and Bitcoin wallet addresses
- Phone numbers
- Credit card numbers (verified with Luhn check)
- Simple name patterns (e.g., "my name is John Doe")

If more than 5 potential PII hits are found, the file is moved to quarantine for safety.

7.5.4 Phase 4: The Quarantine Zone

If a submission fails processing, it's moved to a private `_quarantined` directory.

- **Why it's quarantined:** A file is quarantined if it is malformed (e.g., invalid JSON), fails our content validation rules, or has too many PII hits detected by the redaction script.
- **Who can access it:** Only Project Maintainers can access these files to diagnose processing errors.
- **Retention:** Quarantined files are kept **indefinitely** for manual review.

7.5.5 Phase 5: The Final Public Dataset (Permanent)

After successful processing and PII redaction, your contribution is appended to a monthly JSON Lines file (e.g., `data/2025-08.jsonl`).

- **What it is:** This is the final, permanent, and clean version of your contribution. The content has been redacted, and the metadata is structured according to our public schema.
- **Who can access it:** This dataset is **public**. Anyone in the world can view, download, and use it according to the license you chose for your contribution.
- **Retention:** Contributions to this dataset are **perpetual and irrevocable**, as stated in the Terms of Service you agree to upon submission.

7.6 Licensing & Preference Signals (Beta)

For each contribution, the user selects:

- A Creative Commons license: CC0-1.0, CC-BY-4.0, or CC-BY-SA-4.0.
- An AI preference signal: an IETF AI preference (draft) value and/or CC preference signal ("IETF AI Pref combo").

How we use these:

- We record `license` and `ai_pref` per record.
- Records are partitioned by license into separate release buckets.
- Downstream users must comply with the license; we publish compatibility notes (e.g., ShareAlike).
- Users may set account defaults; each submission can override.

Binding effect: Once a record is included in a release, that license applies to that copy.

7.7 Example Retention schedule

- Web edge
  - Data: IP + UA
  - Purpose: Abuse prevention
  - Retention: 7 days raw → delete/aggregate
  - Deletion path: Automatic purge
- Web UI
  - Data: Minimal telemetry
  - Purpose: Reliability metrics
  - Retention: 7 days raw → delete/aggregate
  - Deletion path: Automatic purge
- Gateway
  - Data: Request envelopes
  - Purpose: Capacity/SLOs
  - Retention: 90 days raw; 13 months aggregates
  - Deletion path: Automatic purge
- Open WebUI Accounts
  - Data: Profile and preferences
  - Purpose: Auth/consent
  - Retention: Kept until user deletes; 30-day inactivity → delete
  - Deletion path: Self-service delete / auto-purge
- Flywheel staging bucket (the "waiting room dir", on HF)
  - Data: Pending contributions
  - Purpose: Moderation/de-duplication
  - Retention: Indefinite (moved to organized buckets in same repo)
  - Deletion path: Manual removal on request
- HF (gated)
  - Data: Released records (by license)
  - Purpose: Research access
  - Retention: Indefinite
  - Deletion path: Exclude from future versions; coordinate HF deletion where possible
- Static Site (public)

- Data: Leaderboards and full dataset access
- Purpose: Benchmarking/transparency
- Retention: Indefinite
- Deletion path: Update/remove in future releases; past d

7.8 Distribution & access control

7.8.1 Hugging Face (gated)

- Separate artifacts and checksums per license bucket and version (e.g., `v1.0-cc0`, `v1.0-cc-by`, `v1.0-cc-by-sa`).
- Access requires a request with acceptance of dataset-specific Terms of Use (no re-identification; honor license and AI preferences).
- Deletion requests: Where possible, we use Hugging Face–native workflows (e.g., issue/repo requests, maintainers' takedowns) to process deletions and exclude items from future versions.

