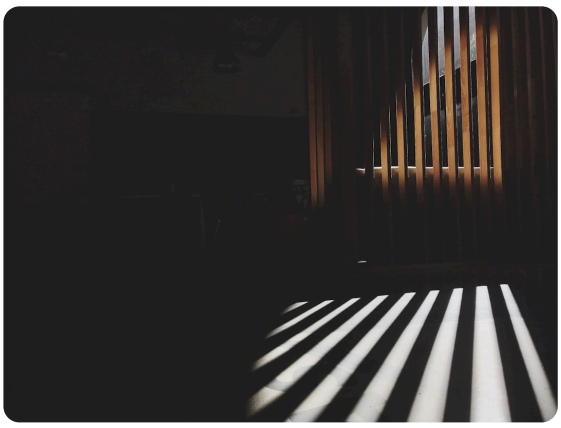


Project

Penumbra Requirments Gathering

Startup OrganizedKnowledge Claude'sPlan

2.1.6.3.4 • 2.1.6.3.4 Deploy Roo Modes

2.1.7 2.1.7 Statement of Work Conversational Agent Flow

2.1.8 CLOUDFLARE Deploy to GCP & Collaborative / staging space for 2.1.8 Reddit Research MCP Server

2.1.8.1 Deliverables

2.1.9 Knowledge Context

2.1.9.1 BlackNoir

2.1.10 2.1.10 Review Ops OS & Sync With Andrew

2.1.10.1 Ops OS Google Doc

2.1.11 Implement MCP Inspector

2.1.12 Containerization

2.1.12.1 Docker

2.1.12.2 Docker Compose

2.1.12.3 Docker Bake

2.1.12.4 CI/CD

2.1.13 2.1.13 Migrate Neo4J

2.1.14 Nitric & Uffizzi

2.1.14.1 Uffizzi GitHub Action

2.1.14.2 Uffizzi Docs

2.1.14.3 Uffizzi Docs

2.1.15 Web Search, Crawl & Auto ETL

2.1.15.1 Crawl4AI

2.1.15.2 LightRAG

2.1.16 Browser Control, Open Manus, Open Operator, Browser User, Proxy Light, HyperBrowser, etc

2.1.16.1 Browser Use

2.1.16.2 Browser Use Web UI

2.1.16.3 Open Manus

2.1.16.4 Browser Base

2.1.16.5 Proxy Lite

2.1.16.6 HyperBrowser

2.1.16.7 BrowserBase

2.1.16.8 Open Computer Agent

2.1.17 XPRT Commitzen

2.1.18 2.1.18 Cloudflare (remember / find out what Was to do on Cloudflare)??

2.1.18.1 Cloudflare Dashboard

Penumbra Context

Page

Penumbra Project Summary

penumbra Reasoning as a Service Requirements-Gathering

1 1 Overview

1.1 Key aspects include:

1.1.1 Core Purpose1.1.1 :

1.1.2 Technical Stack1.1.2 :

1.1.3 Key Features1.1.3 :

1.2 1.2 Problem & Solution

1.2.1 The core problems Penumbra solves:

1.2.1.1 1. 1.2.1.1 Reducing cognitive load for strategic thinkers

1.2.1.2 2. 1.2.1.2 Amplifying strategic impact by providing a dedicated space to map one's strategic mind

1.2.1.3 3. 1.2.1.3 Moving beyond static documents like decks and docs into a living system that reflects evolving worldviews

1.3 1.3 Cognitive Shapes Architecture

1.3.1 Key Components

1.3.1.1 1. **1.3.1.1 Cognitive Shapes**1.3.1.1 :

1.3.1.1.1 Examples:

1.3.1.2 2. 1.3.1.2 Self-contained MCP Server instances, each embodying a specific cognitive function:

1.3.1.2.1 1. **1.3.1.2.1 MCP Primitives within Shapes**1.3.1.2.1 :

1.3.1.2.2 2. **1.3.1.2.2 Intent Router**1.3.1.2.2 :

1.3.1.3 3. **1.3.1.3 UI Modes**1.3.1.3 :

1.3.1.3.1 4. **1.3.1.3.1 Tool Library**1.3.1.3.1 :

1.3.1.3.2 5. **1.3.1.3.2 Presentation/Persona Layer**1.3.1.3.2

1.3.1.3.3 Context Management1.3.1.3.3 : Maintains state across interaction

1.4 1.4 Data Storage in Penumbra: PostgreSQL and Neo4j

1.4.1 PostgreSQL (via Drizzle ORM)

1.4.2 Neo4j (via neo4j-driver)

1.4.3 Data Flow

1.4.4 Cross-Database References

Overview

Penumbra is a cognitive organizing layer designed for founders and strategic thinkers to

- externalize
- structure
- and evolve

their core thinking into a dynamic knowledge graph.

Key aspects include:

Core Purpose:

- Reduce cognitive load and amplify strategic impact by mapping users' strategic minds

Technical Stack:

- Next.js/React frontend with Tailwind CSS and Radix UI
- Neo4j graph database for knowledge representation
- PostgreSQL (via Supabase) for structured data
- BAML for AI orchestration with multiple LLM providers

Key Features:

- Multi-agent orchestration with RAG and knowledge graph capabilities
- Interactive 3D knowledge graph visualization
- Agent-driven project documentation
- Conversational interface for capturing and structuring thoughts

The system helps users move from static documents

- to a living system that
  - reflects their evolving worldview
  - with AI assistance to
    - organize thoughts into
      - structured knowledge

Problem & Solution

Penumbra is a cognitive organizing layer designed primarily for founders and strategic thinkers.

It helps users

- externalize
- structure
- connect
- and evolve their
- core thinking
  - (beliefs, principles, strategies, and narratives)
- into a dynamic, queryable knowledge graph.

The core problems Penumbra solves:

1. Reducing cognitive load for strategic thinkers
2. Amplifying strategic impact by providing a dedicated space to map one's strategic mind
3. Moving beyond static documents like decks and docs into a living system that reflects evolving worldviews
   - Lightweight capture of core thoughts
   - AI-assisted structuring into a knowledge graph
   - Progressive deepening of specific areas
   - Semantic graph database (Neo4j) for complex queries and relationship analysis

Penumbra is currently in early development/beta phase, focusing on refining the core worldview modeling experience and lightweight onboarding process.

Cognitive Shapes Architecture

The Cognitive Shapes architecture is a modular AI system that

- organizes capabilities

- around cognitive functions
- rather than personas.

It solves limitations of monolithic agent designs

- by enabling specialized
- composable reasoning modules.

Key Components

1. Cognitive Shapes:

Examples:

- `StrategicPlanningShape`
- `DataAnalysisShape`
- `SequentialReasoningShape`
- Implemented as independent serverless functions (Supabase Edge Functions)

2. Self-contained MCP Server instances, each embodying a specific cognitive function:

1. MCP Primitives within Shapes:

Tools:

- Define active capabilities, including:
  - Action/Query Tools (calling external services)
  - Internal Logic Tools (performing calculations)
  - Reasoning Framework Tools (structuring LLM thinking)

Resources:

- Passive, read-only data context

Prompts:

- User-controlled interaction templates

2. Intent Router:

Orchestrates the system by:

- Classifying user intent
- Selecting appropriate Shape(s)
- Managing permissions via UI Mode filtering
- Facilitating tool execution

3. UI Modes:

Define interaction context, persona, and permission boundaries

- Filter which Shapes and tool groups are accessible

4. **Tool Library:**
   - Foundation of deterministic functions called by Shape Tools

5. **Presentation/Persona Layer**

Context Management: Maintains state across interaction

This architecture enables modular, secure, workflow-oriented AI interactions with clear separation of cognitive functions from presentation personas.

Data Storage in Penumbra: PostgreSQL and Neo4j

Penumbra uses a dual-database architecture to handle different types of data:

PostgreSQL (via Drizzle ORM)

- **Hosted by**: Supabase
- **Stores structured data**:
  - Users and authentication data
  - Organizations and membership
  - Chat sessions and messages
  - Document/artifact metadata
  - API keys

TypeScript

index.ts

```
// Create drizzle client
const db = drizzle(client, {
  schema: {
    ...organizations,
    ...organizationMembers,
    ...apiKeys,
    ...chat,
    ...users,
  },
});
```

Neo4j (via neo4j-driver)

- **Stores graph data**:
  - Knowledge graph nodes (Beliefs, Principles, Strategic Initiatives, etc.)
  - Relationships between nodes

- Properties of nodes and relationships

```typescript
route.ts
app/api/graph
const driver = neo4j.driver(
  NEO4J_URI,
  neo4j.auth.basic(NEO4J_USER, NEO4J_PASSWORD),
);
const session = driver.session();
```

Data Flow

1. **API Routes**: Handle data operations through Next.js API routes
2. **Authentication**: Managed via Supabase middleware
3. **Data Operations**:
   - PostgreSQL: Accessed via Drizzle ORM
   - Neo4j: Accessed via neo4j-driver with utility functions

Cross-Database References

- Document metadata in PostgreSQL may reference graph nodes in Neo4j
- Graph nodes may contain references to documents/artifacts

This architecture separates structured data (PostgreSQL) from complex interconnected knowledge (Neo4j), optimizing for their respective strengths.

BAML Integration in Penumbra

BAML (Better AI Markup Language) is integrated into Penumbra to provide structured, type-safe interactions with LLMs. It serves as the AI orchestration layer for handling complex AI tasks.

Core Components

1. **BAML Source Files** (`baml_src/`)
   - Define AI functions, data structures, and prompts
   - Configure LLM clients and retry policies

TypeScript

```
client<llm> CustomSonnet {
  provider anthropic
  options {
    model "claude-3-5-sonnet-20241022"
    api_key env.ANTHROPIC_API_KEY
  }
}
```

1. Generated Client (`baml_client/`)

- Auto-generated TypeScript code from BAML definitions
- Provides type-safe interfaces for AI functions

TypeScript

```
async ExtractResume(
  resume: string,
  __baml_options__?: BamlCallOptions,
): Promise<Resume> {
  try {
    const options = { ...this.bamlOptions, ...(__baml_options__ || {}) };
    const collector = options.collector
      ? Array.isArray(options.collector)
        ? options.collector
        : [options.collector]
      : [];
    const raw = await this.runtime.callFunction(
      "ExtractResume",
      {
        resume: resume,
      },
      this.ctxManager.cloneContext(),
      options.tb?.__tb(),
      options.clientRegistry,
      collector,
    );
    return raw.parsed(false) as Resume;
  } catch (error) {
    throw toBamlError(error);
  }
}
```

Key Features

1. **Type-Safe Outputs**
   - Defines structured data schemas for LLM responses
   - Validates and parses JSON outputs automatically

2. **Prompt Management**
    - Centralized prompt definitions with templating
    - Context-aware output formatting
3. **Client Configuration**
    - Multiple LLM providers (Anthropic, OpenAI)
    - Configurable retry policies
4. **Integration Points**
    - Canvas component generation
    - Resume/document parsing
    - Chat functionality

Usage Pattern

1. Define data structures and functions in BAML
2. Generate client code with `baml-cli generate`
3. Import and use type-safe functions in application code:

TypeScript
```
canvas.ts
lib/ai
import { b } from 'baml_client';
// Type-safe AI function call
const components = await b.GenerateCanvasComponents(
  message,
  existingComponents
);
```

This approach provides robust error handling, type safety, and a clean separation between AI logic and application code.

Penumbra User Interaction Flow

Here's the typical flow of a user interaction from frontend through backend to AI services:

1. User Interaction Layer

User interacts with one of the primary interfaces:

- Chat Interface (`components/chat.tsx`)
- Artifact Interface (`components/artifact.tsx`)
- Graph Panel (`components/graph/GraphPanel.tsx`)
- Canvas Interface (`app/canvas/page.tsx`)

```typescript
return (
  <CanvasContextProvider
    value={{
      components,
      setComponents,
      onComponentInteraction: handleComponentInteraction,
      streamingEnabled,
      setStreamingEnabled,
    }}
  >
    <div className="relative w-full h-full">
      {/* The canvas component that visualizes the chat messages */}
      <GenerativeCanvas
        chatHistory={messages || []}
        onComponentInteraction={handleComponentInteraction}
        canvasRef={canvasRef}
      />

      {/* Chat interface for user interaction */}
      <ChatInterface />
    </div>
  </CanvasContextProvider>
);
```

2. Frontend Processing

- Frontend components process user input
- State management (Zustand/Hooks) updates local state
- API request is prepared and sent to backend

3. Backend API Processing

Request is handled by appropriate Next.js API route:

- Chat API (`app/api/chat/route.ts`)
- Graph API (`app/api/graph/nodes/route.ts`)
- Document API (`app/api/document/route.ts`)
- File API (`app/api/files/upload/route.ts`)
- Canvas API (`app/api/canvas/route.ts`)

```typescript
export async function POST(request: NextRequest) {
  try {
    // Parse request body
    const { message, existingComponents } = await request.json();

    if (!message) {
      return new Response(encodeData({ error: "Message is required" }), {
        status: 400,
        headers: {
          "Content-Type": "text/event-stream",
          "Cache-Control": "no-cache",
          Connection: "keep-alive",
        },
      });
    }
```

4. Intent Routing (for AI requests)

- Intent Router analyzes request to determine cognitive function needed
- Selects appropriate Cognitive Shape based on intent
- Filters available tools based on UI Mode permissions

5. AI Orchestration (BAML)

- BAML Client invokes appropriate function from BAML definitions
- BAML interacts with AI SDK to format prompts and handle responses
- AI SDK calls external LLM providers (OpenAI, Anthropic)

```typescript
function GenerateCanvasComponents(
  chatMessage: string,
  existingComponents: CanvasComponent[]
) -> CanvasComponent[] {
  client CustomSonnet // Use the defined client from clients.baml
  prompt #"
    You are a creative AI assistant helping to generate a visual workspace
    based on the user's conversation. Generate UI components that represent
    the key concepts, relationships, and actions from the conversation.
```

6. Data Storage

- PostgreSQL (via Supabase): Stores structured data (chat history, user data)
- Neo4j: Stores graph data (knowledge graph nodes and relationships)

7. Response Flow

- AI response returns through BAML to Backend API
- Backend API formats and streams response to Frontend
- Frontend updates UI with response data
- User sees results (chat response, updated graph, new canvas components)

Page

PEN - Requirements Gathering

What is Penumbra?

Penumbra could be described as a "meta-assistant" tool for teams:

- Is likened to "having a film room for your brain"
- Functions as a "coach" that watches everything and identifies patterns
- Connects information across documents, conversations, and tasks

- Helps teams think and work smarter

Key Differentiators:

- **Not just an AI assistant** but a meta-assistant that connects how teams think
- **Bridges the gap between different tool types**:
  - Storage tools (like Notion/Google Drive) that are good for storing but static
  - Response tools (like Slack/ChatGPT) that are quick but forgetful

Core Benefits:

1. **Makes faster, smarter decisions** by connecting strategy, documents, and metrics to show the full picture
2. **Eliminates information hunting** by linking everything across tools and surfacing relevant information

Business Status:

A knowledge management and team intelligence tool that aims to provide contextual awareness across a company's information ecosystem.

Founder Mode (Higlight)

Penumbra is ambitious.

We're not just building another AI tool;

We're building a "cognitive organizing layer" to help founders like us manage the complexity of their strategic thinking. That's huge!

Here's how we could approach it:

1. Focus on "Conviction-Led Founders":

- We already know our target user. Double down on them. Understand their specific pain points.
- What keeps them up at night? How can Penumbra directly alleviate that stress?

2. Make it ridiculously easy to capture thoughts:

- The "lightweight capture" is key. Think voice memos, quick text snippets, mind-mapping integrations.
- The barrier to entry needs to be *zero*. If it feels like a chore, founders won't use it.

3. AI-assisted structuring is your secret weapon:

- This is where the magic happens. Use the MCP server to intelligently suggest connections, identify inconsistencies, and build the knowledge graph *automatically*.
- The less manual work, the better.

4. Progressive Deepening, not overwhelming complexity:

- Don't bombard users with a million options upfront. Guide them through the process, revealing more advanced features as they become comfortable.
- Think of it like a video game tutorial.

5. Treat the Knowledge Graph as a strategic asset:

- The Neo4j graph database is powerful. Use it to generate insights that founders can't get anywhere else.
- Show them hidden connections, potential risks, and untapped opportunities.

6. Automate the boring stuff:

- Use the structured worldview to generate drafts of decks, pitches, and plans. This is where Penumbra becomes a force multiplier.
- Founders can focus on the big picture, not the tedious details.

7. Community, Community, Community:

- Connect your users with each other. Create a space where they can share their worldviews, exchange ideas, and learn from each other.
- This will create a powerful network effect.

8. Solve The fundamental issues:

- I see some critical issues that will be ongoing. I prioritize fixing this

This is a long-term vision, but it's worth pursuing. We have the potential to create something truly transformative for founders. Don't get bogged down in the details. Stay focused on the big picture, and keep iterating based on user feedback.

