The IQ/MQ Filter
Interface, Training, and Operational Design
A Companion Technical Outline to Paper III of the IQ/MQ Space Series

Lev Neymotin, PhD
Uncompressed Thinking

N5Digital, Inc.

Abstract
Papers I and II established the IQ/MQ framework by introducing the space in which public behavior may be located and by clarifying the meaning of its two principal axes. Paper III then advanced the next necessary claim: if the feed is already being filtered, the decisive question is not whether filtering exists, but who owns it and according to what principles it operates. The present companion document takes the next step. Its purpose is to translate the conceptual architecture of Paper III into structured design logic.

This document does not attempt to replace the main paper, nor does it yet attempt to become software engineering documentation. Its role is intermediate but essential. It defines a working system model for a user-owned adaptive mediation layer that operates between platform output and user intake. It introduces project-specific terminology intended to remain stable across later stages of prototype design and coding. It specifies the principal objects handled by the system, the user interface through which the filter is controlled, the training lifecycle through which it is calibrated, the governance logic through which drift is constrained, the mediation pipeline through which content is interpreted and surfaced, and the privacy architecture through which the user's learned profile remains under user control.

The basic design premise is that the filter must be both cognitively serious and operationally disciplined. It must judge posts in context rather than accounts alone. It must preserve intensity when intensity reflects reality, while reducing distortion and manipulative degradation. It must remain explainable without becoming cluttered. It must learn, but slowly. It must preserve diversity and challenge exposure. It must operate under low-latency conditions. And it must treat the user's mediation profile as owned property, not as a new source of platform extraction.
The sections that follow are therefore not a repetition of Paper III. They are the first structured attempt to describe how Paper III could be built.

1. Purpose and Working Vocabulary
The purpose of this companion document is to move the IQ/MQ Filter from conceptual argument to operational design. Paper III established why such a system is needed and what principles must govern it. The companion must now describe the system in a way that is precise enough to guide actual development while still general enough to precede code, interface mockups, and implementation-specific decisions.

For that reason, this document introduces a working vocabulary that should remain stable across later stages. **Feed Ingress** refers to the raw incoming stream of platform content before mediation. A **Content Object** is the normalized internal representation of any post-like item the system can process. A **Context Envelope** is the attached bundle of contextual signals surrounding that Content Object: author behavior pattern, thread state, topic cluster, corroboration status, event relevance, recency, and other signals needed for interpretation. A **Placement Vector** is the provisional mapping of the Content Object into IQ/MQ space together with its confidence level and auxiliary descriptors.

On the user side, the **Declared Profile** is the set of preferences chosen explicitly by the user. The **Adaptive Profile** is the slower learned layer that adjusts within the limits of the Declared Profile. The user selects a **Target Zone** in IQ/MQ space, together with a **Porosity Radius** that determines how tightly the filter concentrates around that zone. The system also maintains a **Challenge Budget**, meaning the controlled allowance for high-quality content that falls outside the user's preferred region but should still pass in the name of diversity and contact with reality.

The mechanism that prevents unstable learning is the **Drift Governor**. The **Live Path** refers to the real-time decision route used while the feed is being rendered. The **Background Path** refers to asynchronous computation that updates models, contexts, caches, and long-horizon user-state variables without burdening the moment of scrolling. Every mediated item exits the system with a **Surface Action**: it may be elevated, passed normally, attenuated, clustered, or deferred. Where explainability is exposed, the item may also carry a **Route Tag**, a compact indication of why it reached the user in the form it did.

These terms are not branding devices. They are operational handles. Their purpose is to give the project a common language that can later be translated into interface specification, state logic, stored objects, and code modules. In that sense, the present companion document serves as the design bridge between Paper III and a later **Build Specification**, where the first actual build of the app will be defined more concretely.

2. System Architecture Overview

At the highest level, the IQ/MQ Filter is a user-side mediation system. It does not function as a universal public rating service, nor as a platform-wide moderation engine. It functions on the path between platform output and user perception. Its task is to ingest content from one or more platform sources, convert that content into interpretable objects, compare those objects to the user's chosen and slowly learned preferences, and render a feed whose order, visibility, and emphasis are governed by user-owned rather than platform-owned logic.

