Introduction to Designing Notifications: Timing, Priority, and Control

- Poor design costs trust: 67% want ≤1 push/day, 6+/week raises uninstall risk 3.4x
- Three pillars: when to notify (timing), what matters (priority), who controls (user)
- Failure in one pillar breaks the whole notification system
- Learners will evaluate appropriateness, design tiered priority, build respectful preference systems

The Science of Interruption: Cognitive Load and Resumption Lag

- Poorly timed notifications increase cognitive load and prolong task re-engagement
- Interruptions at high mental workload moments cause the greatest resumption costs
- Longer interruptions impose heavier cognitive burdens than short interruptions
- Contextual cues (e.g., placeholders) can reduce resumption lag and mitigate errors

Aligning Delivery with the User's Day: Circadian Rhythms and Context Sensing

- Map daily state: circadian rhythms, social schedules, and focus sessions predict notification receptivity
- Differentiate urgent alerts (e.g., fraud lock) from deferrable content suited for digests
- Respect Focus modes, DND, time zones, and behavioral signals like deep work blocks
- Block interruptions at high-workload moments — resumption costs spike during intense subtask execution

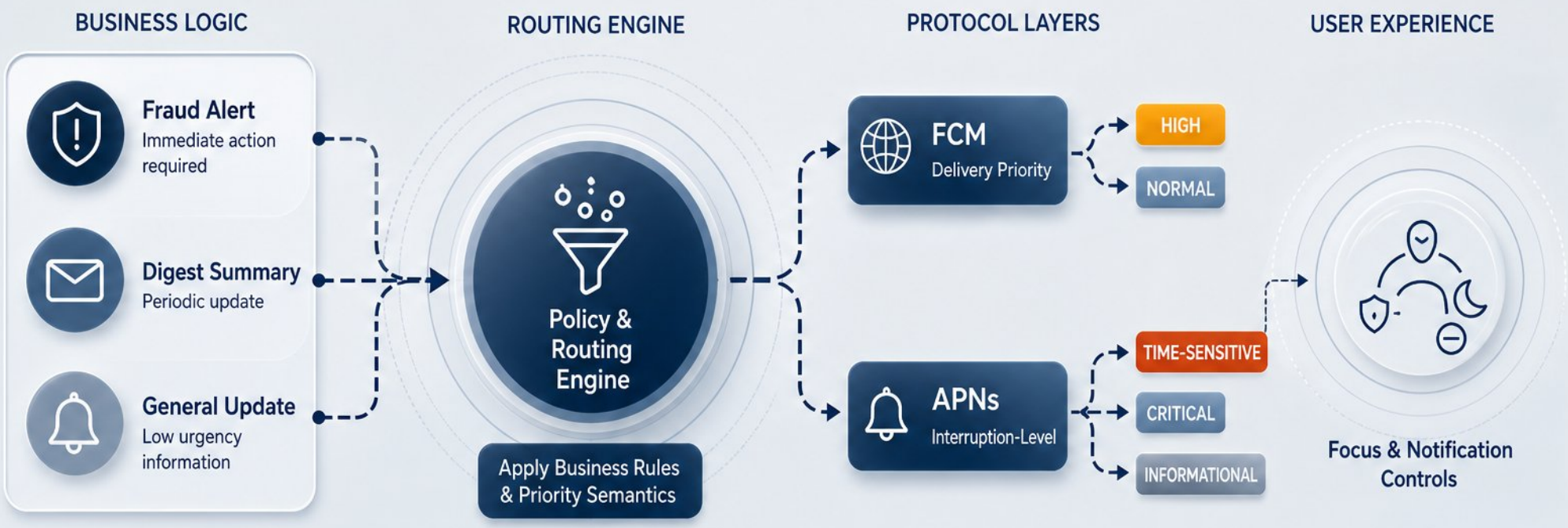
Timing Mechanisms: From Send-Time Optimization to Silent Windows

- Absolute vs. dynamic timing: Scheduled digests (e.g., daily 8 AM) vs. behavior-triggered and event-triggered notifications
- Machine-learning send-time optimization predicts each user's ideal delivery moment based on historical engagement patterns
- Silent hours and inferred quiet periods respect user focus — blocking non-critical alerts during deep work or sleep
- Hybrid models blend recurring schedules with real-time event triggers, using platform controls (FCM/APNs priority, expiration, TTL)

Building a Priority Model: The Triage Matrix and Channel Mapping

- - The Alert Triage Matrix maps urgency × impact to channels: push for critical, in-app/digest for low-value
- - Priority scores combine social proximity, safety data, financial risk, and business rules
- - Labeling everything "urgent" destroys the priority gradient and trains users to disable all alerts
- - Default to lower urgency; earn high-urgency channels by proving value