7.8.2 Flywheel Static Site (public)

- Hosts leaderboards and provides full dataset access for building benchmarks.
- Cloudflare anti-scraping posture: Bot Management/WAF rules, rate limits, Turnstile/JS challenges on download routes, tokenized short-lived URLs, robots/meta controls, pagination/throttling, and anomaly monitoring.
- Reality check: Anti-scraping reduces but cannot prevent copying of public data. We limit risk via license partitioning, logged access flows, and clear terms.

7.9 Contributor rights & controls

- Opt-in only for contributions to the flywheel.
- Per-item license and AI preference (beta).
- Withdrawals: Submit a verified request to exclude your items from future releases. We'll also file/route requests through Hugging Face where applicable.
- Account deletion & inactivity: If you keep an Open WebUI account, we keep your account data until you delete it; if you are inactive for 30 days, we delete the account and associated records. Released dataset copies remain under their chosen licenses.
- Sensitive content: Do not submit personal/sensitive data; automated filters and moderation apply.

7.10 9) Roadmap

- HF-auth contributions: Let contributors submit directly via their Hugging Face accounts to preserve reputation and contributor stats.
- AI preference UX: First-class UI for selecting license + IETF AI Pref combo, plus validator and compatibility helper.
- Provider term links directory: Central index of provider policies; per-model tooltips in the WebUI.

7.11 Contacts

#todo

- Data removal / privacy: {{PRIVACY_EMAIL}}
- Security: {{SECURITY_EMAIL}}
- Research & benchmarks: {{RESEARCH_EMAIL}}

Part III

Appendices

8 Appendix 1: LLM Data Schemas

Here, we describe many variants of LLM data. This will be relevant for when we extend the flywheel to include more types of data, and especially shift towards promoting the sharing (via opt-in flywheels, but also via new market mechanisms) of richer "content data".

- **Open Web / Crawls**
  - **WARC/WAT/WET**
    - *WARC* (container for HTTP request/response records) — spec & overview: IIPC WARC 1.1; Library of Congress format note. (IIPC Community Resources, The Library of Congress)
    - *WAT* (JSON metadata extracted from WARC) and *WET* (plain text extracted from HTML) — Common Crawl guides. (Common Crawl, Common Crawl)
  - **C4 (Colossal Clean Crawled Corpus)** — TFDS catalog & generator code. Fields are essentially clean text segments with basic metadata. (TensorFlow, GitHub)
  - **The Pile** (22-source, mixed corpus) — paper & HTML view. (arXiv, ar5iv)
- **Encyclopedic / Books**
  - **Wikipedia XML dumps** (page/revision XML; SQL tables for links) — MediaWiki dump format; Wikipedia database download. (Meta, Wikipedia)
  - **Project Gutenberg**
    - *Books*: plain text/HTML master formats; ePub/MOBI derived. (Project Gutenberg)
    - *Catalog schema*: daily RDF/XML (also CSV) for metadata; offline catalogs. (Project Gutenberg)
- **Scientific / Legal**
  - **arXiv** (Atom/OAI-PMH metadata; bulk & API) — OAI-PMH + API docs; bulk metadata page. (info.arxiv.org, info.arxiv.org, info.arxiv.org)
  - **JATS XML** (journal article tag suite) — NISO standards; NLM JATS site. (niso.org, jats.nlm.nih.gov)