The system can therefore be understood as having six major internal layers. First comes **Feed Ingress**, which retrieves or receives raw content. Second comes **Normalization**, which converts heterogeneous posts, replies, quoted items, and repost-like forms into standard Content Objects. Third comes **Context Assembly**, which attaches a Context Envelope to each Content Object. Fourth comes **Placement and Mediation**, in which the Content Object receives a Placement Vector and is evaluated against the user's profile state, challenge budget, and drift constraints. Fifth comes **Surface Rendering**, in which the item is assigned a Surface Action and displayed. Sixth comes **Profile Maintenance**, in which explicit and implicit user feedback are interpreted and slowly incorporated into the Adaptive Profile.

One crucial point must be stated early: although some ordering is unavoidable in any feed, the IQ/MQ Filter should not be conceptualized merely as a ranking engine. Ranking is only one downstream

consequence of mediation. The system is broader. It decides not only what rises or falls, but what clusters together, what is delayed until more context is available, what enters through challenge allowance, what remains visible but deemphasized, and what receives an explanatory marker. The filter is therefore a mediation architecture, not simply a sorting function.

A second crucial point is that the system should be modular. Feed Ingress, Context Assembly, Placement, Profile Maintenance, and Surface Rendering should be separable modules, precisely so that later prototypes can vary one without destabilizing the whole. The companion does not yet specify classes, services, or storage engines, but it does insist on modular boundaries. That design choice will matter later when implementation begins.

3. Core Objects, Context Units, and Placement

The system's most fundamental operational decision concerns what exactly it treats as the object of judgment. Paper III already established that a filter confined to account-level judgment would be too coarse. The companion therefore formalizes the primary unit as the **Content Object**. A Content Object is not merely the literal text of a post. It is the minimum standardized entity on which the filter can act. Depending on the platform, it may include text, quoted text, media references, repost relations, reply position, author identifier, timestamp, and linked conversation structure.

A Content Object does not stand alone. It carries a **Context Envelope**, because the meaning of content depends heavily on situation. The Context Envelope should include, at minimum, an **Author Pattern**, a **Thread State**, a **Topic Cluster**, and an **Event Frame**. The Author Pattern represents longer-horizon behavior associated with the author without collapsing the present judgment into a blanket account verdict. The Thread State captures whether the surrounding conversation is constructive, escalating, degraded, chaotic, or unusually informative. The Topic Cluster groups the item into a recurring thematic field. The Event Frame captures whether the content belongs to an unfolding real-world event in which urgency and intensity may be more legitimate than in ordinary discourse.

The system then produces a **Placement Vector**. This vector should contain at least four elements: an estimated IQ coordinate, an estimated MQ coordinate, a **Placement Confidence**, and a set of **Auxiliary Modifiers**. These modifiers include such things as evidence density, corroboration level, manipulative emotionality, degradation risk, novelty, repetition burden, and seriousness of real-world consequence. The Placement Vector should never be mistaken for a claim of perfect truth. It is a working operational estimate used for mediation, not a final metaphysical judgment about a person or a post.

This structure allows the system to distinguish between cases that would otherwise be flattened into the same category. A terse emergency post may carry high emotional force but still merit passage because its Event Frame and corroboration status are strong. A superficially polished post may appear cognitively elevated while carrying weak substance and heavy repetition. A degraded thread may contain one worth-surfacing contribution whose individual Content Object diverges from the thread's broader descent. Without object-context separation, these distinctions would be difficult to preserve.

4. User Interface Architecture

The user interface must make the filter governable without making the user feel like a systems operator. That requirement imposes a discipline on the design from the outset. The interface should not expose every internal variable, nor should it attempt to teach the user the whole machinery of mediation. Its purpose is narrower and more important: it must allow the user to set directional preference, observe how mediation is occurring, correct the system when necessary, and shift temporarily among broader or

narrower informational stances without destabilizing the underlying profile. Everything else should remain subordinate to those four functions.

