Introduction to Designing Offline States

- Offline spans full outage, intermittent, and slow networks – each needs distinct UX.
- Thoughtful offline design builds user trust, boosts retention, and reduces churn.
- Common frustrations: blank screens, lost data, and unresponsive spinners.
- Core pillars: show what's available, what's restricted, and how to recover.

The Spectrum of Connectivity and User Psychology

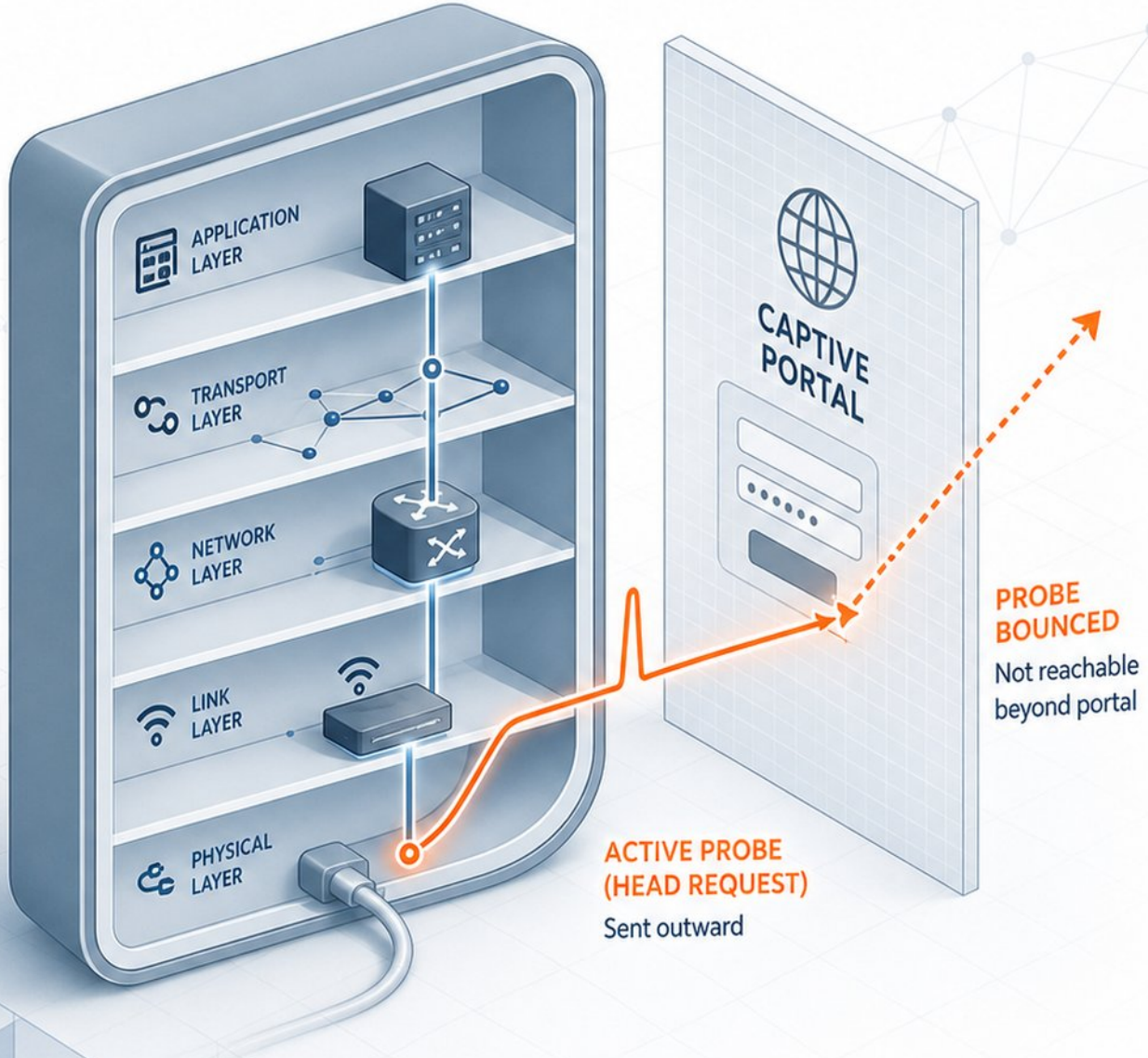
- Distinguish fully online, slow/unstable, and fully offline states — each demands a different UX response

- Uncertainty breeds anxiety; clear, supportive communication manages user emotions and maintains trust

- Poor offline experiences risk 9.5% of revenue on average, driving churn and channel spillover to competitors

- Treat offline-first as a product strategy, not a technical afterthought — plan from the start for resilience

Detecting Connectivity Accurately



- navigator.onLine only checks for network interface, not server reachability
- Active probes (HEAD requests) catch captive portals and dead networks
- Network Information API is Chromium-only; treat as progressive enhancement
- Background Sync fires even after tab close; `visibilitychange` is key on iOS Safari