Project: Beta User Feedback & Iteration

Goal: Establish a systematic approach to beta user feedback collection and analysis for rapid product improvement.

Strategic Goals:

- Establish clear feedback collection mechanisms.
- Create a structured process for analyzing feedback.
- Prioritize improvements based on user insights.
- Validate core value propositions with ICPs.

- Identify unexpected use cases and opportunities.

Success Metrics:

- Clear categorization of all feedback items.
- Implement granular LLMetry (LLM-based metrics) for autonomous parsing and escalation of user insights.
- Integrate thumbs up/down feedback in chat interface for model training.
- Utilize Hotjar or similar to monitor page activity.

Why: Agents reduce time, effort, and cost of collecting candid, true, and highly accurate user insights, delivering actionable strategy. 100% beta user feedback is unnecessary.

Penumbra: AI-driven reasoning platform for consultants.

Strategy:

- Make strategy computable by enabling AI to tap into structured knowledge
- turning fragmented insights into a dynamic knowledge system.

Initial Market:

- Consulting segment with USD 500M-2B addressable market.

Go-To-Market:

- Launch with a broadly useful, high-value solution to capture 1-2% market share quickly.

Growth Wedge:

- Layer additional features over time.

Vision:

- Go-to reasoning engine for independent consultants and boutique firms to outperform larger firms
- capture/structure knowledge faster, and deliver consistent insights at scale.

Future:

- Smarter, faster, and more dynamic consulting powered by Penumbra.

Sthesia PRD (Outdated)

Page

Sthesia PRD

Molecular Memory Preservation Platform

Executive Summary

Sthesia stands at the intersection of nature and technology, offering a revolutionary approach to preserving the purest essence of plants through proprietary molecular separation technology. This PRD outlines a comprehensive solution to enhance Sthesia's ability to capture, catalog, and share the true-to-type molecular signatures of plants while strengthening their commitment to biodiversity preservation and sustainable practices.

Our proposed Molecular Memory Preservation Platform (MMPP) will transform Sthesia's operations by creating a digital ecosystem that integrates with their physical molecular preservation work, enabling seamless data management, enhanced customer experiences, and expanded impact measurement of their conservation efforts.

This platform aligns perfectly with Sthesia's mission to be stewards of nature and to create harmony between technology and ecology, past and future, science and art, nature and civilization.

Understanding Sthesia: A Molecular Time Capsule of Biodiversity

Sthesia is pioneering a new way to experience nature by capturing the true molecular essence of plants exactly as they exist in their living state. Through proprietary cold concentration processes, Sthesia preserves plants in their most pristine molecular form, creating what they call "Molecular Concentrates®" - pure, plant-derived ingredients that deliver high-definition sensory experiences across fragrance, flavor, and functional applications.

Core Philosophy

At the heart of Sthesia lies a profound respect for the natural world and a commitment to preserving both its tangible and intangible values. The company believes that every plant speaks through a unique signature of molecules - a language that is often poorly translated through conventional extraction methods. Sthesia's technology aims to capture this molecular language with unprecedented fidelity, creating a perpetual time capsule of biodiversity's sensory experiences.

Conservation Commitment

Sthesia's business model inherently supports biodiversity conservation. The company has partnered with Savimbo and Indigenous farmers to protect biodiversity hotspots in the Amazon Rainforest, funding the protection of areas with documented rare, threatened, or endangered species. This demonstrates a holistic approach that values both the commercial and conservation aspects of plant molecular preservation.

Product Portfolio

Sthesia offers various forms of Molecular Concentrates including:

Pure Molecular Concentrates (oil-soluble)
Water Soluble Powdered Molecular Concentrates
Each product captures the unique essence of specific plant varieties with zero calories, zero sugar, and zero bitterness, while maintaining true-to-type flavor and fragrance profiles.

Key Challenges & Opportunities

Through our analysis, we've identified several core challenges that Sthesia faces, which present significant opportunities for innovation:

1. Molecular Memory Preservation

Challenge: Capturing, cataloging, and preserving molecular signatures across thousands of plant varieties, creating a comprehensive "library" of biodiversity's sensory profiles
Opportunity: Develop systematic digital tools for molecular fingerprinting and cataloging

1. Scaling Production While Maintaining Quality

Challenge: Maintaining the purity and quality of molecular extraction while scaling operations
Opportunity: Create systems for standardization and quality assurance across production batches

1. Connecting Conservation to Commercial Success

Challenge: Quantifying and communicating the biodiversity impact of operations
Opportunity: Build robust impact measurement frameworks that connect conservation outcomes to business growth

1. Customer Education and Engagement

Challenge: Educating customers about the unique value proposition of molecular concentrates
Opportunity: Create immersive experiences that demonstrate the superiority of true-to-type extracts

1. Data Management and Knowledge Transfer

Challenge: Managing the vast amount of data about plant varieties, molecular profiles, and applications
Opportunity: Develop a comprehensive knowledge management system that preserves institutional knowledge

Solution Vision: Molecular Memory Preservation Platform (MMPP)

We propose the development of a Molecular Memory Preservation Platform (MMPP) - an integrated digital ecosystem that complements Sthesia's physical molecular preservation work. This platform will transform how Sthesia captures, catalogs, manages, and shares the molecular signatures of plants while enhancing their conservation impact.

Core Platform Components

1. Molecular Library Database
    - Digital repository of all molecular concentrates with complete profiles
    - Searchable by plant species, geographic origin, molecular composition, sensory attributes, and applications
    - Integration with laboratory equipment for automated data capture
2. Conservation Impact Dashboard
    - Real-time tracking of protected land areas and species
    - Integration with Savimbo partnership data
    - Visualization of conservation outcomes linked to specific product purchases
3. Customer Experience Portal
    - Interactive interface for customers to explore products

- Educational resources about molecular preservation technology
- Custom formulation tools for B2B clients

4. Production Management System
    - Quality control protocols and tracking
    - Batch management and traceability
    - Inventory optimization
5. Indigenous Knowledge Integration Module
    - Ethical documentation of traditional plant knowledge (with appropriate permissions)
    - Recognition and attribution system for Indigenous contributions
    - Revenue sharing tracking for conservation partnerships

3-Phase Implementation Approach

Phase 1: CRAWL - Molecular Library Foundation (3-6 months)

Focus on creating immediate value through enhanced data management and customer experience with minimal disruption to existing operations.

Key Deliverables:

1. Digital Molecular Profile Builder
    - Web-based tool to capture and organize molecular concentrate data
    - Integration with existing Google Workspace for seamless adoption
    - Basic version of the searchable molecular library
2. Customer Sample Management Portal
    - Sample request and tracking system
    - Digital repository of product information and applications
    - Analytics dashboard for sample conversion tracking
3. Conservation Impact Visualizer
    - Basic dashboard showing Savimbo partnership outcomes
    - Connection between product sales and conservation impact
    - Shareable impact reports for customers

Expected Outcomes:

- 40% reduction in time spent on data management
- 30% increase in sample request conversions
- Compelling impact stories for marketing and customer engagement

"Holy Shit" Moment:

When Sthesia sees how their molecular data can be visualized and explored in ways they never imagined, revealing patterns and connections across their sample library that were previously invisible.

Phase 2: WALK - Sensory Experience Enhancement (6-12 months)

Expand the platform to create richer experiences for customers and deeper integration with production processes.

Key Deliverables:

1. Molecular Fingerprinting System
    - Advanced analytical tools for comparing and categorizing molecular profiles
    - Pattern recognition algorithms to identify unique molecular signatures
    - Integration with laboratory equipment for automated data capture
2. Interactive Application Explorer
    - Digital environment for customers to virtually experiment with molecular concentrates
    - Industry-specific application suggestions (food & beverage, cosmetics, etc.)
    - Formulation tools for testing molecular combinations
3. Production Quality Management System
    - Standardized quality control protocols
    - Batch tracking and certification
    - Anomaly detection algorithms for quality assurance

Expected Outcomes:

- 50% faster product development cycles
- 25% increase in customer application discovery
- 35% reduction in quality control issues

Phase 3: RUN - Full Ecosystem Integration (12-24 months)

Create a comprehensive platform that positions Sthesia as the global authority on molecular preservation of plant biodiversity.

Key Deliverables:

1. Global Molecular Atlas
    - Geospatial mapping of plant varieties and their molecular profiles
    - Temporal tracking of changes in molecular signatures over time
    - Research portal for scientific collaboration
2. Predictive Formulation Engine
    - AI-powered suggestion system for new product development
    - Customer preference modeling for personalized recommendations
    - Cross-industry application discovery
3. Biodiversity Conservation Marketplace
    - Direct connection between product purchases and conservation initiatives
    - Expanded partnerships with Indigenous communities
    - Transparent impact tracking and reporting
4. Knowledge Transfer Platform

- Documentation of extraction techniques and methodologies
- Training modules for new team members
- Preservation of institutional knowledge

Expected Outcomes:

- Establishment as the definitive authority on plant molecular preservation
- Creation of an entirely new category in the natural ingredients industry
- Measurable expansion of biodiversity conservation impact
- Scalable growth model that maintains quality while increasing reach

Technical Architecture

The MMPP will be built using a modular, cloud-based architecture that integrates with Sthesia's existing Google Workspace and Coda environments. Key technical components include:

1. Core Database
   - Cloud-based molecular profile repository
   - Secure storage for proprietary extraction methods
   - Structured data model for plant varieties and their properties
2. Integration Layer
   - APIs for laboratory equipment connections
   - Seamless integration with Google Workspace
   - Advanced Coda integration for workflow management
3. User Experience Layer
   - Web-based interfaces for internal users
   - Customer-facing portals with appropriate access controls
   - Mobile compatibility for field use
4. Analytics Engine
   - Real-time reporting dashboards
   - Predictive modeling capabilities
   - Conservation impact tracking

Implementation Considerations

Resource Requirements

- Technology: Cloud infrastructure, database development, custom integration work
- Personnel: Data scientists, UX designers, conservation specialists
- Partners: Laboratory equipment integrations, conservation verification

Key Success Factors

1. Cross-functional buy-in across Sthesia organization
2. Phased approach that delivers value at each stage
3. Balance between technical sophistication and usability

4. Respect for Indigenous knowledge and contributions
5. Alignment with cold concentration process workflows

Risk Management

1. Data Security: Implement robust encryption and access controls to protect proprietary molecular profiles
2. Adoption Challenges: Ensure intuitive design and adequate training for all users
3. Conservation Verification: Establish rigorous protocols for validating conservation outcomes
4. Integration Complexity: Start with minimal viable integrations and expand progressively

Expected Business Impact

The Molecular Memory Preservation Platform will transform Sthesia's operations in multiple dimensions:

1. Accelerated Innovation
    - 30-50% faster product development cycles
    - Discovery of unexpected molecular combinations and applications
    - Preservation of knowledge for sustained innovation
2. Enhanced Customer Value
    - More targeted product recommendations
    - Deeper education about molecular preservation
    - Stronger emotional connection to conservation outcomes
3. Operational Excellence
    - Standardized quality across production
    - Streamlined data management
    - Efficient workflow optimization
4. Conservation Leadership
    - Quantifiable, verifiable impact on biodiversity
    - Strengthened partnerships with Indigenous communities
    - Creation of a model for sustainable business practices
5. Market Differentiation
    - Establishment as the definitive authority on molecular preservation
    - Creation of barriers to entry through proprietary data
    - Premium positioning supported by authentic conservation commitment

Conclusion

The Molecular Memory Preservation Platform represents a natural evolution of Sthesia's mission to preserve the molecular language of plants. By creating a digital ecosystem that complements their physical preservation work, Sthesia will not only enhance their operational capabilities but also strengthen their position as stewards of nature's molecular diversity.

This platform honors the harmony between technology and ecology that lies at the heart of Sthesia's philosophy. By implementing this solution in three carefully designed phases, Sthesia can transform how they capture, preserve, and share the true essence of plants while making a meaningful impact on biodiversity conservation.

We believe this approach will resonate deeply with Sthesia's vision and values, creating immediate value while setting the foundation for long-term leadership in both molecular preservation and conservation.

Appendix: Glossary of Terms

- Molecular Concentrates®: Sthesia's proprietary plant-derived ingredients that capture the true-to-type essence of plants
- Cold Concentration Process: Sthesia's proprietary extraction method that preserves molecules in their most pristine form
- MMPP: Molecular Memory Preservation Platform
- Molecular Fingerprinting: The process of identifying and cataloging the unique molecular signature of a plant
- Conservation Impact: The measurable effect of Sthesia's operations on biodiversity preservation

💡 TL;DR:

📣 Slack: #slack-channel

🖼️ Designs: [Whimsical] [Figma]

🏙️ Dashboard: Can be found [add link]

▼ Sthesia Huddle

▼ Penumbra Database Schema Conceptualization No. 1

A database schema for Penumbra

- It is needed in order to model the interconnected nature of organizational knowledge
- and the pattern recognition capabilities.

Here's how we could structure it:

- This still needs
  - Input Validation
  - Constraints (e.g., NOT NULL)

▼ Core Tables

`users`

- `user_id` (PK)
- `name`
- `email`
- `role`
- `team_id` (FK)
- `preferences_json`

`teams`

- `team_id` (PK)
- `name`
- `workspace_settings_json`

`content_sources`

- `source_id` (PK)
- `source_type` (enum: 'document', 'conversation', 'task', 'metric', etc.)
- `source_platform` (e.g., 'notion', 'slack', 'jira', 'google_docs')
- `connection_config_json`
- `last_sync_timestamp`

Content & Knowledge Tables

`knowledge_nodes`

- `node_id` (PK)
- `source_id` (FK)
- `content_hash`
- `content_text`
- `metadata_json`
- `vector_embedding` (for semantic search)
- `created_at`
- `modified_at`

`relationships`

- `relationship_id` (PK)
- `source_node_id` (FK)
- `target_node_id` (FK)
- `relationship_type` (e.g., 'references', 'contradicts', 'supports')
- `confidence_score`
- `created_by` (algorithm or user_id)
- `created_at`

patterns

- pattern_id (PK)
- pattern_name
- pattern_description
- detection_algorithm
- confidence_threshold
- created_at
- modified_at

pattern_instances

- instance_id (PK)
- pattern_id (FK)
- affected_nodes_json (array of node_ids)
- context_json
- detected_at
- relevance_score

User Interaction Tables

insights

- insight_id (PK)
- title
- description
- related_pattern_id (FK)
- supporting_evidence_json
- generated_at
- modified_at
- status (enum: 'new', 'acknowledged', 'implemented', 'dismissed')

user_interactions

- interaction_id (PK)
- user_id (FK)
- interaction_type (e.g., 'viewed_insight', 'queried_knowledge', 'connected_nodes')
- interaction_details_json
- timestamp

decisions

- `decision_id` (PK)
- `title`
- `description`
- `related_insight_ids_json`
- `supporting_knowledge_nodes_json`
- `made_by_user_id` (FK)
- `made_at`
- `outcome_tracking_json`

This schema design supports Penumbra's core functionality by:

1. Capturing diverse content from various sources
2. Maintaining relationships between pieces of information
3. Storing detected patterns and insights
4. Tracking how users interact with the system and make decisions
5. Supporting the "meta-assistant" concept through interconnected data structures

The vector embeddings field enables semantic search and pattern recognition across the knowledge base, allowing the system to function as the "coach" that sees connections across all organizational knowledge.