For that reason, the interface should be organized around a persistent but compact **Control Deck**. The Control Deck is the visible command layer of the filter. It contains the user's **Target Zone** in IQ/MQ space, the **Porosity Radius** that determines how tightly the filter should concentrate around that zone, the **Challenge Budget** that determines how much high-quality out-of-zone content may still pass, and the current **Mode State**. The Mode State should support at least three working modes: **Comfort Mode**, in which the feed remains closely aligned to the Target Zone; **Challenge Mode**, in which the Challenge Budget is increased and the effective boundary around the preferred region is loosened; and **Survey Mode**, in which the user deliberately explores the surrounding informational field without treating that exploratory behavior as a durable profile shift. These mode changes should be treated as **Session Overrides** rather than permanent rewrites of user identity.

The primary visual control inside the Control Deck should remain spatial. A small but expressive **Quadrant Map** should allow the user to place or reposition a **Zone Cursor**, defining the current center of preference. Around that cursor sits the Porosity Radius, visually expressing how concentrated or permissive the current filter should be. This arrangement preserves one of the distinctive strengths of the project: the user is not asked to manage a bureaucratic dashboard of disconnected sliders, but to navigate a field whose geometry already integrates several higher-level preferences. The user should be able to move this control deliberately, but it should also be possible to collapse the Control Deck into a quieter summary state once the current session is underway.

The system should support two principal feed surfaces. The first is **Timeline Surface**, which preserves a familiar list-based reading experience and should serve as the default mode for ordinary use. The second is **Space Surface**, which re-presents the current feed spatially, allowing the user to browse clusters, concentrations, and outliers in IQ/MQ space rather than consuming them only as a scrolling strip. Timeline Surface is important for usability and adoption; Space Surface is important for the project's longer-term identity, because it makes visible that the feed is an informational terrain rather than a neutral sequence. Both surfaces should operate over the same mediated content state, so that the user is changing view, not switching into a different logical system.

Within Timeline Surface, each visible item should carry only a minimal mediation signal. This should take the form of a compact **Route Tag** or similarly restrained marker indicating the route by which the item entered the feed. A matched item may simply appear as standard. A challenge item may be marked discreetly. A reality-pass item may carry a different compact signal. Repeated similar items may appear as **Cluster Cards** rather than as repeated standalone interruptions. Items whose interpretation remains uncertain may be temporarily withheld from strong prominence and, when necessary, placed into a quieter **Deferred Layer** rather than forced immediately into the main flow. The purpose of these surface distinctions is not decorative complexity, but controlled visibility.

Inspection and correction must be built directly into the interface. Selecting an item should open an **Inspection Panel** that reveals a compact explanation of why the item was surfaced in its present form. The panel should show, at minimum, the item's route, a simplified indication of its inferred placement, the role of challenge or reality-pass logic if either was involved, and any major contextual factor that shaped the decision. From the same panel, the user should be able to issue **Override Actions**. These should include, at a minimum, actions equivalent to "more like this," "less like this," "misplaced," "important despite mismatch," and "not challenge." These actions are essential because they transform

the user from a passive recipient into a co-author of the mediation layer. They also provide the kind of explicit training input that later adaptation can trust more than ambiguous behavioral traces.

Finally, the interface should distinguish clearly between **Profile State** and **Session State**. Profile State refers to the user's durable Declared Profile and the slower Adaptive Profile built around it. Session State refers to temporary adjustments made for a given period of reading or exploration. A user should be able to enter Challenge Mode for an evening, widen the Porosity Radius for a particular topic, or browse the Space Surface out of curiosity without the system treating every such act as a durable change in identity. This distinction will later matter greatly in implementation, but it belongs first to the architecture of the interface. A filter that does not separate durable preference from temporary session behavior will either become unstable or become unresponsive. The interface therefore has to encode that distinction from the beginning.