Communicating What's Available, Restricted, and How to Recover

- Surface three core messages: accessible content, blocked actions, and the recovery path
- Use supportive, jargon-free language that never blames the user
- Disable—don't hide—unavailable controls, and always explain why
- Show examples of good vs. poor messaging for banners, toasts, and inline indicators

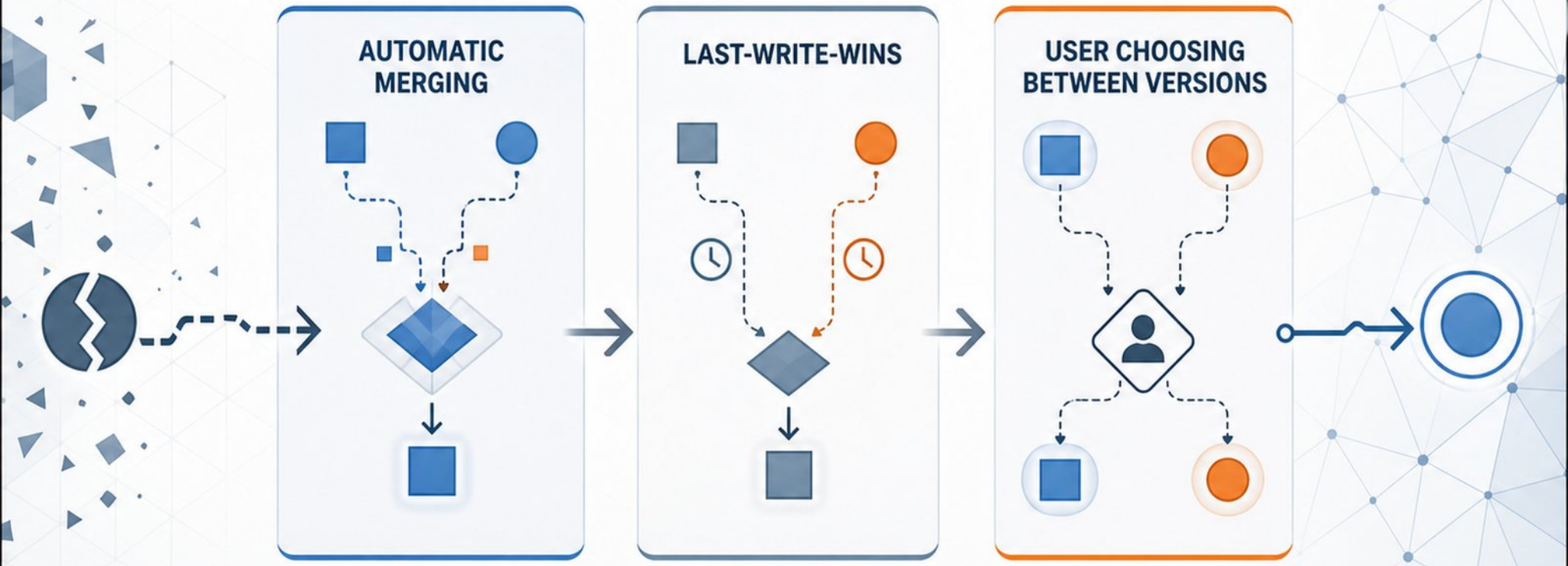
Caching Strategies for Reliable Content

- Cache-first for versioned static assets (JS, CSS, fonts); network-first for fresh API data.
- Stale-while-revalidate: serve cached content instantly, then update cache in the background.
- Precache the app shell at install time to guarantee instant offline rendering.
- Use timestamps and freshness indicators to label stale, cached, and live content.
- Respect storage limits with cache versioning; clean up old caches during the activate event.

Designing Action Queues and Offline Writes

- Save actions locally with a foreground queue and clear “pending” indicators.
- Separate “waiting to sync” from “failed to sync”—each needs its own UI treatment.
- Use idempotency keys and stable request IDs to prevent duplicate writes on retry.
- Give users manual controls: retry now, cancel queued items, and review the outbox.
- Never claim server success for actions that are only stored or queued locally.

Sync and Conflict Resolution Patterns



- Choose automatic triggers (Background Sync) or manual (online event fallbacks)
- Define conflict rules: last-write-wins, merge, or let users choose 'keep mine'
- Show sync progress clearly: 'Syncing...', 'All changes synced', 'Needs attention'
- Never claim a server save when data is only queued locally