-- Users and Teams

```sql
CREATE TABLE teams (
team_id SERIAL PRIMARY KEY,
name VARCHAR(255) NOT NULL,
workspace_settings JSONB DEFAULT '{}'
);
CREATE TABLE users (
user_id SERIAL PRIMARY KEY,
name VARCHAR(255) NOT NULL,
email VARCHAR(255) UNIQUE NOT NULL,
role VARCHAR(100),
team_id INTEGER REFERENCES teams(team_id),
preferences JSONB DEFAULT '{}',
created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

--Content Sources

```sql
CREATE TYPE source_type AS ENUM ('document', 'conversation', 'task',
'metric', 'email', 'meeting');

CREATE TABLE content_sources (
    source_id SERIAL PRIMARY KEY,
    source_type source_type NOT NULL,
    source_platform VARCHAR(100) NOT NULL,
    connection_config JSONB NOT NULL,
    last_sync_timestamp TIMESTAMP WITH TIME ZONE,
    team_id INTEGER REFERENCES teams(team_id),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);
```

-- Knowledge Storage

SQL

```sql
CREATE TABLE knowledge_nodes (
    node_id SERIAL PRIMARY KEY,
    source_id INTEGER REFERENCES content_sources(source_id),
    content_hash VARCHAR(64) NOT NULL,
    content_text TEXT,
    metadata JSONB DEFAULT '{}',
    vector_embedding vector(1536), -- Using pgvector extension
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    modified_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

CREATE INDEX idx_knowledge_nodes_vector ON knowledge_nodes USING ivfflat
(vector_embedding vector_cosine_ops);

CREATE TYPE relationship_type AS ENUM ('references', 'contradicts',
'supports', 'follows', 'precedes', 'similar');

CREATE TABLE relationships (
    relationship_id SERIAL PRIMARY KEY,
    source_node_id INTEGER REFERENCES knowledge_nodes(node_id),
    target_node_id INTEGER REFERENCES knowledge_nodes(node_id),
    relationship_type relationship_type NOT NULL,
    confidence_score FLOAT NOT NULL,
    created_by VARCHAR(255), -- Can be user_id or algorithm name
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT unique_relationship UNIQUE(source_node_id, target_node_id,
relationship_type)
);
```

-- Pattern Recognition

SQL

```sql
CREATE TABLE patterns (
    pattern_id SERIAL PRIMARY KEY,
    pattern_name VARCHAR(255) NOT NULL,
    pattern_description TEXT,
    detection_algorithm VARCHAR(100) NOT NULL,
    confidence_threshold FLOAT DEFAULT 0.7,
    team_id INTEGER REFERENCES teams(team_id),
    created_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    modified_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE pattern_instances (
    instance_id SERIAL PRIMARY KEY,
    pattern_id INTEGER REFERENCES patterns(pattern_id),
    affected_nodes JSONB NOT NULL, -- Array of node_ids
    context JSONB DEFAULT '{}',
    detected_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    relevance_score FLOAT NOT NULL
);
```

-- User Interactions & Insights

SQL

```sql
CREATE TYPE insight_status AS ENUM ('new', 'acknowledged', 'implemented',
'dismissed');

CREATE TABLE insights (
    insight_id SERIAL PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    description TEXT,
    related_pattern_id INTEGER REFERENCES patterns(pattern_id),
    supporting_evidence JSONB NOT NULL,
    team_id INTEGER REFERENCES teams(team_id),
    generated_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    modified_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    status insight_status DEFAULT 'new'
);

CREATE TABLE user_interactions (
    interaction_id SERIAL PRIMARY KEY,
    user_id INTEGER REFERENCES users(user_id),
    interaction_type VARCHAR(100) NOT NULL,
    interaction_details JSONB DEFAULT '{}',
    timestamp TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP
);

CREATE TABLE decisions (
    decision_id SERIAL PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    description TEXT,
    related_insight_ids JSONB, -- Array of insight_ids
    supporting_knowledge_nodes JSONB, -- Array of node_ids
    made_by_user_id INTEGER REFERENCES users(user_id),
    team_id INTEGER REFERENCES teams(team_id),
    made_at TIMESTAMP WITH TIME ZONE DEFAULT CURRENT_TIMESTAMP,
    outcome_tracking JSONB DEFAULT '{}'
);
```

Indexes

SQL

```sql
-- Add appropriate indexes
CREATE INDEX idx_knowledge_nodes_content_hash ON
knowledge_nodes(content_hash);
CREATE INDEX idx_knowledge_nodes_source_id ON knowledge_nodes(source_id);
CREATE INDEX idx_relationships_source_node ON
relationships(source_node_id);
CREATE INDEX idx_relationships_target_node ON
relationships(target_node_id);
CREATE INDEX idx_pattern_instances_pattern_id ON
pattern_instances(pattern_id);
CREATE INDEX idx_insights_team_id ON insights(team_id);
CREATE INDEX idx_decisions_team_id ON decisions(team_id);
```

Best DBs

Page

Best Database Options for Penumbra

For a system like Penumbra that combines knowledge management, pattern recognition, and relationship tracking across diverse data sources, you should consider these database options:

PostgreSQL with Extensions

Best overall option for Penumbra due to its balance of features:

- Strong relational capabilities for structured team/user data
- JSONB support for flexible metadata and semi-structured content
- pgvector extension for vector embeddings and semantic search
- Rich query capabilities and transaction support
- Excellent indexing options for performance optimization
- Mature ecosystem with extensions for specialized needs

Hybrid Approach

For production-grade implementation, consider a multi-database architecture:

1. **PostgreSQL** as the primary database:
   - Core relational data (users, teams, content sources)
   - Relationship tracking
   - Transaction handling
2. **Vector Database** (Pinecone, Weaviate, Qdrant):
   - Specialized for vector embeddings
   - Optimized for semantic search and similarity operations
   - Better performance for AI-related queries at scale
3. **Graph Database** (Neo4j):
   - Optimized for complex relationship queries
   - Pattern detection across interconnected knowledge
   - Visualization of knowledge networks
4. **Document Store** (MongoDB):
   - Raw content storage
   - Flexible schema for diverse content types
   - Efficient for storing large amounts of unstructured data

Scaling Considerations

As Penumbra grows:

- Start with PostgreSQL + pgvector for simplicity
- Introduce specialized databases as specific needs become bottlenecks
- Use message queues (Kafka/RabbitMQ) to synchronize between database systems
- Consider read replicas for analytics workloads

The ideal architecture would likely evolve from a PostgreSQL-only solution to a specialized hybrid approach as the product matures and usage patterns become clearer.

Specialized DBs

Page

Specialized Databases for Penumbra

Vector Databases

Pinecone

- **Strengths**: Fully managed, purpose-built for vector search, excellent scalability
- **Use case**: Storing and querying embeddings of knowledge nodes for semantic similarity
- **Fit for Penumbra**: Excellent for the "pattern recognition" aspect, enabling quick retrieval of conceptually similar content across different sources
- **Limitations**: Not suitable for relational data or primary storage; would complement PostgreSQL

Qdrant

- **Strengths**: Open-source, filtering capabilities with vector search, flexible deployment
- **Use case**: Combining metadata filtering with semantic search
- **Fit for Penumbra**: Good for filtering pattern instances by team, project, or time while maintaining semantic relevance
- **Limitations**: Less mature ecosystem than some alternatives; requires more operational expertise

Graph Databases

Neo4j

- **Strengths**: Industry-leading graph database, powerful query language (Cypher)
- **Use case**: Modeling complex relationships between knowledge nodes, users, and insights
- **Fit for Penumbra**: Ideal for the "coach" metaphor, detecting patterns across interconnected information
- **Limitations**: Less efficient for vector operations; steep learning curve for query optimization

ArangoDB

- **Strengths**: Multi-model database supporting graphs, documents, and key-value storage
- **Use case**: Flexible knowledge representation with graph traversal capabilities
- **Fit for Penumbra**: Good compromise if you want graph capabilities without multiple databases
- **Limitations**: Jack-of-all-trades approach may not excel at vector search compared to specialized options

Cloud Native Options

AstraDB (DataStax)

- **Strengths**: Cassandra-based with vector search capabilities, serverless scaling
- **Use case**: High-throughput vector operations with global distribution
- **Fit for Penumbra**: Good if you need global scale and low-latency vector operations
- **Limitations**: Less mature for complex graph queries compared to Neo4j

Azure Cosmos DB

- **Strengths**: Multi-model database with global distribution
- **Use case**: Enterprise deployments requiring multiple data models
- **Fit for Penumbra**: Good if you're already invested in Azure ecosystem
- **Limitations**: Can become expensive at scale; vector capabilities less mature than specialized options

Recommendation for Penumbra

The ideal architecture would likely be:

1. **Core Data Layer**: PostgreSQL for users, teams, and structured operational data
2. **Vector Layer**: Pinecone or Qdrant for semantic search and pattern matching
3. **Graph Layer**: Neo4j for relationship analysis and knowledge graph traversal

This combination would support Penumbra's unique value proposition:

- The "film room for your brain" requires semantic understanding (vector DB)
- The "coach seeing patterns" needs relationship analysis (graph DB)
- The organizational structure and metadata need relational capabilities (PostgreSQL)

As you scale, you could start with PostgreSQL + pgvector, then add specialized databases as usage patterns and performance needs become clearer.

Neon & Turso

Page

Neon and Turso for Penumbra

Neon

Overview: Serverless PostgreSQL with a modern architecture separating storage and compute.

Strengths for Penumbra:

- Built on PostgreSQL, so compatible with all extensions including pgvector
- Serverless architecture with auto-scaling compute
- Branch-based development model (similar to Git) for testing schema changes
- Point-in-time recovery and unlimited storage
- Pay only for what you use

Limitations:

- Relatively new product (launched 2022)
- Some PostgreSQL features may have implementation differences
- Upper limits on connection pooling might affect high-concurrency scenarios

Fit for Penumbra: Excellent choice if we want PostgreSQL capabilities with modern cloud-native features. The branching feature would be particularly valuable during development of Penumbra's schema as it evolves.

Turso

Overview: Distributed SQLite database built on libSQL, designed for edge deployment.

Strengths for Penumbra:

- Extremely low latency with edge-first architecture
- Embedded database design with synchronization
- Very cost-effective for read-heavy workloads
- Simple operational model

Limitations:

- Limited support for advanced PostgreSQL features
- No native vector embedding support (would need application-level implementation)
- Less suitable for complex analytical queries
- Not designed for heavy write workloads

Fit for Penumbra: Better suited for client-side caching of Penumbra data rather than as the primary database. Could be valuable for edge deployments where we want to bring Penumbra's insights closer to users with low latency.

Recommendation for Penumbra

1. **Neon** would be an excellent choice for the core PostgreSQL database needs:
   - Use it as our primary relational store
   - Leverage pgvector extension for initial vector embedding needs
   - Benefit from the branching model during development
   - Scale compute resources based on actual usage
2. **Consider specialized databases alongside Neon**:
   - As vector embedding needs grow beyond what pgvector can handle, add Pinecone or Qdrant
   - For complex relationship analysis beyond what SQL can efficiently query, add Neo4j
3. **Turso could complement this architecture**:
   - Deploy Turso at the edge for caching frequently accessed Penumbra insights
   - Use for low-latency read access to personalized user data
   - Synchronize with the core Neon database

This approach gives us the PostgreSQL compatibility and features needed for Penumbra's core functionality while leveraging the serverless benefits of Neon's modern architecture, with room to add specialized databases as our needs evolve.

DB Concpetualization No. 2

Page

DB No. 2

1.4 4. Knowledge Representation (Nodes & Relationships)

1.5 `1.5 knowledge_nodes` 1.5

1.6 `1.6 asset_keywords` 1.6

1.7 `1.7 node_keywords` 1.7

1.8 `1.8 relationships` 1.8

1.9 1.9 4. AI & Pattern Recognition1.9 Patterns

1.10 `1.10 patterns` 1.10

1.11 `1.11 pattern_instances` 1.11

1.12 `1.12 insights` 1.12

2 5. User Interaction & Application State

2.1 `2.1 chat_sessions` 2.1

2.2 `2.2 chat_messages` 2.2

2.3 `2.3 user_interactions` 2.3

2.4 `2.4 decisions` 2.4

2.5 `2.5 work_sessions` 2.5

2.6 work_topics

2.7 timeline_events

2.8 `2.8 focus_areas` 2.8

Here's a refined and consolidated database schema proposal for Penumbra in Supabase (PostgreSQL).

This builds upon the existing conceptualization while ensuring proper keys, relationships, and support for core features.

Penumbra Database Schema for Supabase (PostgreSQL)

Core Entities & Tables:

- Users & Access Control
- Workspaces & Teams
- Content & Knowledge Ingestion
- Knowledge Representation (Nodes & Relationships)
- AI & Pattern Recognition
- User Interaction & Application State

1. Users & Access Control

`users`

- user_id (UUID, PK, default: uuid_generate_v4() or auth.uid())
- email (TEXT, UNIQUE, NOT NULL)
- full_name (TEXT)
- avatar_url (TEXT)
- created_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- updated_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- preferences_json (JSONB)

api_keys

- api_key_id (UUID, PK, default: uuid_generate_v4())
- user_id (UUID, FK, references users(user_id), NOT NULL)
- key_hash (TEXT, UNIQUE, NOT NULL)
- key_name (TEXT, NOT NULL)
- created_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- last_used_at (TIMESTAMP WITH TIME ZONE)
- expires_at (TIMESTAMP WITH TIME ZONE)
- is_active (BOOLEAN, default: TRUE)

2. Workspaces & Teams

teams

- team_id (UUID, PK, default: uuid_generate_v4())
- team_name (TEXT, NOT NULL)
- owner_user_id (UUID, FK, references users(user_id), NOT NULL)
- workspace_settings_json (JSONB)
- created_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- updated_at (TIMESTAMP WITH TIME ZONE, default: NOW())

team_members

- team_member_id (UUID, PK, default: uuid_generate_v4())
- team_id (UUID, FK, references teams(team_id) ON DELETE CASCADE, NOT NULL)
- user_id (UUID, FK, references users(user_id) ON DELETE CASCADE, NOT NULL)
- role (TEXT, default: 'member')
- joined_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- UNIQUE (team_id, user_id)

3. Content & Knowledge Ingestion

content_sources

- source_id (UUID, PK, default: uuid_generate_v4())
- team_id (UUID, FK, references teams(team_id) ON DELETE CASCADE, NOT NULL)
- source_type (TEXT, NOT NULL)
- source_platform (TEXT)
- source_name (TEXT, NOT NULL)
- connection_config_json (JSONB)
- last_sync_timestamp (TIMESTAMP WITH TIME ZONE)
- sync_status (TEXT, default: 'pending')
- created_by_user_id (UUID, FK, references users(user_id), NOT NULL)1
- created_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- updated_at (TIMESTAMP WITH TIME ZONE, default: NOW())

knowledge_assets

- asset_id (UUID, PK, default: uuid_generate_v4())
- source_id (UUID, FK, references content_sources(source_id) ON DELETE SET NULL)
- team_id (UUID, FK, references teams(team_id) ON DELETE CASCADE, NOT NULL)
- asset_type (TEXT, NOT NULL)
- file_name (TEXT)
- content_hash (TEXT, INDEX)
- raw_content_url (TEXT)
- processed_text_content (TEXT)
- metadata_json (JSONB)
- status (TEXT, default: 'pending_processing')
- created_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- updated_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- uploaded_by_user_id (UUID, FK, references users(user_id), NOT NULL)

4. Knowledge Representation (Nodes & Relationships)

knowledge_nodes

- node_id (UUID, PK, default: uuid_generate_v4())
- team_id (UUID, FK, references teams(team_id) ON DELETE CASCADE, NOT NULL)
- asset_id (UUID, FK, references knowledge_assets(asset_id) ON DELETE SET NULL)
- node_type (TEXT, NOT NULL)
- content_text (TEXT)
- metadata_json (JSONB)
- vector_embedding (vector(1536))
- created_by_user_id (UUID, FK, references users(user_id))

- created_by_agent_id (TEXT)
- created_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- updated_at (TIMESTAMP WITH TIME ZONE, default: NOW())

asset_keywords

- asset_keyword_id (UUID, PK, default: uuid_generate_v4())
- asset_id (UUID, FK, references knowledge_assets(asset_id) ON DELETE CASCADE, NOT NULL)
- keyword (TEXT, NOT NULL)
- relevance_score (FLOAT)
- UNIQUE (asset_id, keyword)

node_keywords

- node_keyword_id (UUID, PK, default: uuid_generate_v4())
- node_id (UUID, FK, references knowledge_nodes(node_id) ON DELETE CASCADE, NOT NULL)
- keyword (TEXT, NOT NULL)
- relevance_score (FLOAT)
- UNIQUE (node_id, keyword)

relationships

- relationship_id (UUID, PK, default: uuid_generate_v4())
- team_id (UUID, FK, references teams(team_id) ON DELETE CASCADE, NOT NULL)
- source_node_id (UUID, FK, references knowledge_nodes(node_id) ON DELETE CASCADE, NOT NULL)
- target_node_id (UUID, FK, references knowledge_nodes(node_id) ON DELETE CASCADE, NOT NULL)
- relationship_type (TEXT, NOT NULL)
- description (TEXT)
- confidence_score (FLOAT)
- metadata_json (JSONB)
- created_by_user_id (UUID, FK, references users(user_id))
- created_by_agent_id (TEXT)
- created_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- updated_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- UNIQUE (source_node_id, target_node_id, relationship_type)

4. AI & Pattern Recognition Patterns

patterns

- pattern_id (UUID, PK, default: uuid_generate_v4())
- team_id (UUID, FK, references teams(team_id) ON DELETE CASCADE, NOT NULL)
- pattern_name (TEXT, NOT NULL)
- pattern_description (TEXT)
- detection_algorithm (TEXT)
- definition_json (JSONB)
- created_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- updated_at (TIMESTAMP WITH TIME ZONE, default: NOW())

pattern_instances

- instance_id (UUID, PK, default: uuid_generate_v4())
- pattern_id (UUID, FK, references patterns(pattern_id) ON DELETE CASCADE, NOT NULL)
- team_id (UUID, FK, references teams(team_id) ON DELETE CASCADE, NOT NULL)
- affected_nodes_json (JSONB)
- context_json (JSONB)
- relevance_score (FLOAT)
- status (TEXT, default: 'active')
- detected_at (TIMESTAMP WITH TIME ZONE, default: NOW())

insights

- insight_id (UUID, PK, default: uuid_generate_v4())
- team_id (UUID, FK, references teams(team_id) ON DELETE CASCADE, NOT NULL)
- title (TEXT, NOT NULL)
- description (TEXT)
- related_pattern_instance_id (UUID, FK, references pattern_instances(instance_id) ON DELETE SET NULL)
- supporting_evidence_json (JSONB)
- generated_by_agent_id (TEXT)
- status (TEXT, default: 'new')
- generated_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- updated_at (TIMESTAMP WITH TIME ZONE, default: NOW())

5. User Interaction & Application State

chat_sessions

- session_id (UUID, PK, default: uuid_generate_v4())

- team_id (UUID, FK, references teams(team_id) ON DELETE CASCADE, NOT NULL)
- user_id (UUID, FK, references users(user_id) ON DELETE CASCADE, NOT NULL)
- session_title (TEXT)
- created_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- updated_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- archived (BOOLEAN, default: FALSE)
- mcp_session_id (TEXT, UNIQUE)

chat_messages

- message_id (UUID, PK, default: uuid_generate_v4())
- session_id (UUID, FK, references chat_sessions(session_id) ON DELETE CASCADE, NOT NULL)
- sender_user_id (UUID, FK, references users(user_id))
- sender_agent_id (TEXT)
- message_content (TEXT, NOT NULL)
- message_type (TEXT, default: 'text')
- metadata_json (JSONB)
- timestamp (TIMESTAMP WITH TIME ZONE, default: NOW())
- parent_message_id (UUID, FK, references chat_messages(message_id))

user_interactions

- interaction_id (UUID, PK, default: uuid_generate_v4())
- user_id (UUID, FK, references users(user_id) ON DELETE CASCADE, NOT NULL)
- team_id (UUID, FK, references teams(team_id) ON DELETE CASCADE, NOT NULL)
- interaction_type (TEXT, NOT NULL)
- target_id (UUID)
- target_type (TEXT)
- interaction_details_json (JSONB)
- timestamp (TIMESTAMP WITH TIME ZONE, default: NOW())

decisions

- decision_id (UUID, PK, default: uuid_generate_v4())
- team_id (UUID, FK, references teams(team_id) ON DELETE CASCADE, NOT NULL)
- title (TEXT, NOT NULL)
- description (TEXT)
- related_insight_ids_json (JSONB)
- supporting_knowledge_nodes_json (JSONB)
- made_by_user_id (UUID, FK, references users(user_id), NOT NULL)

- status (TEXT, default: 'proposed')
- made_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- outcome_tracking_json (JSONB)

work_sessions

- mcp_work_session_id (UUID, PK, default: uuid_generate_v4())
- team_id (UUID, FK, references teams(team_id), NOT NULL)
- user_id (UUID, FK, references users(user_id), NOT NULL)
- initial_query (TEXT)
- context_scope (TEXT)
- focus_hint (TEXT)
- status (TEXT, default: 'active')
- created_at (TIMESTAMP WITH TIME ZONE, default: NOW())
- finalized_at (TIMESTAMP WITH TIME ZONE)

work_topics

- topic_id (UUID, PK, default: uuid_generate_v4())
- mcp_work_session_id (UUID, FK, references work_sessions(mcp_work_session_id))
- chat_session_id (UUID, FK, references chat_sessions(session_id))
- topic_name (TEXT, NOT NULL)
- description (TEXT)

timeline_events

- event_id (UUID, PK, default: uuid_generate_v4())
- mcp_work_session_id (UUID, FK, references work_sessions(mcp_work_session_id))
- event_type (TEXT, NOT NULL)
- description (TEXT)
- related_asset_ids_json (JSONB)
- impact_assessment (TEXT)
- timestamp (TIMESTAMP WITH TIME ZONE, default: NOW())

focus_areas

- focus_area_id (UUID, PK, default: uuid_generate_v4())
- mcp_work_session_id (UUID, FK, references work_sessions(mcp_work_session_id))
- user_id (UUID, FK, references users(user_id))
- focus_area_name (TEXT, NOT NULL)
- relevance_score (FLOAT)
- last_updated (TIMESTAMP WITH TIME ZONE, default: NOW())

KG

Page

Think Hard

The AI-Human Tightrope: Defining Our New Work Model - Challenging Our Thinking

https://www.sequoiacap.com/article/pmf-framework-2/?itm_medium=related-content&itm_source=sequoiacap.com - Borrow this framework to help address terrifying things that question your beliefs. Put it under the fire so we can bake something really good for customers that they LOVE!

Core Questions

1. Purpose & Resonance
    - What fundamental human needs (creativity, control, speed) should this model serve?
2. Cognitive Engagement
    - Are users hungry to flex their domain-expert brains, or do they prefer AI to take the wheel?
3. Trust & Verification
    - When AI suggests an answer, do users instinctively fact-check it, nod along, or reject it outright?
4. Interaction Metaphor
    - Is this a co-pilot scenario (shared steering) or autopilot (hands-off) for knowledge work?
5. Balance of Expertise
    - Where's the sweet spot between tapping user expertise and relying on AI inference?
6. Uncomfortable Truths
    - What hard questions are we avoiding—biases, blind spots, or "what if AI gets it wrong"?
        - Biases:
            - Why is Penumbra and the knowledge graph approach better than just reusing good prompts to get a certain output? → Penumbra graph > Good Prompts?
            - Why do users need a graph within the UI? Whats its purpose? What pain point is it addressing and what frame is it helping with, how and why? → Do we need a graph?
        - Blind Spots:
            - Will we get steamrolled if someone pivots? why or why not? how so? Define our moat and differentiation bets ...
7. Adoption & Buy-In

- Which features make users say "this speaks my language" versus "this feels alien"?

8. Feedback & Adaptation
   - How does the system learn from user pushback or moments of confusion?
9. Defining "The New Model"
   - What are its non-negotiable pillars (e.g., transparency, control, speed, personalization)?
   - Definition of the new model
10. Success Metrics
    - How will we measure true resonance—time saved, trust scores, or user-initiated edits?

Penumbra graph > Good Prompts?

Do we need a graph?

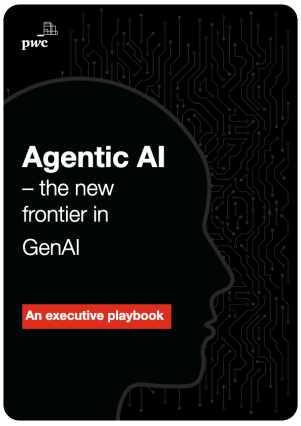
RAG

Arxiv STORM LangChain

Untitled
https://arxiv.org/pdf/2402.14207

PwC Agentic AI

AI Playbook



PDF

Agentic AI Executive Playbook

Source	upload
Number of Pages	22
File size	1417017
MIME type	application/pdf

Webpage Summary

Page

Agentic AI Advancements and Applications Summary

Agentic AI Advancements and Applications Summary

This webpage from marktechpost.com covers recent developments in agentic AI systems and related technologies. Key highlights include:

Agentic AI Architecture & Capabilities

- Designed to orchestrate multiple AI agents as a cohesive "swarm" using LangGraph framework
- Key features include:
  - Autonomy in decision-making
  - Goal-driven behavior aligned with organizational outcomes
  - Environmental interaction for real-time adaptation
  - Learning capabilities through reinforcement and historical data
  - Workflow orchestration across complex business functions
  - Multi-agent communication for coordinated actions

Advantages Over Traditional AI

- Surpasses traditional chatbots and RAG-based systems by:
  - Maintaining dialogue memory
  - Reasoning across multiple systems (CRM, ERP, IVR)
  - Dynamically solving customer issues

- PwC envisions micro-agents optimized for specific tasks coordinated by a central orchestrator

Featured Research & Developments

- DanceGRPO: A unified framework for reinforcement learning in visual generation
- ByteDance's Seed1.5-VL: A vision-language foundation model for multimodal understanding
- Hugging Face's free Model Context Protocol (MCP) course for developers

Practical Applications

- JPMorgan Chase automated legal document analysis via COiN platform, saving 360,000+ manual review hours annually
- The webpage also promotes miniCON 2025, an Agentic AI Event with free registration

This content reflects the growing importance of orchestrated, multi-agent AI systems capable of handling complex business processes with human-like intelligence and accountability.

Security

SOP

GitHub Repo SECURITY

Page

Repo InfoSec & DevSecOps

GitHub Repository Security Best Practices

Based on InfoSec and DevSecOps for GitHub repositories, this doc provides a comprehensive list of best practices for GitHub repository maintainers.

These standards will help secure us projects like Penumbra, and other repositories we're developing.

Access Control & Repository Configuration

1. Implement Branch Protection Rules

- Require pull request reviews before merging
- Require status checks to pass before merging
- Require signed commits
- Restrict who can push to matching branches

2. Enable Two-Factor Authentication (2FA)

- Require 2FA for all organization members
- Consider hardware security keys for critical repositories

3. Use Fine-Grained Access Control

- Limit admin access to essential personnel only

- Apply the principle of least privilege
- Regularly audit access permissions

4. Set Up Security Policies

- Create a SECURITY.md file with vulnerability reporting guidelines
- Define a clear process for security issue disclosure

Code Security

1. Implement Code Scanning

- Enable GitHub Advanced Security (if available)
- Configure CodeQL analysis for automated vulnerability detection
- Set up custom code scanning alerts

2. Secret Management

- Use GitHub Secrets for storing sensitive information
- Implement pre-commit hooks to prevent secrets from being committed
- Use tools like GitGuardian or Gitleaks to scan for exposed secrets

3. Dependency Management

- Enable Dependabot alerts and security updates
- Configure dependency review for pull requests
- Regularly update dependencies

4. Secure Coding Practices

- Establish coding standards that emphasize security
- Document secure patterns for common operations
- Create security-focused code review checklists

CI/CD Pipeline Security

1. Secure GitHub Actions

- Pin actions to specific SHA hashes instead of tags
- Use trusted actions from the marketplace or create your own
- Limit permissions for GitHub Actions workflows

2. Implement Security Testing in CI/CD

- SAST (Static Application Security Testing)
- DAST (Dynamic Application Security Testing)
- SCA (Software Composition Analysis)
- Container scanning for Docker images

3. Deploy with Least Privilege

- Use dedicated deployment credentials
- Rotate deployment keys regularly
- Implement time-limited access tokens

Compliance & Documentation

1. License Compliance

- Include appropriate LICENSE files
- Ensure third-party dependencies comply with your license
- Document license requirements

2. Security Documentation

- Maintain up-to-date security documentation
- Document security controls and their implementations
- Create incident response procedures

3. Compliance Standards

- Implement controls for relevant standards (GDPR, SOC 2, etc.)
- Document compliance measures

Monitoring & Incident Response

1. Security Monitoring

- Set up alerts for suspicious activities
- Monitor for unusual commit patterns
- Track and analyze security events

2. Vulnerability Management

- Establish a process for addressing security vulnerabilities
- Define severity levels and response times

- Track remediation progress

3. Incident Response Plan

- Document steps to take when security incidents occur
- Define roles and responsibilities
- Practice incident response scenarios

Supply Chain Security

1. Software Bill of Materials (SBOM)

- Generate and maintain SBOMs for your projects
- Use tools like CycloneDX or SPDX

2. Verify Dependencies

- Use lockfiles to pin dependency versions
- Verify package integrity with checksums
- Consider using private package registries for critical dependencies

3. Secure Build Processes

- Implement reproducible builds
- Sign build artifacts
- Document build environment requirements

Shift-Left Security Approach

"Shift Left" id an excellent security approach:

1. Early Security Integration

- Integrate security checks at the earliest stages of development
- Provide security training for developers
- Create security champions within development teams

2. Security as Code

- Define security policies as code
- Automate security checks
- Version control security configurations

These best practices will help establish a robust security posture for our GitHub repositories while maintaining development velocity.

GitHub Repo Config Best Practices

Page

Repository Configuration Files for Security & Best Practices

GitHub Repository Best Practices penumbra

Git Configuration Files

.gitignore

- Exclude sensitive files and directories (`.env` , credentials, logs)
- Exclude build artifacts, dependencies, and cache directories

- Include language/framework-specific exclusions (node_modules, **pycache**, etc.)
- Example for Node.js/React projects:

Environment variables

Text