- **Code**
  - **BigCode — The Stack / The Stack v2** (source files + license/provenance metadata; dedup variants) — HF datasets, project docs, arXiv overview. (Hugging Face, Hugging Face, BigCode, arXiv)
- **Forums / Q&A / Social**
  - **Stack Exchange dumps** (XML: Posts, Users, Comments, Votes, etc.) — SE Meta/docs & Data Explorer. (Meta Stack Exchange, data.stackexchange.com)
  - **Reddit**
    - *API JSON* schema — official API docs & help. (Reddit, Reddit Help)
    - *Pushshift* (historical dumps; research dataset) — site & paper. (pushshift.io, arXiv)
- **Instruction / Conversations (Post-training SFT)**
  - **OpenAI-style chat schema** (role-tagged: `system|user|assistant`, plus tool calls) — API reference. (OpenAI Platform)
  - **Alpaca** (JSON prompts/instructions/outputs) — Stanford post & repo; cleaned community set. (crfm.stanford.edu, GitHub, GitHub)
  - **Databricks Dolly-15k** (human-written instruction/response pairs) — repo. (GitHub)
  - **OpenAssistant OASST1** (message-tree conversations with roles) — HF dataset card. (Hugging Face)
- **Preference / Feedback (RLHF & DPO)**
  - **HH-RLHF** (Anthropic helpful/harmless, JSONL pairs: `chosen` vs `rejected`) — dataset repo readme. (GitHub)
  - **DPO format** (prompt + preferred vs dispreferred response) — DPO paper. (arXiv)
- **Multimodal (for VLMs/ASR)**
  - **LAION-5B / Re-LAION-5B** (image–text pairs with CLIP scores; links) — LAION posts. (laion.ai, laion.ai)
  - **Whisper** (weakly-supervised ASR; audio → text pairs) — paper & blog. (arXiv, OpenAI)
  - **HowTo100M** (YouTube instructional video clips + narrations) — project page & paper. (di.ens.fr, arXiv)
- **Math-reasoning (often for post-training/eval)**

- **GSM8K** (grade-school word problems; JSON) — repo & HF dataset card. (GitHub, Hugging Face)
- **MATH** (competition problems with step-by-step solutions) — paper & HF. (arXiv, Hugging Face)

- **Common storage containers**
  - **JSON Lines / NDJSON** — jsonlines.org; ndjson spec. (jsonlines.org, GitHub)
  - **TFRecord** — TensorFlow tutorial. (TensorFlow)
  - **Apache Parquet** — project site. (Apache Parquet)

#todo check all refs

9 Appendix 2 — Preference Signals for AI Data Use (CC signals + IETF AI Preferences)

#todo: improve the references here to specific lines of IETF draft and the CC Preference Signals FAQ

- **What CC signals are** A Creative Commons framework for *reciprocal* AI reuse: content stewards can allow specific machine uses if certain conditions are met (e.g., credit, contributions, openness). Overview & implementation notes. (homepage, implementation)
- **Four proposed CC signals (v0.1)**
  - **Credit (cc-cr)** — cite the dataset/collection; RAG-style outputs should link back when feasible.
  - **Credit + Direct Contribution (cc-cr-dc)** — proportional financial/in-kind support.
  - **Credit + Ecosystem Contribution (cc-cr-ec)** — contribute to broader commons.
  - **Credit + Open (cc-cr-op)** — release model/code/data to keep the chain open. Source (draft repo & posts). (GitHub, Creative Commons)
- **IETF AI Preferences (aipref) — the transport & vocabulary**
  - **Vocabulary**: a machine-readable set of *categories* (e.g., `ai-use`, `train-genai`) and *preferences* (`y` = grant, `n` = deny) with **exceptions**. Drafts. (datatracker.ietf.org, IETF, IETF AI Preferences Working Group)
  - **Attachment**: how to convey these preferences via **HTTP `Content-Usage` header** and **robots.txt** extensions. Drafts. (datatracker.ietf.org, IETF)
  - **Structured Fields**: uses RFC-standardized HTTP structured field values. (datatracker.ietf.org, datatracker.ietf.org, rfc-editor.org)
  - **Robots Exclusion Protocol** baseline. (datatracker.ietf.org, rfc-editor.org)
- **Putting them together (content-usage expression)**
  - Shape:

    ```
    <category>=<y|n>;exceptions=<cc-signal>
    ```

Example in **robots.txt** (allow everything, but *AI use denied unless Credit*):

```
User-Agent: *
Content-Usage: ai-use=n;exceptions=cc-cr
Allow: /
```

Example **HTTP header** (deny *gen-AI training* unless *Credit + Ecosystem*):

```
Content-Usage: train-genai=n;exceptions=cc-cr-ec
```