Recovery Paths and Graceful Degradation

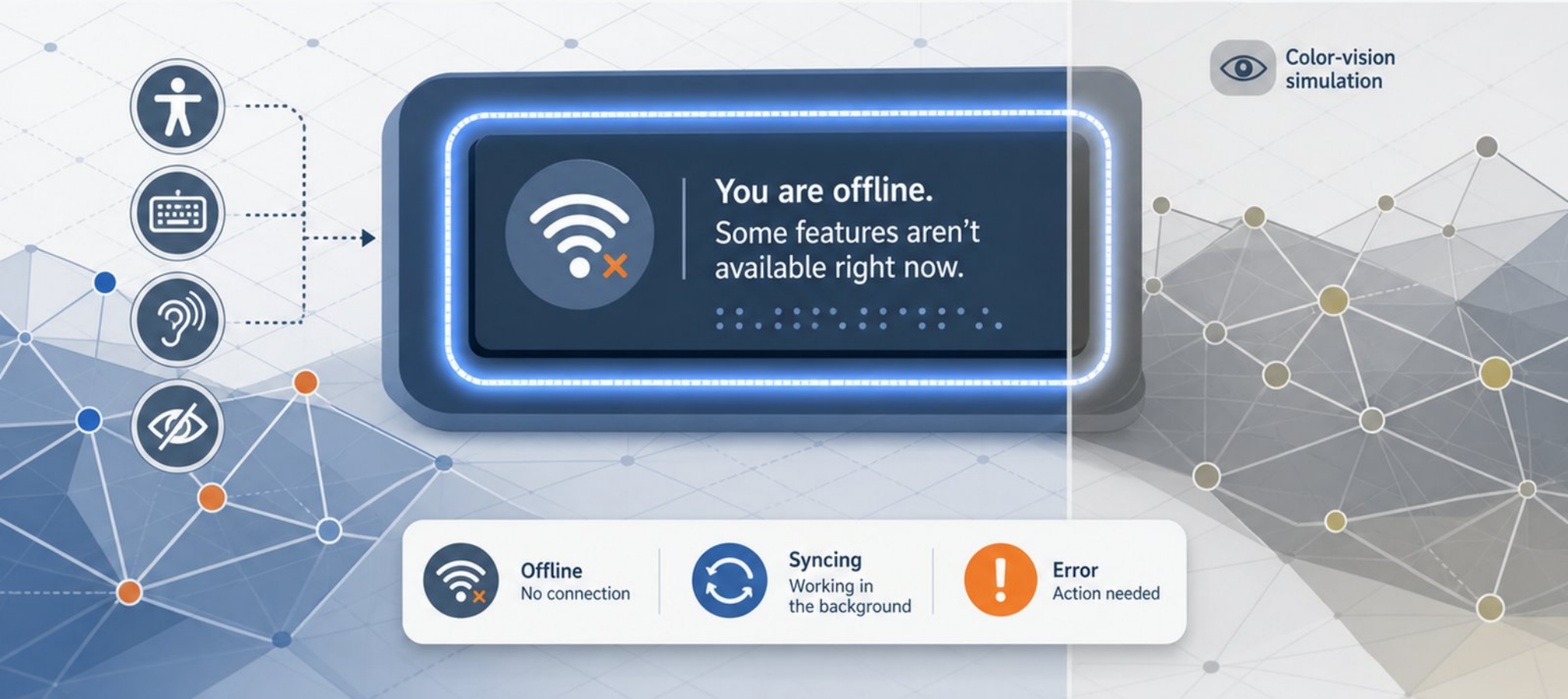
- “Design immediate retry, auto-backoff, and cancel controls”
- “Provide offline fallback: cached page or Outbox screen”
- “Handle expired auth tokens before queued actions sync”
- “Warn users before data loss: unsaved exits, full caches, sync exhaustion”

Cross-Platform and Platform-Specific Conventions

- Compare mobile native (iOS/Android) vs. desktop web and PWA offline expectations
- Respect platform-level indicators (OS status bar, browser UI) to avoid redundant noise
- Adapt for unique contexts: embedded/kiosk, field-service, healthcare, low-storage devices
- Safari lacks Background Sync; Android WebView has limits; desktop 'online' events can be unreliable

Inclusive and Accessible Offline Experiences

- - Announce connectivity changes to screen readers via `aria-live` polite regions.
- - Ensure offline indicators meet a 4.5:1 contrast ratio; never rely on color alone.
- - Manage keyboard focus logically: explain disabled controls, avoid traps.
- - Combine text labels, icons, and color to convey offline, syncing, and error states.



Testing and QA for Offline Scenarios

- “Go beyond toggling Wi-Fi: test mid-upload drops, captive portals, and expired auth tokens.”
- “Use a toolkit: DevTools throttling, Playwright setOffline, HAR replay, network simulators.”
- “Run cold-offline tests: install SW, go offline, navigate key routes.”
- “Validate full resumption: background sync return, UI reconciliation, last-chance failure handling.”

Measuring Success and Building an Offline Strategy

- Track offline task completion, sync success rate, time-to-sync, and rage-tap reduction.
- Use the Offline Capability Maturity Model: move from "Unaware" to "Transactional Read & Write."
- Audit every critical workflow; define its behavior per connectivity state.
- Prioritize fixes by user impact, unify messaging voice, iterate with real data.

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/198610a3/public