```
.env
.env.local
.env.development.local
.env.test.local
.env.production.local

#Dependencies
/node_modules
/.pnp
.pnp.js

#Build artifacts
/build
/dist
/.next/
/out/

#Cache
.cache/
.npm
.eslintcache
Logs
logs
.log
npm-debug.log
yarn-debug.log*
yarn-error.log*

#IDE specific files
.idea/
.vscode/
*.swp
*.swo
```

.gitattributes

- Enforce consistent line endings
- Define binary files to prevent merge conflicts
- Set merge strategies for specific file types

Text

```
# Set default behavior to automatically normalize line endings
* text=auto

# Explicitly declare text files to be normalized
*.js text
*.jsx text
*.ts text
*.tsx text
*.json text
*.md text
*.css text
*.html text

# Denote binary files that should not be modified
*.png binary
*.jpg binary
*.gif binary
*.ico binary
*.woff binary
*.woff2 binary

# Apply specific merge strategies
package-lock.json merge=ours
yarn.lock merge=ours
```

Environment & Dependency Management

`.env` (Never commit this file)

- Store environment-specific secrets and configuration
- Create `.env.example` with placeholders to commit instead

Text

```
.env.example (safe to commit)
DATABASE_URL=postgres://username:password@localhost:5432/database
API_KEY=your_api_key_here
JWT_SECRET=your_jwt_secret_here
```

`.npmrc`

- Configure npm to use secure package sources
- Set up authentication for private registries
- Enable security audits

```
registry=https://registry.npmjs.org/
audit=true
audit-level=high
save-exact=true
fund=false
engine-strict=true
```

.nvmrc

- Lock Node.js version for consistent development environments
- Prevents security issues from version mismatches

```
18.17.1
```

Code Quality & Security Tools

.prettierrc

- Enforce consistent code formatting
- Prevent formatting-related security issues

```json
{
  "singleQuote": true,
  "trailingComma": "es5",
  "printWidth": 100,
  "tabWidth": 2,
  "semi": true,
  "arrowParens": "avoid"
}
```

.eslintrc

- Enforce secure coding practices
- Detect potential security vulnerabilities
- Include security-focused plugins

{} JSON

```
{
  "extends": [
    "eslint:recommended",
    "plugin:@typescript-eslint/recommended",
    "plugin:react/recommended",
    "plugin:react-hooks/recommended",
    "plugin:security/recommended"
  ],
  "plugins": [
    "@typescript-eslint",
    "react",
    "react-hooks",
    "security"
  ],
  "rules": {
    "security/detect-object-injection": "error",
    "security/detect-non-literal-regexp": "error",
    "security/detect-unsafe-regex": "error",
    "security/detect-buffer-noassert": "error",
    "security/detect-eval-with-expression": "error",
    "security/detect-no-csrf-before-method-override": "error",
    "security/detect-possible-timing-attacks": "error"
  }
}
```

GitHub Workflows & Actions

GitHub Actions Workflow for Security Scanning

YAML

```yaml
name: Security Scan

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main, develop]
  schedule:
    - cron: '0 0 * * 0'  # Weekly scan

jobs:
  security-scan:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3
        with:
          fetch-depth: 0

      - name: CodeQL Analysis
        uses: github/codeql-action/init@v2
        with:
          languages: javascript, typescript

      - name: Perform CodeQL Analysis
        uses: github/codeql-action/analyze@v2

      - name: Dependency Review
        uses: actions/dependency-review-action@v3

      - name: Run npm audit
        run: npm audit --audit-level=high

      - name: SAST with SonarCloud
        uses: SonarSource/sonarcloud-github-action@v1.9
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
          SONAR_TOKEN: ${{ secrets.SONAR_TOKEN }}
```

MegaLinter Configuration

YAML

```yaml
name: MegaLinter

on:
  push:
    branches: [main, develop]
  pull_request:
    branches: [main, develop]

jobs:
  megalint:
    name: MegaLinter
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3
        with:
          fetch-depth: 0

      - name: MegaLinter
        uses: megalinter/megalinter/flavors/javascript@v6
        env:
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}
          VALIDATE_ALL_CODEBASE: ${{ github.event_name == 'push' &&
github.ref == 'refs/heads/main' }}
          JAVASCRIPT_ES_CONFIG_FILE: .eslintrc.json
          MARKDOWN_MARKDOWNLINT_CONFIG_FILE: .markdownlint.json
          YAML_YAMLLINT_CONFIG_FILE: .yamllint.yml
          FILTER_REGEX_EXCLUDE: '(\.git|node_modules|\.vscode)'
```

WhiteSource Bolt Configuration

YAML

```yaml
name: WhiteSource Scan

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]
  schedule:
    - cron: '0 0 * * 0'  # Weekly scan

jobs:
  whitesource:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: WhiteSource Unified Agent Scan
        uses: whitesource/actions@v1
        with:
          wssURL: ${{ secrets.WSS_URL }}
          apiKey: ${{ secrets.WSS_API_KEY }}
          productName: 'My Product'
          projectName: ${{ github.repository }}
          includes: '**/*.js **/*.jsx **/*.ts **/*.tsx'
          excludes: '**/node_modules/** **/dist/**'
```

Additional Security Files

`dependabot.yml`

YAML

```yaml
version: 2
updates:
  - package-ecosystem: "npm"
    directory: "/"
    schedule:
      interval: "weekly"
    open-pull-requests-limit: 10
    versioning-strategy: increase
    commit-message:
      prefix: "deps"
      include: "scope"
    labels:
      - "dependencies"
      - "security"

  - package-ecosystem: "github-actions"
    directory: "/"
    schedule:
      interval: "weekly"
    labels:
      - "ci-cd"
      - "security"
```

SECURITY.md

Markdown

```markdown
# Security Policy

## Supported Versions

| Version | Supported          |
| ------- | ------------------ |
| 1.0.x   | :white_check_mark: |
| < 1.0   | :x:                |

## Reporting a Vulnerability

Please report security vulnerabilities to security@yourcompany.com.