5. Training Lifecycle

The training lifecycle must begin from a simple premise: the filter cannot infer a stable informational identity from raw behavior alone. If it tries to do so, it will confuse curiosity with endorsement, disturbance with attraction, and temporary mood with durable preference. Training must therefore be treated as a distinct lifecycle rather than as a side effect of ordinary usage. Its purpose is to establish a trustworthy starting profile, to protect that profile during early instability, and only then to permit bounded adaptation.

The first phase is **Cold Start**. At Cold Start, the system has no meaningful Adaptive Profile and only a provisional operating stance. It may begin with a moderate default Target Zone, moderate Porosity Radius, and moderate Challenge Budget, but these are placeholders only. The purpose of this phase is not to serve the user in finished form. It is to move quickly into explicit calibration. For that reason, the app should avoid pretending it already understands the user. Instead, it should openly indicate that the filter is entering a guided setup phase.

That guided setup phase is **Calibration**. During Calibration, the system should present a bounded **Calibration Set** made up of representative Content Objects drawn from different regions of IQ/MQ space, different tonal intensities, and different contextual forms. Some items should be straightforward. Others should be deliberately ambiguous, precisely because the system needs to see how the user distinguishes intensity from distortion, challenge from noise, and seriousness from manipulation. The user should not be asked for abstract theory. The user should be asked for simple but meaningful reactions: whether more or less of that kind of content is desired, whether the system's suggested placement appears plausible, whether the item should count as challenge-worthy, and whether it should still pass despite tension with the preferred zone. These explicit responses should be stored as a **Calibration Ledger**, because they form the first trusted layer of training evidence.

From Calibration, the system derives the initial **Declared Profile**. This profile contains the user's chosen Target Zone, Porosity Radius, Challenge Budget, and baseline Mode preference, together with any explicit stances on the treatment of urgent and well-grounded content. At this stage, the Adaptive Profile should still remain weak. The system should not yet assume that it has learned the user in any rich sense. It has only established a principled starting position.

The next phase is **Stabilization**. Stabilization is the period in which the user begins using the mediated feed while the system begins learning cautiously. During this phase, the **Drift Governor** should remain highly restrictive. Explicit user corrections should carry strong weight. Behavioral cues such as dwell

time, fast exits, revisits, saves, or route inspections may be recorded, but they should not yet produce strong movement in the Adaptive Profile. The aim of Stabilization is not fast personalization. It is to protect the user from having the filter rewritten by the noise of early usage. This phase is especially important because the first interactions with a new system often contain heightened curiosity, unusual exploration, and deliberate testing behavior. A system that adapts too quickly during Stabilization will misread experimentation as identity.

Only after that period should the system enter **Live Adaptation**. In Live Adaptation, the Adaptive Profile becomes meaningfully active, but still under bounded rates of change. Adaptation should proceed through a clear **Signal Hierarchy**. At the top sit explicit Override Actions and direct profile edits. Below them sit strong behavioral signals such as saves, purposeful revisits, or repeated opening of similar content despite attenuation. Below them sit weak signals such as lingering scroll pauses or incidental taps. The system may learn from all three levels, but it should learn in very different ways. Strong signals may alter local tendencies. Weak signals should modify confidence only gradually. No plausible interpretation of user experience justifies letting casual behavior rewrite the Target Zone rapidly.

The lifecycle must also include **Recalibration**, but recalibration should not be intrusive. The system should not pester the user with constant training prompts. Instead, it should trigger a recalibration opportunity under identifiable conditions: persistent correction patterns, widening disagreement between Declared Profile and Adaptive Profile, sustained low confidence in certain topic clusters, or direct user request. Recalibration may be brief. It need not recreate the original onboarding experience. Its purpose is to repair drift, refresh uncertain zones, and keep the filter aligned with the user's durable intentions rather than with accumulated noise.

The training lifecycle therefore has a specific constitutional form: explicit before implicit, slow before fast, bounded before free, and reviewable before invisible. This sequence is not merely a user-experience choice. It is one of the primary safeguards by which the IQ/MQ Filter avoids becoming another opaque engagement engine under a different name.