(Syntax and examples from CC & IETF drafts.) (Creative Commons, IETF)

- **Operational notes (for this repo's flywheel)**
  - **Per-record fields** to store: `license` (CC0/CC-BY/CC-BY-SA) and `ai_pref` (IETF aipref value + optional CC signal), plus optional `attribution` handle. (Aligns with CC write-ups & IETF drafts.) (Creative Commons, datatracker.ietf.org)
  - **Placement**:
    - *Location-based* signals via **robots.txt** for site/paths. (datatracker.ietf.org)
    - *Unit-based* signals via **HTTP** `Content-Usage` on dataset files and API responses. (datatracker.ietf.org)
  - **Interoperability expectations**: signals are normative *preferences*; adherence relies on ecosystem norms (similar to robots.txt & CC license culture). (Creative Commons)
- **Context & momentum**
  - CC's 2025 launch posts; IETF WG activity updates (e.g., IPTC note). (Creative Commons, Creative Commons, IPTC)

#todo check all refs

10 Appendix 3: Example Legal Terms

Modeled after Mozilla Common Voice terms. #todo: closer comparison.

10.1 Opt-in Data Flywheel — Legal Terms (Draft)

Effective: [DATE]

Through the Opt-in Data Flywheel, you may contribute chats, corrections, and related materials to build openly accessible evaluation sets and datasets.

You may participate only if you agree to these Opt-in Data Flywheel Legal Terms (the "Terms"). 1. Eligibility

The Flywheel is open to individuals who are the age of majority in their jurisdiction, or to younger participants with verified parental/guardian consent and supervision. You must also comply with Our Community Guidelines/Acceptable Use Policy ([LINK]). 2. Your Contributions; Licensing; AI Preferences

2.1 Opt-in Only. Submitting to the Flywheel is purely voluntary and separate from using the Open WebUI Instance.

2.2 License Grant (per item). For each Contribution, you select one License (CC0-1.0, CC-BY-4.0, or CC-BY-SA-4.0). You grant Us a non-exclusive, worldwide right to publish, reproduce, modify (solely for formatting, moderation, and aggregation), distribute, and sublicense the Contribution under the selected License. Once included in a Release, that License applies to that copy of the Contribution.

2.3 AI Preference Signals. If you attach an AI Preference Signal, We will transmit and display it with the Contribution and document how Our systems interpret such signals. We cannot guarantee that downstream users or Providers will honor such signals.

2.4 Assurances. You represent and warrant that (a) you have the necessary rights to your Contributions; (b) your Contributions do not infringe third-party rights; (c) you will not include Sensitive Personal Data; and (d) you will comply with Our Acceptable Use Policy. 3. Accounts; Attribution; Pseudonymity

3.1 Auth. Contributions require authentication (e.g., Hugging Face OAuth). 3.2 Attribution. You may choose to publish under your Hugging Face username, under a pseudonym, or as

"anonymous." 3.3 Leaderboards. We may publish contribution metrics (counts, languages, tags) with your chosen public handle. We will not publish your email address. 4. Processing; Waiting Room; Release

4.1 Waiting Room. Submissions write to a staging directory. 4.2 Validation. We may run automated and human review for formatting, de-duplication, PII/safety checks, and License/AI-preference validation. 4.3 Release. Validated items are appended to License Buckets (e.g., vYYYY-MM) and published to a Gated Repository and mirrored to the Static Site. 5. Distribution; Access Control

5.1 Gated Repository. Access requires acceptance of dataset-specific terms (e.g., no re-identification; respect License and AI preferences). 5.2 Static Site. Public access includes anti-scraping measures (WAF/bot management, rate limits, tokenized URLs). Copying cannot be fully prevented; rely on License controls for downstream obligations. 6. Deletions & Takedowns

6.1 Future-Only Removal. Upon verified request, We will exclude the identified Contribution(s) from future Releases and update mirrors where feasible. Past Releases and third-party copies may persist. 6.2 Hugging Face Workflows. Where possible, We will route or honor takedowns via the Hugging Face repository's native workflows. 7. Provider Transparency (No Guarantees)