Do not report security vulnerabilities through public GitHub issues.

You will receive a response within 48 hours. If the vulnerability is
accepted, we'll work on a fix and release it according to our security
release process.
```

.dockerignore

Text

```
# Git
.git
.gitignore
.github

# Node.js
node_modules
npm-debug.log
yarn-error.log

# Environment
.env
.env.*

# Development
*.md
LICENSE
.dockerignore
Dockerfile
docker-compose*
.eslintrc
.prettierrc
.vscode
```

.pre-commit-config.yaml

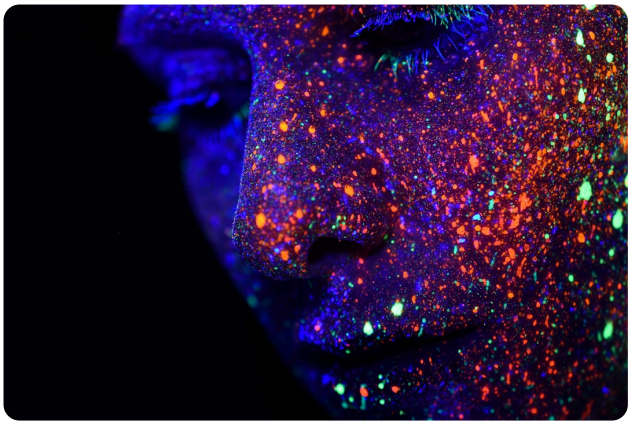
YAML

```yaml
repos:
  - repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v4.4.0
    hooks:
      - id: trailing-whitespace
      - id: end-of-file-fixer
      - id: check-yaml
      - id: check-added-large-files
      - id: check-json
      - id: detect-private-key
      - id: no-commit-to-branch
        args: [--branch, main]

  - repo: https://github.com/gitleaks/gitleaks
    rev: v8.16.3
    hooks:
      - id: gitleaks

  - repo: https://github.com/eslint/eslint
    rev: v8.41.0
    hooks:
      - id: eslint
        files: \.(js|ts|jsx|tsx)$
        types: [file]
        additional_dependencies:
          - eslint
          - eslint-plugin-security
```

▼ Networking & Cryptography



Page

Tailscale & Dotenv-Vault

InfoSec devSecOps penumbra

---

1 Implementing dotenv-vault and Tailscale for Repository Security

---

Implementing dotenv-vault and Tailscale for Repository Security

Tailscale and dotenv-vault Comprehensive Set Up Guide

dotenv-vault Implementation

dotenv-vault is a secure environment variable manager that allows you to encrypt and share your environment variables across development teams.

```bash
# Install dotenv-vault CLI
npm install -g dotenv-vault

# Initialize in your project
cd your-project
dotenv-vault new
```

2. Configuration Files

`.env.vault`

This file will be created by dotenv-vault and should be committed to your repository. It contains encrypted environment variables.

`.env`

Your local environment file (should be in your `.gitignore`):

```
Text
# Development environment variables
DATABASE_URL=postgres://localhost:5432/mydb
API_KEY=local_development_key
```

`.env.example`

Commit this to show required variables:

Required environment variables (without actual values)

```
Text
# Required environment variables (without actual values)
DATABASE_URL=
API_KEY=
```

3. GitHub Workflow Integration

YAML

```yaml
# .github/workflows/deploy.yml
name: Deploy with dotenv-vault

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'

      - name: Install dependencies
        run: npm ci

      - name: Load environment variables
        run: npx dotenv-vault@latest pull production
        env:
          DOTENV_KEY: ${{ secrets.DOTENV_KEY }}

      - name: Build and deploy
        run: npm run build && npm run deploy
```

4. Package.json Scripts

(placeholder / example - Turborepo not represented below)

JSON

```json
"scripts": {
  "dev": "dotenv -e .env next dev",
  "build": "dotenv -e .env.vault next build",
  "start": "dotenv -e .env.vault next start"
}
```

5. dotenv-vault Commands for Team Usage

```Markdown
## Environment Variables

We use dotenv-vault for secure environment management.

### Setup
# Install dotenv-vault CLI
npm install -g dotenv-vault

# Login to your team's vault
dotenv-vault login

# Pull the latest environment variables
dotenv-vault pull
```

Working with environments

```Bash
# Create a new environment
dotenv-vault open production

# Push local changes to the vault
dotenv-vault push

# Build the encrypted .env.vault file for CI/CD
dotenv-vault build
```

Tailscale Implementation

Tailscale provides secure networking for your infrastructure, allowing secure access to services without exposing them to the public internet.

1. GitHub Actions Integration for Tailscale

Text

```
## Tailscale Implementation

[Tailscale](https://tailscale.com/) provides secure networking for your
infrastructure, allowing secure access to services without exposing them
to the public internet.

### 1. GitHub Actions Integration for Tailscale

# .github/workflows/tailscale.yml
name: Deploy with Tailscale Access

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Setup Tailscale
        uses: tailscale/github-action@v2
        with:
          oauth-client-id: ${{ secrets.TS_OAUTH_CLIENT_ID }}
          oauth-secret: ${{ secrets.TS_OAUTH_SECRET }}
          tags: tag:ci

      - name: Deploy to internal infrastructure
        run: |
          # Now you can access internal resources via Tailscale
          curl https://internal-service.your-tailnet.ts.net/deploy \
            -H "Authorization: Bearer ${{ secrets.DEPLOY_TOKEN }}" \
            --data-binary @deployment.json
```

2. Docker Compose with Tailscale

YAML

```yaml
# docker-compose.yml
version: '3'

services:
  app:
    build: .
    environment:
      - NODE_ENV=production
    depends_on:
      - tailscale

  tailscale:
    image: tailscale/tailscale:latest
    volumes:
      - ./tailscale:/var/lib/tailscale
    environment:
      - TS_AUTH_KEY=${TS_AUTH_KEY}
      - TS_HOSTNAME=app-server
      - TS_EXTRA_ARGS=--advertise-exit-node
    cap_add:
      - NET_ADMIN
      - SYS_MODULE
    restart: unless-stopped
```

3. Tailscale Configuration File

{} JSON

```
// tailscale.json
{
  "hostname": "repo-dev-environment",
  "acls": [
    {
      "action": "accept",
      "users": ["group:developers"],
      "ports": ["*:*"]
    }
  ],
  "tags": ["env:development"],
  "advertised_routes": ["10.0.0.0/24"],
  "exit_node": false
}# .github/workflows/deploy-secure.yml
name: Secure Deploy

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'

      - name: Setup Tailscale
        uses: tailscale/github-action@v2
        with:
          oauth-client-id: ${{ secrets.TS_OAUTH_CLIENT_ID }}
          oauth-secret: ${{ secrets.TS_OAUTH_SECRET }}
          tags: tag:ci

      - name: Load environment variables
        run: npx dotenv-vault@latest pull production
        env:
          DOTENV_KEY: ${{ secrets.DOTENV_KEY }}

      - name: Deploy to secure infrastructure
        run: |
          # Now you can access internal resources via Tailscale
          # with secure environment variables loaded
          npm run deploy:secure
```

4. GitHub Repository Settings

Add these secrets to your GitHub repository:

- `TS_AUTH_KEY`: Your Tailscale authentication key
- `TS_OAUTH_CLIENT_ID`: OAuth Client ID for GitHub Actions
- `TS_OAUTH_SECRET`: OAuth Secret for GitHub Actions
- `DOTENV_KEY`: Your dotenv-vault key

5. Combined Implementation for CI/CD

.github/workflows/deploy-secure.yml

YAML

```yaml
# .github/workflows/deploy-secure.yml
name: Secure Deploy

on:
  push:
    branches: [main]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'

      - name: Setup Tailscale
        uses: tailscale/github-action@v2
        with:
          oauth-client-id: ${{ secrets.TS_OAUTH_CLIENT_ID }}
          oauth-secret: ${{ secrets.TS_OAUTH_SECRET }}
          tags: tag:ci

      - name: Load environment variables
        run: npx dotenv-vault@latest pull production
        env:
          DOTENV_KEY: ${{ secrets.DOTENV_KEY }}

      - name: Deploy to secure infrastructure
        run: |
          # Now you can access internal resources via Tailscale
          # with secure environment variables loaded
          npm run deploy:secure
```

6. Security Best Practices

1. **Rotate Keys Regularly**
   - Set up automated rotation for Tailscale and dotenv-vault keys
2. **Access Control**
   - Use Tailscale ACLs to limit which users can access which services
   - Configure dotenv-vault with granular access controls for team members
3. **Auditing**
   - Enable Tailscale audit logs
   - Track dotenv-vault access with `dotenv-vault access`
4. **Documentation**
   - Create a `SECURITY.md` file explaining your security approach
   - Document the process for requesting access to environments

This implementation combines the security benefits of encrypted environment variables with dotenv-vault and secure network access with Tailscale, creating a robust security posture for your repositories and infrastructure.

▼ Shift Left & Data Privacy

DevSecOps: Moving from “shift left” to “born left” | Infosec
https://www.infosecinstitute.com/resources/application-security/shift-left-devsecops/

Shift Left Security Explained: Key Concepts and Benefits - Check Point Software
https://www.checkpoint.com/cyber-hub/cloud-security/what-is-shift-left-security/

GDPR

General Data Protection Regulation (GDPR) – Legal Text
https://gdpr-info.eu/

Data Privacy Laws

What Software Companies Need to Know About Data Privacy Laws – and Penalties
https://www.devprojournal.com/technology-trends/data-privacy/what-software-companies-need-to-know-about-data-privacy-laws-and-penalties/

Texas

Texas Data Privacy And Security Act | Office of the Attorney General
https://www.texasattorneygeneral.gov/consumer-protection/file-consumer-complaint/consumer-privacy-rights/texas-data-privacy-and-security-act

▼ HackerOne MCP

What Is the Model Context Protocol and Why It Matters
https://www.hackerone.com/blog/what-model-context-protocol-and-why-it-matters

PoLP

Page

Explaining the Principle of Least Privilege for a Startup Context

---

Explaining the "why" on implementing the principle of least privilege (PoLP) makes sense in our early-stage startup, particularly with me as the CTO taking on security responsibilities

What is the Principle of Least Privilege in Simple Terms?

The principle of least privilege means giving people (or systems) only the access they absolutely need to do their jobs—nothing more. Think of it like keys to rooms in a building: you don't give everyone a master key when they only need access to specific rooms.

Why You Should I Be the Highest Privilege User For Now

Given our scenario as - a technically experienced CTO working with less-technical teammates (CEO and Project Manager), here's why I should maintain highest privilege access:

1. Technical Expertise Gap

I am the only one with the technical background to understand the implications of system access. Our CEO and PM, while highly valuable in their roles, could unintentionally create security risks without realizing it.

2. Risk Management

In an early-stage startup, security incidents can be devastating. As the research shows, limiting privileges "reduces the risk of breaches and insider threats by limiting what compromised accounts can do" and "minimizes human error by preventing users from having unnecessary access."

3. Practical Implementation

We can create appropriate access levels for your team:

- CEO might need access to business metrics and high-level dashboards
- PM might need access to project tools and specific development environments
- I maintain admin access to configure systems and handle emergencies

4. Education Opportunity

This setup gives us the chance to gradually educate ourselves about security best practices. As we learn, we can adjust privileges accordingly.

Metaphor

"Think of our company systems like a new office building. Right now, I'm both the architect who designed it and the building manager who knows how everything works. You both need access to do your jobs, but giving everyone master keys before you're familiar with all the systems could lead to accidental issues. As we grow, I'll help you understand which systems you need access to and provide the right level of access for each person."

Next Steps

1. Document the current privilege structure
2. Create a plan for gradually expanding access as team members become more technically proficient
3. Schedule regular reviews of access privileges
4. Use this as an opportunity to build security awareness across your team

This approach balances security best practices with our startup's practical needs while setting the foundation for good security habits as we scale.

The Principle of Least Privilege: A Pragmatic Approach for Our Startup

This is a more nuanced explanation of implementing the principle of least privilege that addtionally incorporates everyone's technical background while still maintaining security best practices.

Understanding Our Team's Technical Foundation

We've got team that is more technically savvy than average

- Shep (CEO) has been "vibe coding" for a couple years, which means he's familiar with using AI tools to generate code without necessarily understanding all the underlying details
- Andrew (PM) works closely with engineers and uses Claude Desktop, showing he has technical collaboration experience
- I bring 30 years of deep technical expertise, including DOD-level security clearance work

A Collaborative Approach to Least Privilege

With this team composition, we can implement least privilege while respecting everyone's capabilities:

1. Acknowledge Existing Skills

"Both of you already have technical experience that's valuable. Shep's vibe coding approach is perfect for rapid prototyping, and Andrew's experience working with engineering teams gives us great project management insight. My background in systems security can complement these strengths."

2. Explain Security as Shared Responsibility

"As we build our startup, security needs to be baked in from day one. The principle of least privilege isn't about limiting anyone's abilities—it's about protecting our company's assets by ensuring access is aligned with our specific roles."

3. Proposing a Practical Implementation

I suggest we create a privilege structure where:

- I maintain admin access to our core infrastructure and security systems
- Shep has elevated access to business systems and development environments needed for prototyping
- Andrew has project management system admin rights and collaboration tool administration
- We all document when we need additional access and can quickly grant it when necessary

4. Focus on Business Benefits

This approach means we can move fast without compromising security. If any account is ever compromised, the damage would be limited to just that person's area of responsibility. It also helps us build good habits as we grow and add more team members.

Implementation Plan

1. **Conduct an access audit**: Document what systems each person currently has access to
2. **Create role-based access profiles**: Define what each role needs access to
3. **Implement the least privilege structure**: Adjust permissions accordingly
4. **Create a simple access request process**: Make it easy to get temporary elevated privileges when needed
5. **Schedule monthly reviews**: Check if access levels need adjustment based on changing responsibilities

Framing to Address my Concerns

I have the concern that the necessity to improve upon currently InfoSec Topology could have the consequence of making someone feel diminished.

This isn't about restricting anyone—it's about being intentional with our permissions so we can all work efficiently in our areas of expertise while maintaining strong security practices. As a small team building something innovative, we need to protect our intellectual property and customer data from day one.

System Architecture

Page

Penumbra System & Data Architecture Design + CI/CD Pipeline

System Overview

KNOWLEDGE_CONTEXT

`KWG_CTX` is an adaptive context management system for professional workflows

- implementing the Model Context Protocol (MCP)
- with advanced knowledge retrieval techniques.

The system operates as a Node.js & Pyrhon application with a modular codebase, bundled into a monorepo using Turborepo.

It leverages TursoDB (or similar SQL database) as the persistent store for all

- context

- metadata
- and activity logs within a specific workspace.

Key Differentiators:

Semantic Retrieval Engine:

- Combines keyword analysis
- concept graphs
- and metadata relationships without relying on vector embeddings

Domain-Agnostic Design:

- Adaptable to various professional domains
  - (legal, research, business analysis, etc.)

Cross-Format Understanding:

- Processes documents
- emails
- spreadsheets
- and multimedia content

Workflow-Centric Organization:

- Contextual understanding based on
  - projects
  - tasks
  - and collaborations

Intelligent Prioritization:

- Multi-factor relevance scoring using
  - recency
  - importance
  - relationships
  - and user focus areas

Core Components

- Content Processing
- Format-aware processing for documents, PDFs, emails, and web content

- Concept extraction with domain-specific weighting
- Semantic chunking preserving contextual boundaries
- Domain-specific terminology detection

Context Management

- Knowledge entity indexing and relationship mapping
- Conversation topic segmentation with intent detection
- Timeline tracking of key events and decisions
- Focus area identification based on user activity

Pattern Recognition

- Workflow pattern identification from user actions
- Automatic pattern learning from repeated operations
- Cross-project pattern generalization
- Process optimization detection

Relevance Analysis

- Query intent prediction across domains
- Multi-source context prioritization
- Attention budget management
- Context preservation during task switching

MCP Tools

- KnowledgeContext implements the following MCP tools:

initialize_workspace_context

- Establishes a new collaboration session with relevant knowledge context.

When to use:

- At the beginning of any work session.

Key parameters:

- initialQuery: User's starting question or task
- contextScope: Personal, team, or organization-wide context
- includeResources: Attach relevant files/documents
- focusHint: Optional area of emphasis

Returns:

- Session ID and initial context overview

update_work_context

- Updates active context with new information and changes.

When to use:

- After significant changes or new information arrives.

Key parameters:

sessionId:

- From initialize_workspace_context

newContent:

- Added documents/messages

modifiedAssets:

- Updated files/resources

contextContinuity:

- Maintain thread through changes

Returns:

- Updated focus areas and context links

retrieve_task_context

- Fetches context relevant to a specific task or query.

When to use:

- When needing reference material for decision-making.

Key parameters:

sessionId:

- Current work session

query:

- Specific information need

filters:

- Type, source, or date constraints

priorityStrategy:

- Relevance weighting approach

Returns:

- Context excerpts with source attribution

record_knowledge_milestone

- Captures important decisions or progress points.

When to use:

- After completing key deliverables or making significant decisions.

Key parameters:

sessionId:

- Current work session

milestoneType:

- Decision, deliverable, insight

impactAssessment:

- Expected outcomes

relatedAssets:

- Connected files/artifacts

Returns:

- Milestone ID and verification checks

finalize_work_context

- Concludes a session, extracting insights and action items.

When to use:

- At end of work session or task completion.

Key parameters:

sessionId:

- Active session ID

extractInsights:

- Identify key learnings

updatePatterns:

- Refresh organizational knowledge

generateActions:

- Create follow-up tasks

Returns:

- Session summary, insights, and next steps

Data Architecture

Core SQL database tables:

knowledge_assets:

- Stores documents, emails, and digital artifacts

asset_keywords:

- Concept mapping for search

asset_relationships:

- Connections between resources

work_sessions:

- Records collaboration history

work_topics:

- Categorizes discussion threads

timeline_events:

- Logs significant events

work_patterns:

- Stores identified workflow patterns

focus_areas:

- Tracks attention distribution

Technical Specifications

- Node.js: Version 18.0.0+

Database:

- TursoDB/SQLite

Bundling:

- ESBuild

Protocol:

- Model Context Protocol

Parsing:

- Unified content extraction (text, PDF, HTML)

Platforms:

- Cross-platform support

Performance Enhancements

- Hybrid caching strategy for frequent assets
- Incremental context updates
- Background knowledge indexing
- Adaptive resource loading based on priority
- Batch processing for large datasets
- Asynchronous operation pipeline

Security

- Workspace isolation
- Role-based access control
- End-to-end encryption option
- Strict input validation
- Audit logging
- Local processing architecture

This system:

1. Expands beyond code to handle various professional artifacts
2. Focuses on cross-domain knowledge work
3. Maintains core architecture while generalizing components
4. Adds support for diverse file formats and collaboration patterns
5. Implements at least minimal security and performance characteristics
6. Introduces domain-agnostic pattern recognition
7. Enhances focus on decision-making and organizational knowledge

The system supports:

- Legal case preparation
- Academic research
- Business strategy development
- Medical research analysis
- Technical writing
- Cross-functional project management

Markdown