6. Adaptive Control and Drift Governance

Training and adaptation are not identical. Training establishes initial fit. Adaptation governs how that fit evolves. For this reason, the companion separates **Adaptive Control** from the broader training lifecycle. At the center of this section stands the **Drift Governor**. The Drift Governor is not one rule but a family of constraints. Its purpose is to prevent the Adaptive Profile from being captured by temporary emotional states, episodic obsession, crisis periods, or accidental overexposure to one style of content. It should therefore combine rate limits, long-memory weighting, hysteresis, and explicit preference priority.

The system should maintain a clear **Signal Hierarchy**. At the top are explicit user acts: direct placement corrections, route disagreement, manual elevation or attenuation, and durable preference adjustments. Below these come strong behavioral signals such as saves, revisits, or deliberate expansion of explanation views. Below these come weaker signals such as dwell time, partial scrolling pauses, and incidental clicks. The Adaptive Profile should respond strongly to the first category, modestly to the second, and cautiously to the third.

The system should also maintain a **Correction Ledger**, meaning a running memory of the user's explicit disputes with system interpretation. A system that repeatedly receives the same kind of correction should not merely tweak one item; it should adjust its inference tendencies. At the same time, the

Correction Ledger should itself be interpreted with time depth. Repeated corrections over months mean something different from a burst of corrections on one unusually charged day.

Diversity protection also belongs to drift governance. Without a floor beneath the Challenge Budget, the system could slowly collapse into overfit personalization. Therefore the Drift Governor should protect not only against instability, but also against excessive convergence. Some minimum level of high-quality challenge should remain structurally present unless the user explicitly turns it off.

7. Mediation Pipeline and Surface Actions

Operationally, the system proceeds through a mediation pipeline rather than a single scoring act. The first stage is Feed Ingress and normalization. The second stage is Context Assembly, where each Content Object receives its Context Envelope. The third stage is Placement, where the system computes the Placement Vector. The fourth stage is Profile Comparison, where the Placement Vector is evaluated against the Declared Profile, Adaptive Profile, Mode State, Challenge Budget, and Drift Governor. The fifth stage is Surface Action selection. The sixth is rendering and explanation.

The most important operational concept here is the **Surface Action**. The system should not rely on only two outcomes, pass or block. It needs a richer surface vocabulary. **Elevate** means the item is surfaced with above-baseline prominence because it strongly matches the Target Zone or because it is high-value challenge content. **Pass** means the item enters the feed without special treatment. **Attenuate** means the item remains available but receives reduced prominence because its relation to the user's active filter state is weak or degraded. **Cluster** means the item is grouped with similar content rather than granted repeated standalone visibility. **Defer** means the item is temporarily held or context-delayed because the system lacks sufficient confidence or expects better interpretation after further background analysis.

This richer pipeline helps the system avoid the false harshness of simple suppression. A user-side mediation layer gains much of its power not by deleting content, but by controlling prominence, repetition, and contextual visibility. The pipeline should therefore be understood as surface management under user-owned logic.

The pipeline must also incorporate **Challenge Injection** and **Reality Passage**. Challenge Injection allows out-of-zone content to enter according to the Challenge Budget when it is judged to be high-value disagreement. Reality Passage allows urgent, well-grounded content to bypass ordinary strictness when the Event Frame and corroboration signals warrant it. These are not exceptions added late. They are integral to keeping the system from becoming an elegant cocoon.

8. Explainability Architecture

The explainability architecture should be layered, because the user's need for explanation is variable. A main feed overloaded with textual justification would become unusable. But a system that acts invisibly would violate one of the core promises of Paper III.

The first layer is the **Surface Signal**. This is the minimal visible indication that mediation has occurred. It may take the form of a small marker, a compact color signal, a route icon, or a concise Route Tag. Its purpose is not to explain everything. Its purpose is to tell the user that the item has not simply appeared without mediation.

The second layer is the **Explanation View**. When the user requests more detail, the system should reveal a compact but meaningful account of the route by which the item arrived. That explanation might indicate that the item matched the Target Zone strongly, entered through challenge allowance, received reality-pass treatment due to urgency and corroboration, or was attenuated but still surfaced due to topic relevance. The Explanation View should also allow the user to disagree with the system's route or placement.

