NexusMap Product Requirement Document

1. Vision & Purpose

NexusMap is a high-velocity Information Architecture (IA) and UX mapping platform. It bridges the gap between static mind maps and complex logic-based prototyping by allowing designers to map user flows, conditional logic, and persona-based visibility at the "speed of thought."

2. System Foundation: "NexusMarkdown"

The core engine treats a specialized Markdown syntax as the single source of truth, enabling instant portability and LLM-ready data structures.

2.1 NexusMarkdown Syntax Rules

- **Hierarchy:** Indentation (2 spaces or 1 tab) defines parent-child relationships. Multiple unindented lines represent independent root nodes or sub-graphs (supporting Zoom/Isolation mode).
- **Conditions:** `(key=value)` - All indented items under this line belong to this logical state. For multi-dimensional logic, multiple conditions can be comma-separated: `(status=pending, type=form)`.
- **Shared Nodes:** `(common)` - Indicates this node is an instance of a global node. Editing one updates all others with the same text.
- **Cross-Links & Relationships:** `--> [Node Name]` - Creates a dashed non-hierarchical connection. Labels or cardinality can be appended: `--> [Node Name] (1:N)`.
- **Metadata/Personas:** `[persona: name]` or `[tag: name]` - Sets visibility or categorization. Multiple tags can be combined: `[persona: admin][tag: internal]`.
- **Line Breaks:** Standard text within nodes supports line breaks (using Shift+Enter in the UI).

3. Core Functional Requirements

3.1 High-Velocity Mapping (Phase 1)

- **Keyboard-First Interaction:**
  - `Tab` : Create child.
  - `Enter` : Create sibling.
  - `Shift+Enter` : New line within the same node.
- **Smart Drag-and-Drop:** Dragging a parent node automatically groups and moves all children/descendants.
- **Auto-Layout:** Nodes snap to a clean, readable grid while maintaining the "logic tree" shape.

3.2 Conditional Content & Multi-Dimensions (Phase 2)

- **Node-Level Conditions:** Any node in the tree can be designated as a "Conditional Hub." Applying a condition (e.g., `status`) to a node automatically encompasses and partitions all of its children and descendants based on the condition values.
- **1D Navigation:** For a selected conditional node, its possible values (e.g., `status=pending`, `status=approved`) are shown as toggleable button tabs. Selecting a tab instantly swaps the visible subtree beneath that node.
- **2D Matrix View:** A grid-based popup triggered from a conditional node where X-axis is Condition A and Y-axis is Condition B. This allows viewing the intersection of two logic sets for a specific branch.
- **3D+ Filter View:** For nodes with three or more nested or overlapping conditions, dropdown selectors are provided to filter the specific logic state of the subtree.
- **Shared Node Logic:** If a node exists in multiple conditional paths (e.g., a "Cancel" button that appears in both `pending` and `rejected` states), marking it as `(common)` ensures that any changes to its content or children sync globally.

3.3 Deep Mapping & Personas (Phase 3)

- **Isolation/Zoom Mode:** Double-click or select "Zoom" to focus exclusively on one node's sub-tree. Nodes created within this mode do not strictly have to be part of the primary tree structure; they can exist as independent sub-graphs or be mirrored instances of nodes sourced from entirely different branches/trees to show interconnected logic.
- **Relationship Indicators:** Toggle between simple lines, flow arrows, or data cardinality labels (1:N, M:M).
- **Persona Management & Visibility:** A dedicated page-level setting and management view for defining access and visibility across the entire tree.
  - **Persona List Management:** A centralized interface to create, edit, and delete user personas (e.g., Admin, Guest, Premium).
  - **Assignment View:** A list-based management interface where designers can rapidly assign or restrict node access per persona.
  - **Tag-Based Overrides:** Set visibility overrides at the node level using tags for granular control.
  - **Inherited Hiding:** If a parent is hidden for a persona, all children are hidden by default unless an explicit tag-based exception is granted.

3.4 AI Ingestion & Testing (Phase 4 & 5)

- **Vision-to-Tree:** Upload image (whiteboard, FigJam) → Multimodal AI visual digestion → Generate NexusMarkdown → Render Tree.
- **Mermaid/LLM Import:** Auto-convert standard Mermaid `graph TD` or chat-based descriptions into NexusMap structures.

