Product Requirements Document (PRD)

Product Name

Kushim — The Ambient Ledger

Document Purpose

This PRD defines the functional, technical, and experiential requirements for building Kushim end-to-end. It is intended for product, engineering, design, and ML teams to execute without additional conceptual clarification.

---

1. Problem Statement

Knowledge work is fragmented across tools (Jira, GitHub, Slack, Email, Docs). Humans are forced to manually reconstruct context by correlating identifiers, timelines, and conversations across systems. This violates core Human-Computer Interaction (HCI) constraints, particularly working memory limits and recognition-over-recall principles.

Current tools either: - Aggregate information (dashboards, feeds), increasing noise, or - Require users to manually organize context (folders, links, projects)

None perform the cognitive task of contextualization automatically.

---

2. Product Vision

Kushim is an ambient work context system that automatically discovers, links, and maintains relationships between digital artifacts across platforms, presenting them as coherent cognitive units.

Vision Statement

> Kushim eliminates the mental tax of context reconstruction by assembling the complete picture of any project or initiative from its scattered digital footprints—allowing users to think about problems, not platforms.

---

3. Target Users & Personas

Primary Persona: Knowledge Worker

- Roles: Software engineers, product managers, tech leads, consultants

- Environment: Multi-tool, async-heavy, high context switching
- Pain: Loses time and cognitive energy reconstructing "what's going on"

Secondary Persona: Team Lead / Manager

- Needs rapid situational awareness across initiatives
- Values compressed, high-signal summaries

---

4. Success Metrics (KPIs)

Core Metrics

- Context Reconstruction Time (CRT): ↓ 50%
- Average Platforms Visited per Task: ↓ 40%
- Daily Active Context Groups Used: ↑
- Action Completion Without Tool Switching: ↑

Quality Metrics

- Link Precision / Recall (ML evaluation)
- User-confirmed vs auto-generated links
- False-positive cluster rate

---

5. Core Concepts & Definitions

Artifact

Any discrete object from an external system (ticket, PR, message, doc).

Context Group

A dynamically generated cluster of related artifacts representing one cognitive work unit.

KushimStandardRecord (KSR)

The normalized internal representation of all ingested artifacts.

---

6. Functional Requirements

6.1 Data Ingestion

Supported Platforms (v1) - Jira - GitHub - Slack - Google Docs - Email (Gmail)

Requirements - OAuth-based authentication - Read-only by default; scoped write permissions for actions - Incremental sync + webhook support where available

---

6.2 Normalization Layer

All ingested data MUST be converted into a KSR schema.

KSR Core Fields - id (internal) - external_id - source_platform - artifact_type - title - body/content - author(s) - timestamp(s) - participants - metadata (labels, branch names, ticket IDs, etc.)

6.3 Feature Engineering

Extract linking signals including but not limited to: - Explicit identifiers (e.g., PROJ-101) - Temporal proximity - Shared participants - Keyword similarity - Branch / commit references - URL cross-references

---

6.4 Relationship Modeling

Phase 1 (Deterministic) - Rule-based heuristics with confidence scoring - Hard links (explicit IDs) - Soft links (signal aggregation threshold)

Phase 2 (ML-augmented) - NLP embeddings for semantic similarity - Graph-based clustering - User feedback as labeled data

---

6.5 Context Group Engine

Requirements - Automatic creation and update of context groups - Groups evolve over time - One artifact may belong to multiple groups (with weighting)

6.6 User Interface

6.6.1 Global Command Bar

- Universal entry point (Cmd/Ctrl + K)
- Search, navigate, act
- Natural language queries

6.6.2 Ambient Feed

- Live, real-time updates
- Grouped by context, not source
- Human-readable summaries

6.6.3 Context Group View

- Timeline of related artifacts
- Strong information scent
- Expand/collapse detail levels

6.6.4 Integrated Action Center

- Comment, assign, merge, reply
- Abstracted command syntax
- Native execution via APIs

---

7. Non-Functional Requirements

Performance

- Context group load < 300ms
- Sync latency < 5 seconds (webhooked sources)

Reliability

- Eventual consistency guaranteed
- Graceful degradation if a platform is unavailable

Security

- OAuth token encryption
- Least-privilege scopes
- SOC2-ready architecture

---

8. Architecture Overview

High-Level Components

- Ingestion Services (per platform)
- Normalization Pipeline
- Feature Extraction Service
- Relationship Engine
- Context Group Store (Graph DB recommended)
- API Gateway
- Frontend (Web + Desktop)

---

9. Data Model

Storage

- Relational DB: Users, auth, metadata
- Graph DB: Artifacts, relationships, context groups
- Vector Store (Phase 2): Embeddings

---

10. Feedback & Learning Loop

Signals - User opens a group - User dismisses a link - Manual re-grouping

Usage - Reinforce or penalize link heuristics - Improve clustering thresholds

---

11. Rollout Plan

Phase 1: MVP

- Jira + GitHub + Slack
- Deterministic linking
- Read-only actions

Phase 2: Intelligence Layer

- NLP-based similarity
- Feedback learning

Phase 3: Platform Expansion

- Docs, Email, Calendar

---

12. Risks & Mitigations

Risk	Mitigation
Over-linking noise	Conservative confidence thresholds
Privacy concerns	Explicit permissions + transparency
User trust	Explainable links

---

13. Open Questions (Post-MVP)

- Cross-org context boundaries
- Long-term knowledge retention
- Proactive surfacing vs passive ambient mode

---

14. Definition of Done

- User can answer "What’s going on with X?" without visiting multiple tools
- Context groups form automatically with >80% precision
- Actions executed without platform switching

End of PRD