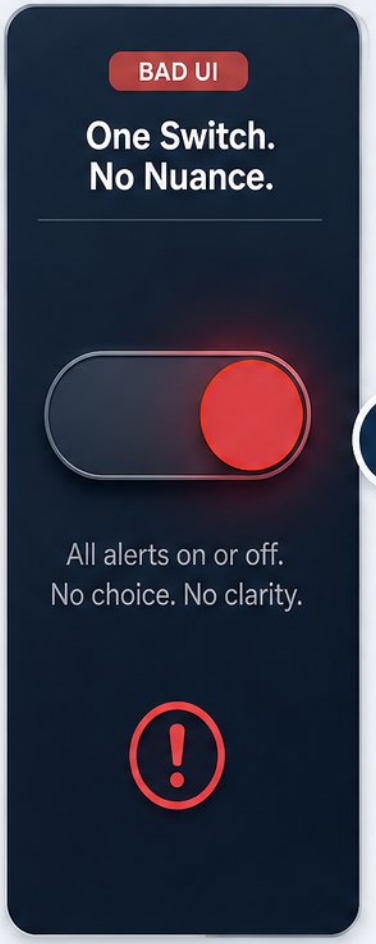
From Signal to Software: Priority Semantics in FCM and APNs

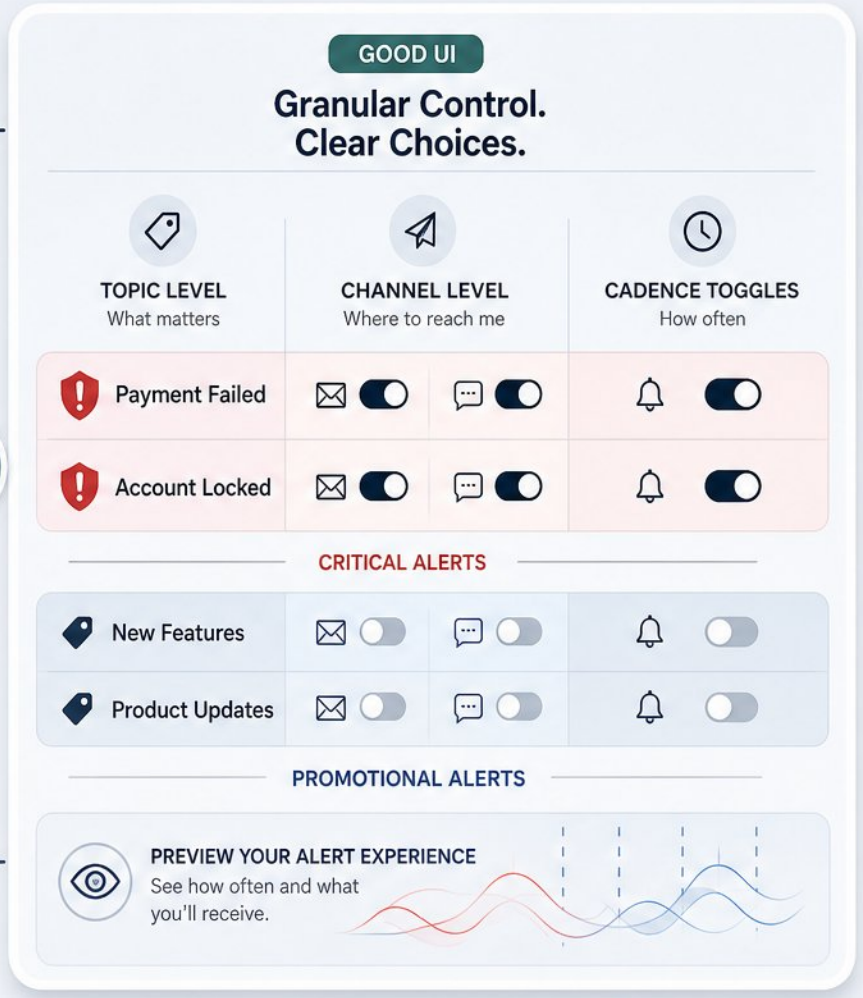


- Delivery priority (FCM high/normal) controls wake-up timing, not display importance.
- Interruption-level (time-sensitive) bypasses Focus for truly urgent iOS alerts.
- Map business rules: high + immediate expiry for fraud, normal + TTL for digests.
- A policy table prevents stale delivery: Critical, Time-Sensitive, and Informational lanes.

User Control:
Granular Preferences, Transparency, and Trust

- Spectrum of control: topic-level, channel-level, and cadence toggles
- Use concrete event names, like “Payment Failed” not “Billing”
- Preview frequency and visually separate critical vs. promotional alerts
- Pre-emptive control: quiet hours, budgets, and digests before overload





Preference Architecture: Modes, Digests, and Safe Defaults

- **Notification modes:** Calm, Regular, Power User bundles adjust frequency/channel volume without per-event tweaks

- **Digest design:** daily/weekly summaries for high-volume, low-urgency events; keep security and billing real-time

- **Safe defaults:** start quiet on non-critical channels; critical alerts (security, billing) bypass quiet hours

- **Provide clear digest vs.** real-time labels so users never get both unexpectedly

Designing Feedback Loops That Learn Without Friction

- Detect negative sentiment implicitly via dismissals, mute duration, and swipe-away velocity
- Capture frustration at the moment with in-line thumbs-down and "see less like this" controls
- Reprogram priority and timing engines in near real-time using streaming feedback data
- Prevent cascading fatigue by closing the loop between user signals and delivery models