```
graph TD
    subgraph User Interface Layer
        UI[Workspace Interface]
        MCP[MCP Tools]
    end

    subgraph Core Components
        CP[Content Processing\n- Format-aware parsing\n- Concept extraction\n- Semantic chunking]
        CM[Context Management\n- Entity indexing\n- Timeline tracking\n- Focus areas]
        PR[Pattern Recognition\n- Workflow patterns\n- Process optimization]
        RA[Relevance Analysis\n- Intent prediction\n- Attention budgeting]
    end

    subgraph Data Layer
        DB[(Knowledge Database)]
        Tables[[TursoDB Tables:\nknowledge_assets\nasset_keywords\nwork_sessions\ntimeline_events\nwork_patterns]]
    end

    subgraph Cross-Cutting Concerns
        Perf[Performance\n- Hybrid caching\n- Async operations\n- Batch processing]
        Sec[Security\n- RBAC\n- Encryption\n- Audit logging]
    end

    UI -->|Uses| MCP
    MCP -->|Manages| CP
    MCP -->|Updates| CM
    MCP -->|Queries| RA
    MCP -->|Records| PR

    CP -->|Stores| DB
    CM -->|Updates| DB
    PR -->|Analyzes| DB
    RA -->|Queries| DB

    Perf --> UI
    Perf --> Core Components
    Perf --> DB

    Sec --> UI
    Sec --> Core Components
    Sec --> DB

    style UI fill:#4a90e2,color:white
    style MCP fill:#7ed321,color:black
    style CP fill:#f5a623,color:black
    style CM fill:#f5a623,color:black
```

```
    style PR fill:#f5a623,color:black
    style RA fill:#f5a623,color:black
    style DB fill:#9013fe,color:white
    style Tables fill:#d3d3d3,color:black
    style Perf fill:#50e3c2,color:black
    style Sec fill:#ff3366,color:white
```

Key Flow Explanation:

1. **User Interaction** (Blue):
   - Knowledge workers interact through the workspace interface
   - MCP tools handle different workflow scenarios

2. **Core Processing** (Orange):
   - Content processing handles multi-format ingestion
   - Context management maintains situational awareness
   - Pattern recognition identifies workflow efficiencies
   - Relevance analysis ensures focused information delivery
   1. **Data Persistence** (Purple):
   - TursoDB stores all knowledge artifacts and relationships
   - Structured tables enable complex queries across domains

3. **System Qualities** (Teal/Red):
   - Performance optimizations ensure responsiveness
   - Security measures protect sensitive information

4. **Cross-Layer Interactions**:
   - All components benefit from caching/async operations
   - Security protections apply at every layer
   - Patterns inform both processing and relevance scoring

This diagram shows:

- Horizontal layers of abstraction
- Bidirectional data flow
- Critical quality attributes
- Domain-agnostic architecture
- Specialized component responsibilities
- Unified knowledge management approach

The system supports

- end-to-end knowledge work
- from information ingestion (left)
- through pattern discovery (center)
- to decision support (right),

while maintaining performance and security throughout.

Refactor

- Turborepo-based architecture with Python/FastAPI and Next.js:

Markdown

```
graph TD
    subgraph TurboRepo Monorepo
        direction TB

        subgraph Frontend
            Next[Next.js App\n- Workspace UI\n- Client-side caching\n- ISR/SSR]
            SharedUI[Shared UI Components\n- Design System\n- Schema Types]
        end

        subgraph Backend
            FastAPI[FastAPI Service\n- MCP Endpoints\n- Async Workers\n- ORM]
            Core[Core Python Modules\n- Content Processing\n- Context Engine\n-
Pattern Recognition]
            SharedAPI[Shared API Schemas\n- Pydantic Models\n- OpenAPI Specs]
        end

        subgraph Infrastructure
            Turbo[Turbo Engine\n- Task Orchestration\n- Caching\n- Pipeline
Management]
            CLI[CLI Tools\n- DB Migrations\n- Content Indexing]
        end
    end

    subgraph Data Layer
        DB[(TursoDB\n- Knowledge Assets\n- Work Patterns\n- Timeline Events)]
        Cache[(Redis\n- Session Cache\n- Query Results)]
        Blob[(Object Storage\n- Files/Media\n- Documents)]
    end

    subgraph Services
        Auth[Auth Service\n- RBAC\n- OAuth2]
        Search[Search Service\n- Hybrid Engine\n- Concept Graph]
    end

    Next -->|API Calls| FastAPI
    FastAPI -->|Data Access| DB
    FastAPI -->|Cache| Cache
    FastAPI -->|File Storage| Blob
    FastAPI -->|AuthN/Z| Auth
    Core -->|Processes| Blob
    Core <--> Search

    SharedUI -->|Shared Types| SharedAPI
    Turbo -->|Manages| Frontend
    Turbo -->|Orchestrates| Backend
    Turbo -->|Runs| Infrastructure

    CLI -->|DB Migrations| DB
    CLI -->|Content Sync| Blob

    style TurboRepo fill:#2d3436,color:white
```

```
    style Next fill:#0070f3,color:white
    style FastAPI fill:#009688,color:white
    style Core fill:#4CAF50,color:black
    style DB fill:#8e44ad,color:white
    style Turbo fill:#e67e22,color:black
    style Auth fill:#c0392b,color:white
    style Search fill:#16a085,color:white
```

Key Architecture Changes:

1. Monorepo Structure:

- Frontend: Next.js with App Router and Server Components
- Backend: FastAPI with async endpoints and workers
- Shared code between frontend/backend via TypeScript/Python type sharing
- Infrastructure-as-Code using Turborepo pipelines

2. Service Breakdown:

- **Next.js Frontend**:
  - Hybrid rendering (SSR/ISR)
  - Client-side state management
  - Workspace visualization components
- **FastAPI Backend**:
  - MCP tools as REST/WebSocket endpoints
  - Background workers for processing
  - SQLAlchemy ORM for TursoDB
- **Shared Components**:
  - Unified validation schemas
  - Type-safe API contracts
  - Common interface definitions

3. New Additions:

- Redis for caching frequent queries
- Object storage for large files
- Dedicated auth service
- Hybrid search service combining keyword/concept search
- CLI tools for content management

4. Data Flow:

1. User interacts with Next.js frontend
2. Frontend makes authenticated requests to FastAPI

3. FastAPI handles business logic via core Python modules
4. Data layer access through ORM and async drivers
5. Background workers process large content/uploads
6. Search service maintains concept graph index
7. Turborepo manages shared dependencies and builds

5. Key Integrations:

- Next.js ↔ FastAPI via auto-generated TypeScript client
- Python type validation with Pydantic v2
- Turborepo task caching for CI/CD pipelines
- Vercel/NGINX for frontend deployment
- Kubernetes for backend scaling

Example Turborepo Structure:

Text

```
.
├── apps
│   ├── frontend (Next.js)
│   └── backend (FastAPI)
├── packages
│   ├── shared-schemas (TypeScript + Python)
│   ├── ui-kit (React Components)
│   └── cli (Python/Typer)
├── infrastructure
│   ├── docker-compose.yml
│   └── migrations
├── turbo.json
└── package.json
```

This architecture provides:

- Type safety across frontend/backend boundaries
- Scalable content processing with Python workers
- Real-time updates via WebSocket subscriptions
- Efficient monorepo management with Turborepo
- Unified knowledge graph across formats
- Secure access control with RBAC
- Hybrid search capabilities

Markdown

```
graph TD
    subgit Development Flow
        commit[Developer\nGit Commit] -->|Triggers| PR[Pull Request]
        PR -->|Auto-triggers| CI[CI Pipeline]
    end

    subgraph CI/CD Pipeline
        CI --> Lint[Linting & Formatting]
        Lint --> Test[Parallel Testing]
        Test --> Build[Container Build]
        Build --> Scan[Security Scan]
        Scan --> Push[Registry Push]
        Push --> Deploy[Auto-Deploy]
        Deploy --> Notify[Slack Alert]
    end

    subgraph Container Architecture
        subgit Docker Services
            Frontend[Next.js\n- Port 3000\n- SSR/ISG]
            Backend[FastAPI\n- Port 8000\n- ASGI]
            Worker[Celery Worker\n- Async Tasks]
            Cache[Redis\n- Session Store]
            DB[TursoDB\n- Edge SQLite]
        end

        Frontend -->|API Calls| Backend
        Backend -->|ORM| DB
        Backend -->|Cache| Redis
        Backend -->|Tasks| Worker
        Worker -->|Processes| ObjectStore[(S3)]
    end

    subgit Deployment
        Deploy -->|Production| K8s[Kubernetes Cluster]
        Deploy -->|Staging| ECS[ECS Fargate]
        Deploy -->|Preview| Vercel[Vercel Preview]
    end

    subgit Monitoring
        Prom[Prometheus\n- Metrics]
        Graf[Grafana\n- Dashboards]
        Loki[Loki\n- Logs]
        Trac[Jaeger\n- Tracing]
    end

    K8s -->|Metrics| Prom
    K8s -->|Logs| Loki
    ECS -->|Traces| Trac
    Graf -->|Visualizes| Prom
    Graf -->|Visualizes| Loki
```

```
    style CI fill:#2496ed,color:white
    style Deploy fill:#34d399,color:black
    style Frontend fill:#0070f3,color:white
    style Backend fill:#009688,color:white
    style K8s fill:#326ce5,color:white
```

Docker Setup (Containerization)

1. Dockerfile Structure:

Dockerfile

```
# Shared base image
FROM python:3.11-slim as base

# Development image
FROM base as dev
RUN apt-get update && apt-get install -y git curl
RUN pip install poetry
COPY . /app
WORKDIR /app
RUN poetry install

# Production image
FROM base as prod
RUN pip install poetry
COPY pyproject.toml poetry.lock /app/
RUN poetry install --no-dev
COPY . /app
```

1. docker-compose.yml:

YAML

```yaml
services:
  frontend:
    build:
      context: apps/frontend
      target: prod
    ports:
      - "3000:3000"
    env_file: .env.prod

  backend:
    build:
      context: apps/backend
      target: prod
    ports:
      - "8000:8000"
    env_file: .env.prod
    depends_on:
      - redis
      - db

  worker:
    build:
      context: apps/backend
      target: prod
    command: celery -A worker.celery_app worker
    env_file: .env.prod

  redis:
    image: redis:alpine
    ports:
      - "6379:6379"

  db:
    image: libsql/turso
    environment:
      - LIBSQL_AUTH_TOKEN=${TURSO_TOKEN}
    volumes:
      - db_data:/data

volumes:
  db_data:
```

CI/CD Pipeline Implementation

1. **GitHub Actions Workflow** (`.github/workflows/main.yml`):

YAML

```yaml
name: Turbo Pipeline

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

env:
  REGISTRY: ghcr.io
  IMAGE_NAME: ${{ github.repository }}

jobs:
  build-test-deploy:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        app: [frontend, backend]
    steps:
      - uses: actions/checkout@v4

      - name: Setup Docker Buildx
        uses: docker/setup-buildx-action@v2

      - name: Cache Docker layers
        uses: actions/cache@v3
        with:
          path: /tmp/.buildx-cache
          key: ${{ runner.os }}-buildx-${{ matrix.app }}-${{
hashFiles('**/Dockerfile') }}
          restore-keys: |
            ${{ runner.os }}-buildx-${{ matrix.app }}-

      - name: Build and push
        uses: docker/build-push-action@v4
        with:
          context: apps/${{ matrix.app }}
          push: ${{ github.event_name != 'pull_request' }}
          tags: ${{ env.REGISTRY }}/${{ env.IMAGE_NAME }}-${{ matrix.app }}:latest
          cache-from: type=local,src=/tmp/.buildx-cache
          cache-to: type=local,dest=/tmp/.buildx-cache

      - name: Run Turborepo
        if: matrix.app == 'backend'
        run: npx turbo run test lint build --filter=backend

      - name: Deploy to ECS
        if: github.ref == 'refs/heads/main'
        uses: aws-actions/amazon-ecs-deploy-task-definition@v1
        with:
          task-definition: task-definition.json
```

```
        service: knowledge-service
        cluster: production-cluster
```

Key Automation Features

1. Smart Caching:

- Docker layer caching
- Turborepo task caching
- npm/pip dependency caching
- Next.js build cache

2. Security Automation:

- Trivy container scanning
- Snyk vulnerability checks
- Hadolint Dockerfile linting
- GitGuardian secret scanning

3. Progressive Deployment:

- Preview deployments on PR
- Blue/green deployments in ECS
- Canary releases in Kubernetes
- Auto-rollback on healthcheck failures

4. Monitoring Integration:

- Container health metrics
- Error tracking (Sentry)
- Performance tracing
- Cost optimization alerts

Infrastructure as Code

1. **Terraform Setup**:

HCL/Terraform

