CONCEPT STUDY · INTERACTIVE PROTOTYPE

Stockroom: High-density warehouse inventory cycle counter with barcode rapid-fire and mandatory discrepancy reason codes.

Auditable discrepancy logging with mandatory business reason codes. When physical barcode counts diverge from system expectations, the application interrupts the rapid-fire scan sequence and requires an explicit, audited reason code (e.g. "Damaged in Bin", "Wrong SKU Mixed", "Supplier Short-Pack") before allowing submission.

INDUSTRY VERTICAL **Logistics & Field Operations**	DOMAIN SECTOR **Warehouse & Supply Chain Logistics**
PROTOTYPE STACK **Next.js / TypeScript**	TARGET PRODUCTION RUNTIME **React Native / Expo**

Executive Summary

Alex, a fulfillment warehouse specialist, performs weekly cycle counts in Bay 14 across 120 storage bins containing small electronic components. Handheld devices frequently allow blind tally increments, allowing damaged or mislabeled inventory to distort enterprise ERP records.

Silent inventory drift happens when warehouse staff bypass discrepancies to hit hourly scan quotas, resulting in stockouts during customer order fulfillment and costly emergency audits.

Operational Hurdles & The Failure of Conventional Approaches

In real-world mobile deployments within Logistics & Field Operations, applications encounter operational friction that desktop web portals never face.

Core Engineering Challenges Addressed:

1. Operational Bottleneck: Silent inventory drift happens when warehouse staff bypass discrepancies to hit hourly scan quotas, resulting in stockouts during customer order fulfillment and costly emergency audits.

2. Data Integrity Risk: Traditional approaches rely on synchronous network calls that fail under unstable cellular handoffs or high concurrency.

3. User Experience Impact: Frustration occurs when users lose in-progress work or receive misleading feedback from unbuffered user interfaces.

Auditable discrepancy logging with mandatory business reason codes. When physical barcode counts diverge from system expectations, the application interrupts the rapid-fire scan sequence and requires an explicit, audited reason code (e.g. "Damaged in Bin", "Wrong SKU Mixed", "Supplier Short-Pack") before allowing submission.

Architectural Solution & Interface Design

Demonstrated UX & Architecture Decision:

Auditable discrepancy logging with mandatory business reason codes. When physical barcode counts diverge from system expectations, the application interrupts the rapid-fire scan sequence and requires an explicit, audited reason code (e.g. "Damaged in Bin", "Wrong SKU Mixed", "Supplier Short-Pack") before allowing submission.

Principal Interface Screens (5 Views):

1. **Storage Bay & Aisle Selector**
   Warehouse map showing scheduled cycle count zones and completion percentages.
2. **Rapid-Fire Barcode HUD**
   High-contrast counter interface optimized for physical trigger buttons and audio chimes.
3. **Discrepancy Resolution Sheet**
   Variance comparison with mandatory reason code selection and optional photo note.
4. **Supervisor Authorization Queue**
   Manager approval dashboard for counts with financial impact exceeding thresholds.
5. **Bay Reconciliation Summary**
   Completed count overview with accuracy percentages and audit log export.

Interactive Journey: Continuous Bay Count to Mandated Discrepancy Commitment

Scan Bay 14 → Rapid-scan expected SKUs → Detect count variance (-3 units) → Select mandatory reason code → Submit to supervisor queue → Confirm bay audit.

1. Open Stockroom and select "Bay 14-C: Microcontrollers"
2. Engage continuous scanner: scan 15 units of SKU #8841-A
3. System flags expected count was 18 units (variance: -3 units, -$45.00)
4. Interface locks submission and prompts mandatory discrepancy reason
5. Select reason code "Damaged / Packaging Crushed in Bin" and attach note
6. Submit variance to supervisor queue and review reconciled bay status

Reset Mechanism: One-click reset to initial uncounted bay inventory.

Handled Edge Cases & Exception Recovery

⚠ **Edge Condition #1:**
Unrecognized barcode scan modal allowing instant on-floor SKU lookup or bin reassignment

⚠ **Edge Condition #2:**
Accidental double-scan throttle preventing accidental duplicate reads within 300 milliseconds

⚠ **Edge Condition #3:**
Offline batch count synchronization queue storing up to 5,000 scans during warehouse dead zones

Actual Prototype vs. Target Native Production Runtime

Actual Prototype Stack

Platform & Framework:

Next.js / TypeScript

Languages:

TypeScript, Tailwind CSS

Warehouse scanning state machine simulating continuous barcode entry, count variances, and supervisor sign-offs.

Target Production Stack

Platform & Framework:

React Native / Expo

Languages & Storage:

TypeScript, Honeywell / Zebra Scanner SDK, SQLite

Hardware integration with enterprise laser scan engines via native intent listeners and high-speed local caching.

Explicit Prototype Boundaries & Validation Methodology

Documented Prototype Limitations:

- Simulated in browser using keyboard and touch triggers; does not connect to physical Zebra DataWedge / Honeywell laser scan engines.
- ERP reconciliation simulated without live SAP / Oracle NetSuite database connections.

Target Production Telemetry:

Measure reduction in inventory count variance and audit hours across 3 distribution facilities.

Commercial Offer Alignment & Handover Guarantees

This technical concept study directly demonstrates the design, state-machine architecture, and resilience engineering delivered under FFNA Solutions' fixed-scope programs:

Product Prototype Sprint
2-week fixed sprint delivering high-fidelity interactive simulation with edge recovery.

Mobile MVP Launch
6–8 week end-to-end production build with backend API, store release, and full IP transfer.

Handover Guarantee
100% intellectual property ownership transferred upon milestone completion. Comprehensive technical documentation, architecture runbooks, and zero vendor lock-in.