We forward prompts to third-party Providers. We display a payload transparency panel and link to Provider terms when available. We do not control Provider retention, training, or other uses of data once sent to them. 8. Privacy; Retention

Retention and access for telemetry, envelopes, accounts, staging, Releases, and the Static Site are governed by the Data Retention & Contribution Policy (Section 3). That Policy is incorporated by reference. 9. Communications

By creating an account or requesting repository access, you may receive administrative emails (e.g., access decisions, policy updates). 10. Disclaimers; Limitation of Liability; Indemnity

THE FLYWHEEL AND RELEASES ARE PROVIDED "AS IS." TO THE MAXIMUM EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES (INCLUDING MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT). WE WILL NOT BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, OR PUNITIVE DAMAGES. Our aggregate liability under these Terms will not exceed USD $500 (or the maximum permitted by law if lower). You agree to indemnify Us for third-party claims arising from your Contributions or breach of these Terms. 11. Updates

We may update these Terms by posting a new effective date. Continued use after the effective date constitutes acceptance. 12. Termination

We may suspend or terminate access at any time. Contributions included in prior Releases remain available under their Licenses. 13. Governing Law; Venue

These Terms are governed by the laws of [LAW & VENUE], without regard to conflict-of-laws rules. Exclusive venue lies in the courts of [VENUE].

10.2 Frontend Instance

Open WebUI Instance — Terms of Use (Draft)

Effective: [DATE]

These Terms govern your use of Our hosted Open WebUI Instance at {{app_link}} (or successor URLs). 1. Eligibility; Community Rules

The service is available to individuals who are the age of majority in their jurisdiction, or younger participants with verified parental/guardian consent and supervision. You must follow Our Community Guidelines/Acceptable Use Policy ([LINK]). 2. Accounts; Content

2.1 Accounts Optional. You may use OWUI without an account; certain features (history, settings, opt-in share flows) require an account. 2.2 Your Content. Prompts and outputs in your account are stored as Chat Objects to provide history and UX features. They are not used for training or evaluation by Us unless you explicitly opt in via the Flywheel. 2.3 Feedback Data. Thumbs, flags, and similar signals may be stored to improve product reliability and moderation and are handled per Section 3 (Retention Policy). 3. Provider Transparency

OWUI forwards your prompts to third-party Providers. We display a payload transparency panel and, where available, links to Provider terms. We do not control Provider retention, training, or other uses of your data. 4. Privacy; Retention; Security

Retention, deletion, and access controls for telemetry, Security Logs, Request Envelopes, account data, and error logs are governed by the Data Retention & Contribution Policy (Section 3), incorporated here by reference. 5. Sharing to the Flywheel

Sharing to the Flywheel is separate and requires explicit opt-in with per-item License and AI Preference Signal selections. See the Flywheel Terms. 6. Acceptable Use

You agree not to: (a) upload Sensitive Personal Data; (b) violate laws or third-party rights; (c) attempt to reverse engineer or abuse rate limits; (d) circumvent access controls; or (e) interfere with service integrity. 7. Communications

If you create an account, We may send administrative emails (e.g., login links, security alerts, policy updates). 8. Disclaimers; Limitation of Liability

OWUI IS PROVIDED "AS IS." TO THE MAXIMUM EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES (INCLUDING MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT). WE WILL NOT BE LIABLE FOR INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, OR PUNITIVE

DAMAGES. Our aggregate liability will not exceed USD $500 (or the maximum permitted by law if lower). 9. Updates; Termination

We may update these Terms by posting a new effective date. Continued use after the effective date constitutes acceptance. We may suspend or terminate accounts for any reason, including AUP violations or security risk. 10. Governing Law; Venue

These Terms are governed by the laws of [LAW & VENUE], without regard to conflict-of-laws rules. Exclusive venue lies in the courts of [VENUE].