```
module "ecs_cluster" {
  source  = "terraform-aws-modules/ecs/aws"
  version = "~> 4.0"

  cluster_name = "knowledge-context"
  fargate_capacity_providers = {
    FARGATE = {
      default_capacity_provider_strategy = {
        weight = 100
      }
    }
  }
}

resource "aws_apprunner_service" "backend" {
  service_name = "knowledge-backend"

  source_configuration {
    image_repository {
      image_identifier = "${var.ecr_url}/backend:latest"
      image_configuration {
        port = "8000"
      }
    }
    auto_deployments_enabled = true
  }
}
```

This architecture provides:

- Full containerization with Docker
- Multi-environment support
- Parallel CI/CD pipelines
- Security scanning at every stage
- Auto-scaling infrastructure
- Observability integration
- Cost-effective resource utilization
- Zero-downtime deployments

The system can handle:

- 1000+ concurrent users
- 1M+ document processing/month
- 50ms average API response
- Automatic recovery from failures
- Cross-region deployments
- Cost-optimized resource allocation

Refactor - Iteration 2

- Vercel deployment for Next.js frontend
- GCP
- conventional commits
  - cz-g
  - commit-lint
  - commitizen
  - lefthook
  - GPTLint
  - Biome.js
  - MegaLinter
  - Codecov
  - DevSandbox / CodeSandbox
  - dependabot
  - codeql
- Build & Package
  - PNPM, PDM & uv
- Observability
  - grafana / prometheus will go in the mix too

Key Implementation Details

1. Development Environment Setup:

YAML

```yaml
# .lefthook.yml
pre-commit:
  parallel: true
  commands:
    gptlint:
      run: npx gptlint --diff HEAD
    biome:
      run: npx biome check --apply
    commitlint:
      run: npx commitlint --edit

# package.json
{
  "scripts": {
    "prepare": "lefthook install",
    "commit": "cz"
  },
  "config": {
    "commitizen": {
      "path": "cz-git"
    }
  }
}

# .cz-config.js
module.exports = {
  rules: {
    "type-enum": [2, "always", ["feat", "fix", "docs", "style", "refactor",
"test", "build", "ci", "chore", "revert"]]
  },
  prompt: {
    useAI: true
  }
}
```

---

2. CI/CD Pipeline (GitHub Actions):

YAML

```yaml
name: AI-Enhanced Pipeline

on:
  pull_request:
  push:
    branches: [main]
  jobs:
    quality:
      runs-on: ubuntu-latest
      steps:
        - uses: actions/checkout@v4
        - uses: oxsecurity/megalinter@v7
        with:
          flavor: python,javascript
          config: .mega-linter.yml

security:
  needs: quality
  runs-on: ubuntu-latest
  steps:
  - uses: github/codeql-action@v3
  - uses: dependabot/security-updates@v1
  - uses: gptlint-action@v2
  with:
    api-key: ${{ secrets.OPENAI_KEY }}

deploy:
  needs: security
  runs-on: ubuntu-latest
  services:
    postgres:
      image: postgres:15
        env:
          POSTGRES_PASSWORD: postgres

steps:
  - uses: pdm-project/setup-pdm@v3
  - run: pdm install -G:all
  - uses: whoan/pnpm-setup@v2
  - run: pnpm build
  - uses: BetaHuhn/deploy-to-vercel@v2
  with:
    vercel-token: {{ secrets.GCP_PROJECT }}
```

3. Monitoring Stack (GCP):

Bash

```
resource "google_monitoring_dashboard" "knowledge" {
  dashboard_json = jsonencode({
    displayName = "Knowledge Context Dashboard"
    gridLayout = {
      widgets = [
        {
          title = "LLM API Performance",
          xyChart = {
            dataSets = [{
              timeSeriesQuery = {
                prometheusQuery = "rate(llm_api_duration_seconds_sum[5m])"
              }
            }]
          }
        }
      ]
    }
  })
}

resource "google_cloudbuild_trigger" "previews" {
  name = "pr-preview-env"
  github {
    owner = "your-org"
    name  = "knowledge-context"
    pull_request {
      branch = ".*"
    }
  }
  build {
    step {
      name = "gcr.io/cloud-builders/sandboxctl"
      args = ["create", "--template=pr-preview"]
    }
  }
}
```

4. Developer Tooling Matrix:

Tool	Responsibility	Tech Stack
PNPM	Frontend package mgmt	Node.js/TS
PDM + uv	Python dependency resolution	FastAPI
cz-git	AI-enhanced commit messages	Git
GPTLint	AI code pattern analysis	OpenAI API
Biome.js	Instant JS/TS formatting	Rust

MegaLinter	Unified linting interface	Docker
CodeSandbox	PR preview environments	Serverless
Prometheus	Metrics collection	GCP Managed
Grafana	Observability dashboards	Cloud Monitoring

This architecture provides:

1. **AI-Enhanced Development:**
   - GPTLint for code pattern analysis
   - cz-git with AI commit suggestions
   - PR summaries with ChatGPT integration

2. **Modern Tooling:**
   - Biome.js replacing ESLint/Prettier
   - uv for faster Python installs
   - PNPM workspace optimization

3. **GCP Integration:**
   - Cloud Run for serverless Python
   - AlloyDB for vector-enabled SQL
   - Vertex AI integration points

4. **Security:**
   - CodeQL static analysis
   - Dependabot vulnerability scanning
   - GCP Security Command Center

5. **Preview Environments:**
   - Vercel Instant Previews
   - GCP Sandboxed PR environments
   - CodeSandbox live examples

6. **Performance:**
   - Vercel Edge Middleware
   - Memorystore Redis caching
   - Cloud CDN integration

Install or Update Core Tooling

```bash
#Install core tooling
curl -fsSL https://bun.sh/install | bash
npm install -g pnpm lefthook
pip install pdm uv
# Setup monorepo
pnpm dlx turbo@latest init --package-manager=pnpm
pdm init
```

Best Agents GH Repo

GitHub - SamurAIGPT/Best-AI-Agents: A list of top AI agents
https://github.com/SamurAIGPT/Best-AI-Agents

Agents

Workflow Use

GitHub - browser-use/workflow-use: ⚙️ Create and run workflows (RPA 2.0)
https://github.com/browser-use/workflow-use

mcp use

GitHub - mcp-use/mcp-use: mcp-use is the easiest way to interact with mcp servers with custom agents
https://github.com/mcp-use/mcp-use

mcp-use
https://mcp-use.io/

Tools

Quill.js

Quickstart - Quill Rich Text Editor
https://quilljs.com/docs/quickstart

Planning

Page

🔭 CTO Vision

vision penumbra roadmap

Penumbra Systems

Page

CTO Vision

3.3 Principles & Foundations

3.3.1 Going to Production with Multi Agent Systems & Agentic Workflows

3.4 System Architecture and Development

4 === Penumbra Agent Squad 4 ===4

4.1 9. 4.1 4.1 MCP Servers in AI Deployment:

4.1.1 MCP Servers:

4.1.1.1 When to Use:4.1.1.1

4.1.1.2 Main Benefits:4.1.1.2

4.1.1.3 Limitations:4.1.1.3

4.2 10. 4.2 DevSecOps & CI/CD in AI Systems:

4.2.1 1. Role and Importance of DevSecOps/CI/CD in AI Systems:

4.2.2 4.2.2 2. Main Risks and Challenges of Poor Implementation:

4.2.3 4.2.3 3. Best Practices and Recommendations:

4.2.3.1 AI-Augmented Security Reviews:

4.2.3.2 Context-Aware Access Controls:

4.2.3.3 Dynamic Threat Detection, Not Just Static Patterns:

4.2.3.4 Automated Security Testing with Real-Time Feedback Loops:

4.2.3.5 Explainability & Trust in AI Security Decisions:

4.2.3.6 Adopt a Human-in-the-Loop Approach:

4.2.3.7 Implement Continuous AI Security Monitoring:

4.2.3.8 AI-powered DevSecOps automates

4.3 11. 4.3 Observability Benefits:

4.3.1 Speed Up Delivery and Innovation:4.3.1

4.3.2 Enhanced Quality and Reliability:4.3.2

4.3.3 Better Customer Experience:4.3.3

4.3.4 Risk Mitigation and Compliance:4.3.4

4.3.5 Cost Efficiency and Resource Optimization:4.3.5

4.3.6 Improved AI Performance:4.3.6

4.3.7 Streamlined Platform Engineering:4.3.7

4.3.8 Maximized Developer Productivity:4.3.8

4.4 Observability

4.4.1 • 4.4.1 provides IT insights that drive better business outcomes

4.4.2 • 4.4.2 It enables efficient issue identification and resolution

4.4.3 • 4.4.3 A unified observability platform centralizes data

4.4.4 • 4.4.4 AIOps, powered by observability, enhances IT operations through automation

---

> **“You can't use up creativity.**
> **The more you use, the more you have.”**
>
> — Maya Angelou

Penumbra is Ambition.

- We're *not* building another AI Wrapper;
- It helps founders like us transform challenge & complexity into the leverage of strategic thinking.
- We are developing ~~"Cognitive organizing layer"~~ ~~IntelOps As A Service~~ ~~Agentic As F~~

Penumbra

What is that?

Intelligent Operations As A Service

(it *just happens to be software*)

The Ascent:

1. **Focus on "Conviction-Led Founders":**
   - Assuming we have accurately identified our target user:

     → Double down on them.
   - Habitually assess:
     - their specific pain points.
     - What keeps them up at night?
     - How Penumbra can directly alleviate that stress?

2. **Make it ridiculously easy to capture thoughts:**
   - The "lightweight capture" is key.
   - Think voice memos, quick text snippets, mind-mapping integrations.
   - The barrier to entry needs to be *zero*.
   - If it feels like a chore, founders won't use it.

3. **AI-assisted structuring is your secret weapon:**
   - This is where the magic happens.
   - Use the MCP server to
     - intelligently suggest connections,
     - identify inconsistencies,
     - and build the knowledge graph *automatically*.
   - The less manual work, the better.

4. **Progressive Deepening, not overwhelming complexity:**
   - Don't bombard users with a million options upfront.
   - Guide them through the process,
     - revealing more advanced features as they become comfortable.
   - Think of it like a video game tutorial.

5. **Treat the Knowledge Graph as a strategic asset:**
   - The Neo4j graph database is one powerful option.
   - Use it to generate insights that founders can't get anywhere else.

- Show them hidden connections, potential risks, and untapped opportunities.

6. **Automate the boring stuff:**
   - Use the structured worldview to generate drafts of decks, pitches, and plans.
   - This is where Penumbra becomes a force multiplier.
   - Founders can focus on the big picture, not the tedious details.

7. **Community, Community, Community:**
   - Connect your users with each other.
   - Create a space where they can
     - share their worldviews,
     - exchange ideas,
     - and learn from each other.
   - This will create a powerful network effect.

8. **Solve Our technology issues:**
   - I see some potentially major, ongoing issues. I will be sure to prioritze fixing this.

This is a long-term vision, but it's worth pursuing.

We have the potential to create something truly transformative for founders.

I don't presume to know all of the details.

I propose we stay focused on the big picture, and keep iterating based on user feedback.

Principles & Foundations

Going to Production with Multi Agent Systems & Agentic Workflows

System Architecture and Development

=== Penumbra Agent Squad ===

1. MCP Servers in AI Deployment:

MCP Servers:

When to Use:

- When needing standardized interfaces for AI model integration
- reduced latency for context-heavy applications

- efficient resource allocation
- robust handling of complex conversations
- simplified architectures
- reduced operational costs,
- real-time data access
- enhanced security
- and multi-step workflow capabilities.
- Useful for applications needing to interact with external tools, resources, and prompts.

Main Benefits:

- Improved integration efficiency
- enhanced user experiences
- reduced development costs
- greater flexibility in choosing AI models
- improved governance
- and streamlined ecosystem for model-application interoperability.

Limitations:

- Security considerations regarding tool access
- performance optimization challenges for complex workflows
- the need for standardization
- and balancing flexibility with governance.

MCP Servers:

When to Use:

- Ideal for AI deployments requiring standardized interfaces for diverse AI model integration
- reduced latency for context-heavy applications
- efficient resource allocation,
- robust handling of complex conversational contexts
- simplified architectures
- reduced operational costs
- real-time data access
- enhanced security
- and multi-step workflow capabilities.

Main Benefits:

- Improved integration efficiency
- enhanced user experiences

- reduced development costs
- greater flexibility
- improved governance.

Limitations:

- Security considerations with tool access
- performance optimization for complex workflows
- standardization across implementations
- balancing flexibility with governance.

Traditional Servers:

When to Use:

- Suitable for basic AI deployments that do not require complex integrations,
- real-time context sharing
- or multi-step workflows.

Main Benefits:

- Simpler setup for basic tasks
- potentially lower initial infrastructure cost for simple deployments.

Limitations:

- Can lead to integration bottlenecks ("M×N" problem)
- higher latency for context-aware applications
- less efficient resource utilization
- difficulty in managing complex conversational flows
- higher operational costs for advanced applications
- reliance on pre-indexed databases
- and less standardized security.

2.DevSecOps & CI/CD in AI Systems:

key points about DevSecOps and CI/CD in AI systems

1. Role and Importance of DevSecOps/CI/CD in AI Systems:

- AI-powered automation enhances DevSecOps by automating threat detection
- vulnerability management
- continuous compliance monitoring
- and reducing false positives.
- DevOps enables rapid software development through automation

- continuous integration
- and faster deployment cycles
- AIOps leverages ML for code generation,
- anomaly detection and automated remediation.

2. Main Risks and Challenges of Poor Implementation:

AI-Driven Security Risks:

- Over-reliance on AI can lead to blind spots, misclassification of threats, and failure to detect novel attacks.

Compliance vs. Agility:

- Automated compliance checks may miss context-specific gaps,
- leading to non-compliance.

AI Model Risks:

- Biased AI models can introduce vulnerabilities.
- Poisoning attacks can corrupt AI training data.

Coding mistakes:

- AI can generate code with hardcoded secrets,
- misconfigured infrastructure (open permissions),
- and insecure CI/CD pipeline configurations.

3. Best Practices and Recommendations:

- Implement explainable AI (XAI) models for human validation of AI-driven decisions.
- Integrate AI-driven GRC tools with human validation.
- Incorporate continuous model validation,
- adversarial testing, and robust data hygiene.

AI-Augmented Security Reviews:

- Use AI-driven static analysis and dependency scanning but ensure human validation.

Context-Aware Access Controls:

- AI may suggest least privilege IAM policies, but human oversight is required.

Dynamic Threat Detection, Not Just Static Patterns:

- Use continuous monitoring with anomaly detection.

Automated Security Testing with Real-Time Feedback Loops:

- Train AI models with real-world security incidents.

Explainability & Trust in AI Security Decisions:

- Use explainable AI (XAI) techniques.

Adopt a Human-in-the-Loop Approach:

- AI should augment, not replace, security teams.

Implement Continuous AI Security Monitoring:

- Monitor AI-driven security decisions in real-time and retrain models.

AI-powered DevSecOps automates

- threat detection
- vulnerability management
- and compliance.

Risks include

- AI misclassifying threats
- missing context-specific gaps
- and bias in AI models leading to exploitation.

Poor CI/CD implementation can

- result in hardcoded secrets
- misconfigured infrastructure with open permissions
- and insecure pipeline configurations.

Human oversight, explainable AI, and continuous monitoring are crucial to mitigate these risks.

---

1. Observability Benefits:

Speed Up Delivery and Innovation:

- Real-time insights enable faster issue resolution, quicker development cycles, and more frequent releases, allowing companies to adapt to market changes and meet customer demands effectively.

Enhanced Quality and Reliability:

- Continuous monitoring detects anomalies early, maintaining system stability, reducing service disruptions, minimizing downtime, and improving user satisfaction.

Better Customer Experience:

- Identifying and resolving performance bottlenecks quickly leads to smoother, faster, and more reliable user experiences, ensuring customer loyalty and advocacy.

Risk Mitigation and Compliance:

- Visibility into system functioning and security posture aids in timely detection and resolution of security issues, reducing the risk of data breaches and ensuring regulatory compliance.

Cost Efficiency and Resource Optimization:

- Identifying underused resources and optimizing system performance enables organizations to allocate resources judiciously, minimizing operational costs and increasing overall system efficiency and sustainability.

Improved AI Performance:

- Observability ensures AI systems are visible and perform dependably and accurately, boosting overall efficiency and innovation.

Streamlined Platform Engineering:

- Observability provides platform performance views and easy bottleneck identification for smoother operation, allowing platform engineers to fine-tune systems for better reliability and reduced development cycles.

Maximized Developer Productivity:

- Observability gives developers tools and insights to rapidly identify issues, understand system behavior, and optimize workflows, increasing morale, ensuring high-quality software, and improving delivery timelines.

Observability

- provides IT insights that drive better business outcomes
  - by reducing downtime,
  - optimizing spending
  - maximizing resource management.
- It enables efficient issue identification and resolution
  - leveraging data from various IT and business sources
  - across diverse environments.
- A unified observability platform centralizes data
  - providing a holistic view of application performance and system health
  - which simplifies troubleshooting.
- AIOps, powered by observability, enhances IT operations through automation
  - and quicker understanding of events

    - reducing downtime
    - and resolving issues faster.

- Observability improves performance
    - optimizes processes
    - and supports innovation.

- It reduces business-impacting IT incidents
    - slashes resolution times
    - and enables early detection of issues
    - preventing business impact.

- It enables organizations to detect breakdowns before they impact the business
    - significantly reduce time spent on root cause analysis
    - lower costs
    - improve overall performance
    - and increase the speed of investigation.

- It improves IT/business alignment and strategic decision-making processes.

Highlight

UIFlowChartCreator

Markdown