Metrics That Matter: Measuring Notification Health, Not Just Opens

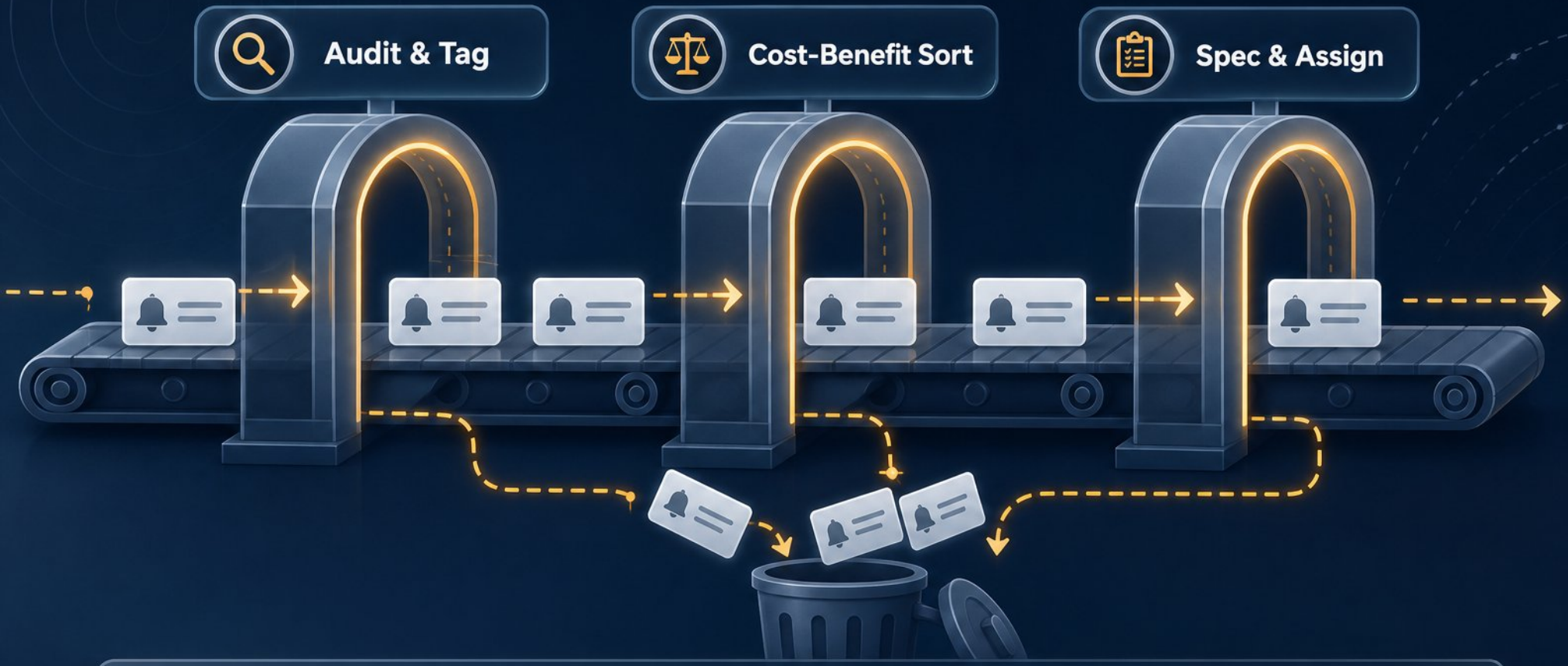
- Use conversion rate and incremental sessions as primary outcomes, not just CTR
- Guardrails: opt-out rate, mute rate, and uninstall risk signal trust erosion
- Introduce a Notification Load Score to cap throughput per user's attention
- Use long-term holdouts and staggered rollouts to detect fatigue early

Accessibility and Inclusivity in Notification Design

- Design beyond visuals: use haptics, LEDs, and audio for sensory-inclusive alerts
- Reduce sensory anxiety: respect prefers-reduced-motion and badge overload
- Standardize haptic patterns for error vs. success to build user trust
- Ensure critical safety alerts work on 2G/3G and low-spec devices
- Offer text alternatives and captions for all rich-media notifications

Inclusive by design.
Usable by everyone.

Practical Workshop: Auditing and Rebuilding a Notification System



- **Audit:** inventory events, tag by urgency & impact, calc mute rates, list top 10 noise candidates
- **Align:** score each event in a cost (trust) vs. benefit (value) matrix; route to must-send, digest, or suppress
- **Specify:** for one feature, define priority tier, timing window, channel, feedback mechanism, and fallback

Course Summary and Action Plan

- Synthesizing timing, priority, and control to respect cognitive rhythms
- Be timely, not just fast; prioritize ruthlessly to preserve the urgency gradient
- Measure health via mute/opt-out rates, not vanity opens
- Select one noisy notification for decommission or digest migration
- Implement one implicit feedback signal and draft a preference UI prototype

PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/52007758/public