The third layer is the **Profile Reflection View**. This layer is not about one post but about the user's filter state over time. It shows how the Declared Profile and Adaptive Profile currently relate, whether the system has drifted modestly in one direction, what proportion of the feed is challenge content, and whether certain forms of correction have become recurrent. Without such a view, the user cannot fully own the filter. With it, the user can periodically inspect whether the system still reflects the intended informational environment.

Explainability, then, is not an ornamental feature. It is the public face of legitimacy within the app.

9. Runtime Latency and Background Computation
A user-owned filter succeeds only if it feels usable. No matter how elegant the design, a system that visibly drags during ordinary scrolling will fail in practice. This makes the distinction between **Live Path** and **Background Path** fundamental rather than optional.

The Live Path should contain only operations that are light enough to preserve smooth rendering. These include profile lookup, retrieval of cached Placement Vectors and Context Envelopes, final mediation comparison, Surface Action selection, and rendering. The Live Path should not attempt to recompute deep thread interpretation, author history synthesis, large-cluster updates, or long-horizon model revision on demand.

Those heavier functions belong to the Background Path. Background computation should continuously update author patterns, thread summaries, topic clusters, event frames, confidence adjustments, and profile-learning aggregates. It should also precompute likely future needs. If the system expects that a given topic cluster or thread will remain active, it should refresh those objects before the user requests them. This is not mere optimization. It is architectural necessity.

The system should also maintain a **Confidence Discipline**. When placement confidence is low, the Live Path should not become slow by trying to solve the uncertainty from scratch. Instead, it should choose from bounded behaviors: pass with caution, attenuate, or defer pending background reinforcement. This keeps latency controlled even when ambiguity rises.

In practical terms, the success of the filter may depend less on peak analytical sophistication than on the discipline with which the boundary between Live Path and Background Path is maintained.

10. Ownership, Privacy, and Portability
The IQ/MQ Filter claims user ownership. That claim is hollow unless the system's data model reflects it. The most sensitive object in the entire system is not the individual Content Object, but the user's accumulated profile state: the Declared Profile, Adaptive Profile, Correction Ledger, challenge preference pattern, route disagreements, and long-horizon adjustment history.

For that reason, the system should maintain a secure **Profile Vault**. The Profile Vault is the protected storage domain in which user-specific mediation state is kept. The default assumption should be local-first or strongly user-consented storage rather than silent surrender to a platform backend. Even if cloud synchronization is later introduced, it should remain clearly subordinate to user authorization and transparent data movement.

The system should also define a **Portability Package**. This is the exportable form of the user's mediation identity. A user who changes device, environment, or platform should not be forced to rebuild the filter from zero. At the same time, portability must not expose raw behavioral intimacy unnecessarily. Therefore the Portability Package should be designed to preserve structural preference state without naively exporting every low-level event trace.

Ownership also implies inspectability. The user should be able to see what kinds of data the filter is using, what categories of signals are weighted heavily, and whether the Adaptive Profile has meaningfully diverged from the Declared Profile. A privacy system that hides its own architecture behind abstraction will not support actual user control.

The guiding rule is simple: the learned profile is not platform fuel. It is part of the user's informational property.

11. Deployment Paths

The deployment question remains open by design, but the companion should narrow the possibilities into a manageable set of paths.

The first path is the **Standalone Client**. In this model the IQ/MQ Filter exists as an independent app or companion interface that ingests content through available platform access routes and renders its own mediated feed. This path aligns best with the philosophical claim of user ownership, because the mediation layer visibly belongs to the user rather than being hidden inside another tool.

The second path is the **Browser Mediation Layer**. In this model the filter operates as a browser-side extension or augmentation. This may be more feasible in some early contexts, but it is philosophically weaker and often technically more fragile, because the mediated environment remains nested inside the platform's own presentation.

The third path is **Partner Integration**. In this model a platform or cooperative client adopts the filter logic as part of a native or semi-native user option. This offers the deepest technical integration, but it also introduces the greatest governance risk, because the distinction between user-owned mediation and platform-owned mediation can begin to blur.