- **Interactive Card Sort & Tree Test Generator:** A specialized module to convert structural trees into interactive research environments for user validation.
  - **Test Configuration:** Designers select a specific branch or the entire tree, choose a target persona, and set the "initial state" (pre-defined condition values).
  - **Participant Navigation:** Users receive a live link where they navigate through the tree branch-by-branch in a "drill-down" interface.
  - **Dynamic Logic Processing:**
    - If a participant clicks a node that triggers a state change, the generator re-renders navigation options in real-time.
    - Shared `(common)` nodes maintain their state across different branches during the test.
  - **Result Tracking:** The system logs participant "paths," identifying where users struggled to find specific information.

4. Platform & Collaboration Requirements

4.1 Multi-User Multiplayer

- **Real-time Sync:** Powered by CRDTs (Conflict-free Replicated Data Types) or WebSockets to ensure no data loss during simultaneous edits.
- **Presence:** Show user cursors, avatar labels, and the node currently being edited.
- **Scoped Undo:** Undo history is per-user, not per-document.

4.2 File & Folder Management

- **Hierarchy:** Folder > Project File > Multiple Pages.
- **Sharing:** Folder-level or File-level access with specific permissions (Owner, Editor, Viewer).

Implementation Roadmap

Phase 0: The "Steel Thread" & Multi-User Infrastructure

Building the foundation for a professional-grade collaborative platform. This phase focuses on the architecture required for sharing and real-time synchronization.

- **Identity & Authentication:** Implementing secure user accounts and session management.
- **Collaboration Engine:** WebSocket server architecture and CRDT integration (Yjs/Automerge).
- **Permission & Sharing Logic:** Access Control List (ACL) system for folders and files.
- **NexusMarkdown Core:** High-performance parser to sync visual graph and text.
- **Basic Tree Renderer:** SVG/Canvas implementation with remote cursor presence.

Phase 1: High-Velocity UI & File Management

- **Keyboard Interaction Driver:** Full implementation of Tab, Enter, and Shift+Enter logic.
- **Group Manipulation:** Bounding box logic for automatic descendant grouping during moves.
- **Workspace Explorer:** Folder, Project, and Page interface with invitation/sharing UI.

Phase 2: Node-Level Logic & Dimensions

- **Condition Engine:** `(key=value)` parsing and logical subtree inheritance.
- **Dimensional UI Tools:** Node-contextual Tab-bar, Matrix Grid Popup, and Filter Menus.
- **Global Node Syncing:** UUID-based cross-referencing for `(common)` nodes.

Phase 3: Advanced Visualization & Personas

- **Sub-Canvas Architecture:** Zoom/Isolation mode with support for independent/mirrored sub-graphs.
- **Persona Management Page:** Dedicated list-management and assignment interface for persona-based visibility.
- **Structural Styling:** Flow arrows, dashed lines, and cardinality labels (1:N, M:M).

Phase 4: AI Ingestion & Assisted Authoring

- **Multimodal Vision Bridge:** API bridge to models (e.g., Gemini Flash) for image-to-IA digestion.
- **AI Architecture Copilot:** Chat sidebar for restructuring and generating branches via natural language.
- **Contextual Suggestions:** AI-driven analysis of node labels to suggest missing branches or logical inconsistencies.

Phase 5: Interactive Simulation Engine (Tree Testing)

A standalone phase dedicated to the complex logic required for research validation and dynamic state-aware testing.

- **Participant Player:** A standalone, lightweight interface for test participants to navigate trees without the full editor.
- **State-Change Logic Mapper:** A UI for designers to link specific nodes to "State Change Events" (e.g., clicking node "Confirm" sets `status=success`).
- **Real-time Reactive Re-rendering:** Logic that instantly rebuilds navigation branches for participants based on condition-changing clicks.
- **Logic-Aware Path Tracking:** Recording navigation breadcrumbs that account for conditional state changes.
- **Condition Navigation (Buttons/Matrix/Dropdown):** Participant-side controls to manually toggle conditions if the test configuration allows it.