A fourth path is a **Hybrid Architecture**, where the system keeps user profile state and mediation logic in a user-controlled domain while allowing some cloud-assisted or service-assisted processing for performance or cross-device continuity. This path may ultimately prove the most realistic, but only if the ownership boundary is carefully preserved.

The companion does not resolve this question finally. It frames the decision space so that later design work can proceed with explicit tradeoffs rather than vague assumptions.

12. Failure Modes and Safeguards

Any serious system proposal must specify how it can fail. The IQ/MQ Filter can fail by becoming too rigid, too reactive, too opaque, too slow, too paternalistic, too easily gamed, or too privacy-invasive. These failure modes are not peripheral. They shape the architecture.

One failure mode is **Overfit Calm**: the system becomes so aligned with the user's comfort that it suppresses necessary difficulty. The safeguard is a protected Challenge Budget floor and Reality Passage logic. A second failure mode is **Adaptive Capture**: the system drifts sharply because temporary behavior is misread as durable preference. The safeguard is the Drift Governor and Signal Hierarchy. A third failure mode is **Cosmetic Gaming**: content creators learn to imitate the visible signs of high-IQ/high-MQ content while remaining substantively empty. The safeguard is contextual evaluation, historical consistency checks, and user correction loops.

A fourth failure mode is **Explanation Overload**: the interface becomes technically honest but unusable. The safeguard is layered explainability. A fifth failure mode is **Latency Collapse**: the system attempts to solve too much in the Live Path. The safeguard is strict architectural separation between Live Path and Background Path. A sixth failure mode is **Profile Extraction**: the very system intended to return control to the user becomes a new machine for observing and monetizing the user's deeper mediation pattern. The safeguard is the Profile Vault, local-first logic, explicit portability design, and governance transparency.

A seventh failure mode is **Moral Overreach**. If the system presents its own judgments as final rather than provisional and inspectable, it begins to reproduce the paternalism it claims to resist. The safeguard is explicit modesty in design: confidence levels, correction pathways, and visible distinction between mediation estimate and moral certainty.

13. Development Roadmap

The development roadmap should reflect the layered structure of the project. Paper III established the normative and civic argument for user-owned mediation. The present companion document defines the operational architecture of the filter. The next artifact should therefore not be another conceptual paper, but a **Build Specification** that translates this companion into concrete product structure: screens, user flows, state transitions, stored objects, version-one scope, and implementation priorities.

The first milestone is **Terminology Freeze**, meaning that the working vocabulary introduced in this companion becomes stable enough to carry into later documents. The second milestone is **Prototype Specification**, where the general architecture of this companion is translated into concrete user flows, screens, states, and interaction models. That document will move closer to product design than the present companion, but should still remain above code.

The third milestone is **Simulation and Data Modeling**. At that stage, the project should test whether Content Objects, Context Envelopes, Placement Vectors, Surface Actions, and profile states behave coherently under synthetic or controlled datasets. The goal is not full public deployment but operational sanity checking. The fourth milestone is **Interface Prototyping**, where the Quadrant Control, explanation layers, correction flows, and Timeline Mode versus Space Mode are explored in mock or semi-interactive form.

Only after those steps should the project enter **Implementation Prototyping**, where actual modules are built and tested: Feed Ingress handlers, profile storage, mediation pipeline components, route tagging,

and drift governance logic. That stage should remain intentionally narrow at first, because premature breadth is one of the fastest ways to lose architectural discipline.

The final roadmap stages concern evaluation. The correct success measures are not engagement maximization or time-on-platform, but perceived signal quality, retained exposure to meaningful difference, trust in explainability, stability of profile behavior, and low-friction usability. If those are not measurably improved, the project will have failed even if its algorithmic sophistication is impressive.

The companion therefore ends not with code, but with design maturity. Its function is to make the next artifact possible: a Build Specification in which the first actual build of the IQ/MQ Filter can be defined in product terms precise enough to support disciplined coding.