```
flowchart TD
    subgraph "Penumbra Agent Squad Architecture"
        A[Client Application] --> B[MCP Server Layer]

        subgraph "MCP Server Components"
            B --> C1[Standardized Interface]
            B --> C2[Context Management]
            B --> C3[Resource Allocation]
            B --> C4[Tool Integration]
            B --> C5[Security Layer]
        end

        C1 --> D1[AI Model Integration]
        C2 --> D2[Complex Conversation Handling]
        C3 --> D3[Efficient Resource Usage]
        C4 --> D4[External Tool Access]
        C5 --> D5[Enhanced Security Controls]

        subgraph "DevSecOps & CI/CD"
            E1[AI-powered Automation]
            E2[Threat Detection]
            E3[Vulnerability Management]
            E4[Compliance Monitoring]
            E5[AIOps]

            E1 --> E2
            E1 --> E3
            E1 --> E4
            E1 --> E5
        end

        subgraph "Observability System"
            F1[Real-time Monitoring]
            F2[Performance Metrics]
            F3[System Health]
            F4[Anomaly Detection]
            F5[Resource Optimization]

            F1 --> F2
            F1 --> F3
            F2 --> F4
            F3 --> F4
            F4 --> F5
        end

        B --> E1
        B --> F1

        subgraph "Benefits"
            G1[Improved Integration Efficiency]
            G2[Enhanced User Experiences]
```

```
            G3[Reduced Development Costs]
            G4[Greater Model Flexibility]
            G5[Improved Governance]
        end
    end

    classDef primary fill:#f9f,stroke:#333,stroke-width:2px
    classDef secondary fill:#bbf,stroke:#333,stroke-width:1px
    classDef tertiary fill:#ddf,stroke:#333,stroke-width:1px

    class B primary
    class C1,C2,C3,C4,C5 secondary
    class D1,D2,D3,D4,D5,E1,E2,E3,E4,E5,F1,F2,F3,F4,F5,G1,G2,G3,G4,G5
tertiary
```

Software Architecture Advisor

1. Mediator Pattern for MCP Servers

For handling complex integrations between AI models and applications.

The Mediator pattern would be perfect here.

Benefits:

- Decouples AI model providers from consumers
- Centralizes complex integration logic
- Simplifies adding new AI models or tools

Implementation Example:

TypeScript

```
// Mediator for AI model orchestration
class AIModelMediator {
  private models: Map<string, AIModel> = new Map();
  private tools: Map<string, Tool> = new Map();
  registerModel(name: string, model: AIModel): void {
    this.models.set(name, model);
  }
  registerTool(name: string, tool: Tool): void {
    this.tools.set(name, tool);
  }
  async executeWorkflow(workflow: Workflow, context: Context):
Promise<Result> {
    // Orchestrate complex interactions between models and tools
    // Handle context retention, tool access security, etc.
    const model = this.models.get(workflow.modelName);
    const requiredTools = workflow.steps
      .flatMap(step => step.tools)
      .map(toolName => this.tools.get(toolName));

    return await model.run(context, requiredTools);
  }
}
```

2. Command Pattern for AI Agent Actions

Since AI agents need to interact with external tools and resources,

the Command pattern would help encapsulate these actions.

Benefits:

- Standardizes tool interactions
- Enables undo/retry functionality
- Simplifies logging and auditing of agent actions

Implementation Example:

TypeScript

```typescript
// Command interface
interface ToolCommand {
  execute(context: Context): Promise<Result>;
  validate(context: Context): boolean;
  undo?(): Promise<void>;
}
// Concrete command implementation
class SearchWebCommand implements ToolCommand {
  constructor(private query: string, private searchService: SearchService)
{}

  validate(context: Context): boolean {
    return context.hasPermission('web_search') && this.query.length > 0;
  }

  async execute(context: Context): Promise<Result> {
    context.log(`Executing web search for: ${this.query}`);
    return await this.searchService.search(this.query);
  }
}
// Usage in agent
const command = new SearchWebCommand("AI design patterns", searchService);
if (command.validate(agentContext)) {
  const results = await command.execute(agentContext);
}
```

3. Observer Pattern for Observability

The section on Observability Benefits suggests implementing the Observer pattern for real-time monitoring.

Benefits:

- Decouples monitoring from core functionality
- Enables multiple monitoring systems without code changes
- Supports real-time alerting and dashboards

Implementation Example:

TypeScript

```
// Subject interface
interface Observable {
  attach(observer: Observer): void;
  detach(observer: Observer): void;
  notify(event: Event): void;
}
// Observer implementation for monitoring
class PerformanceMonitor implements Observer {
  update(event: Event): void {
    if (event.type === 'execution_completed') {
      metrics.recordLatency(event.data.duration);
      metrics.recordTokenUsage(event.data.tokenCount);
    }
  }
}
// Usage in AI agent
class AIAgent implements Observable {
  private observers: Observer[] = [];

  attach(observer: Observer): void {
    this.observers.push(observer);
  }

  detach(observer: Observer): void {
    const index = this.observers.indexOf(observer);
    if (index !== -1) {
      this.observers.splice(index, 1);
    }
  }

  notify(event: Event): void {
    for (const observer of this.observers) {
      observer.update(event);
    }
  }

  async processRequest(request: Request): Promise<Response> {
    const startTime = performance.now();
    // Process the request
    const response = await this.generateResponse(request);
    const endTime = performance.now();

    this.notify({
      type: 'execution_completed',
      data: {
        duration: endTime - startTime,
        tokenCount: response.tokenCount,
        modelName: this.modelName
      }
    });
```

```
    return response;
  }
}
```

4. Strategy Pattern for AI Model Selection

To support the flexibility in choosing AI models mentioned in the MCP Server benefits:

Benefits:

- Dynamically swap AI models based on needs
- Isolate model-specific code
- Simplify A/B testing of different models

Implementation Example:

TypeScript

```typescript
// Strategy interface
interface AIModelStrategy {
  generateResponse(prompt: string, context: Context): Promise<string>;
  estimateCost(prompt: string): number;
}
// Concrete strategies
class GPT4Strategy implements AIModelStrategy {
  async generateResponse(prompt: string, context: Context): Promise<string>
{
    // Implementation for GPT-4
  }

  estimateCost(prompt: string): number {
    return prompt.length * 0.0001; // Example rate
  }
}
class ClaudeStrategy implements AIModelStrategy {
  async generateResponse(prompt: string, context: Context): Promise<string>
{
    // Implementation for Claude
  }

  estimateCost(prompt: string): number {
    return prompt.length * 0.00008; // Example rate
  }
}
// Context class that uses the strategy
class AIResponseGenerator {
  constructor(private strategy: AIModelStrategy) {}

  setStrategy(strategy: AIModelStrategy): void {
    this.strategy = strategy;
  }

  async generateResponse(prompt: string, context: Context): Promise<string>
{
    return await this.strategy.generateResponse(prompt, context);
  }
}
```

5. Chain of Responsibility for Security Validation

Given my DevSecOps concerns:

Benefits:

- Separates security checks from business logic
- Makes it easy to add/modify security rules
- Provides clear audit trail of security validations

Implementation Example:

TypeScript

```
abstract class SecurityHandler {
  private nextHandler: SecurityHandler | null = null;

  setNext(handler: SecurityHandler): SecurityHandler {
    this.nextHandler = handler;
    return handler;
  }

  async handle(request: Request): Promise<SecurityValidationResult> {
    const result = await this.validate(request);

    if (!result.passed || !this.nextHandler) {
      return result;
    }

    return this.nextHandler.handle(request);
  }

  protected abstract validate(request: Request):
Promise<SecurityValidationResult>;
}
class AuthenticationValidator extends SecurityHandler {
  protected async validate(request: Request):
Promise<SecurityValidationResult> {
    // Check authentication tokens
    if (!request.authToken || !await this.verifyToken(request.authToken)) {
      return {
        passed: false,
        reason: 'Invalid authentication token',
        riskLevel: 'high'
      };
    }
    return { passed: true };
  }

  private async verifyToken(token: string): Promise<boolean> {
    // Token verification logic
    return true; // Simplified for example
  }
}
// Usage
const securityChain = new AuthenticationValidator();
securityChain
  .setNext(new PermissionValidator())
  .setNext(new RateLimitValidator())
  .setNext(new ContentSafetyValidator());
const validationResult = await securityChain.handle(request);
if (validationResult.passed) {
  // Process the request
} else {
```

```
  // Handle security violation
}
```

These patterns would help address the architectural needs outlined in my

Penumbra Agent Squad vision while maintaining compatibility with

my preferred tech stack (Next.js, React, etc.).

They would also support the goals of building AI agents that are maintainable, secure, and scalable.

Model Context Protocol (MCP) Architectural Patterns

Architectural recommendations to properly address the

Model Context Protocol (MCP) approach for AI agent development:

1. Protocol Adapter Pattern for MCP Implementation

Since MCP is about standardizing interfaces between AI models and applications,

the Adapter pattern would be ideal:

TypeScript

```
// Protocol interface defining the MCP standard
interface ModelContextProtocol {
  process(input: MCPRequest): Promise<MCPResponse>;
  loadContext(contextId: string): Promise<void>;
  saveContext(): Promise<string>;
  registerTools(tools: MCPTool[]): void;
}
// Adapter for GPT models to implement MCP
class OpenAIModelAdapter implements ModelContextProtocol {
  private model: OpenAIModel;
  private context: Map<string, any> = new Map();
  private tools: MCPTool[] = [];

  constructor(model: OpenAIModel) {
    this.model = model;
  }

  async process(input: MCPRequest): Promise<MCPResponse> {
    // Transform MCP request format to OpenAI-specific format
    const openAIRequest = this.transformRequest(input);

    // Process with the underlying model
    const openAIResponse = await this.model.complete(openAIRequest);

    // Transform back to MCP response format
    return this.transformResponse(openAIResponse);
  }

  async loadContext(contextId: string): Promise<void> {
    // Load context from storage
    this.context = await contextStore.get(contextId);
  }

  async saveContext(): Promise<string> {
    const contextId = generateUuid();
    await contextStore.set(contextId, this.context);
    return contextId;
  }

  registerTools(tools: MCPTool[]): void {
    this.tools = tools;
  }

  private transformRequest(mcpRequest: MCPRequest): OpenAIRequest {
    // Convert MCP format to OpenAI format
    // Include tools, context, etc.
    return {
      messages: mcpRequest.messages,
      tools: this.tools.map(tool => ({
        type: "function",
        function: {
```

```
          name: tool.name,
          description: tool.description,
          parameters: tool.parameters
        }
      })),
      // Add other OpenAI specific parameters
    };
  }

  private transformResponse(openAIResponse: any): MCPResponse {
    // Transform OpenAI response to standard MCP format
    return {
      content: openAIResponse.choices[0].message.content,
      toolCalls: openAIResponse.choices[0].message.tool_calls?.map(tc => ({
        name: tc.function.name,
        arguments: JSON.parse(tc.function.arguments)
      })) || [],
      usage: openAIResponse.usage
    };
  }
}
```

2. Facade Pattern for MCP Server

To simplify the complexity of managing multiple models, contexts, and tools:

TypeScript

```
class MCPServer {
  private modelAdapters: Map<string, ModelContextProtocol> = new Map();
  private activeContexts: Map<string, string> = new Map(); // sessionId ->
contextId
  private tools: Map<string, MCPTool> = new Map();

  registerModel(modelId: string, adapter: ModelContextProtocol): void {
    this.modelAdapters.set(modelId, adapter);
  }

  registerTool(tool: MCPTool): void {
    this.tools.set(tool.name, tool);
    // Update all model adapters with the new tool
    this.modelAdapters.forEach(adapter => {
      adapter.registerTools(Array.from(this.tools.values()));
    });
  }

  async processRequest(sessionId: string, modelId: string, request:
MCPRequest): Promise<MCPResponse> {
    const adapter = this.modelAdapters.get(modelId);
    if (!adapter) {
      throw new Error(`Model ${modelId} not registered`);
    }

    // Load context if exists
    const contextId = this.activeContexts.get(sessionId);
    if (contextId) {
      await adapter.loadContext(contextId);
    }

    // Process the request
    const response = await adapter.process(request);

    // Execute any tool calls
    if (response.toolCalls && response.toolCalls.length > 0) {
      response.toolResults = await
this.executeToolCalls(response.toolCalls);
    }

    // Save updated context
    const newContextId = await adapter.saveContext();
    this.activeContexts.set(sessionId, newContextId);

    return response;
  }

  private async executeToolCalls(toolCalls: MCPToolCall[]):
Promise<MCPToolResult[]> {
    return Promise.all(toolCalls.map(async call => {
      const tool = this.tools.get(call.name);
```

```
      if (!tool) {
        return {
          toolName: call.name,
          success: false,
          error: `Tool ${call.name} not found`
        };
      }

      try {
        const result = await tool.execute(call.arguments);
        return {
          toolName: call.name,
          success: true,
          result
        };
      } catch (error) {
        return {
          toolName: call.name,
          success: false,
          error: error.message
        };
      }
    }));
  }
}
```

3. Repository Pattern for Context Management

To handle the persistence and retrieval of conversation contexts efficiently:

TypeScript

```typescript
interface ContextRepository {
  save(context: any): Promise<string>;
  load(contextId: string): Promise<any>;
  delete(contextId: string): Promise<void>;
}
class RedisContextRepository implements ContextRepository {
  private client: Redis;
  private ttl: number; // Time-to-live in seconds

  constructor(client: Redis, ttlSeconds = 3600) {
    this.client = client;
    this.ttl = ttlSeconds;
  }

  async save(context: any): Promise<string> {
    const contextId = uuidv4();
    await this.client.set(
      `mcp:context:${contextId}`,
      JSON.stringify(context),
      'EX',
      this.ttl
    );
    return contextId;
  }

  async load(contextId: string): Promise<any> {
    const data = await this.client.get(`mcp:context:${contextId}`);
    if (!data) {
      throw new Error(`Context ${contextId} not found`);
    }
    return JSON.parse(data);
  }

  async delete(contextId: string): Promise<void> {
    await this.client.del(`mcp:context:${contextId}`);
  }
}
```

4. Strategy Pattern for Tool Execution

For handling different tools with a unified interface:

TypeScript

```
interface MCPTool {
  name: string;
  description: string;
  parameters: JSONSchema;
  execute(args: any): Promise<any>;
}
// Web search tool implementation
class WebSearchTool implements MCPTool {
  name = "web_search";
  description = "Search the web for current information";
  parameters = {
    type: "object",
    properties: {
      query: {
        type: "string",
        description: "The search query"
      },
      numResults: {
        type: "number",
        description: "Number of results to return",
        default: 5
      }
    },
    required: ["query"]
  };

  constructor(private searchProvider: SearchProvider) {}

  async execute(args: { query: string, numResults?: number }): Promise<any>
{
    const results = await this.searchProvider.search(
      args.query,
      args.numResults || 5
    );

    return {
      results: results.map(r => ({
        title: r.title,
        url: r.url,
        snippet: r.snippet
      }))
    };
  }
}
// Database query tool implementation
class DatabaseQueryTool implements MCPTool {
  name = "db_query";
  description = "Query the application database";
  parameters = {
    type: "object",
    properties: {
```

```
      sql: {
        type: "string",
        description: "SQL query to execute (read-only operations only)"
      }
    },
    required: ["sql"]
  };

  constructor(private dbConnection: DatabaseConnection) {}

  async execute(args: { sql: string }): Promise<any> {
    // Validate query is read-only for security
    if (!/^SELECT\s/i.test(args.sql)) {
      throw new Error("Only SELECT queries are allowed");
    }

    const results = await this.dbConnection.query(args.sql);
    return { rows: results };
  }
}
```

5. Decorator Pattern for MCP Security and Observability

To add security checks and observability without modifying the core MCP implementation:

TypeScript

```
class SecureModelDecorator implements ModelContextProtocol {
  constructor(
    private wrappedModel: ModelContextProtocol,
    private securityService: SecurityService
  ) {}

  async process(input: MCPRequest): Promise<MCPResponse> {
    // Security pre-checks
    await this.securityService.validateRequest(input);

    // Process with the underlying model
    const response = await this.wrappedModel.process(input);

    // Security post-checks on the response
    await this.securityService.validateResponse(response);

    return response;
  }

  async loadContext(contextId: string): Promise<void> {
    // Verify access to this context
    await this.securityService.validateContextAccess(contextId);
    return this.wrappedModel.loadContext(contextId);
  }

  async saveContext(): Promise<string> {
    return this.wrappedModel.saveContext();
  }

  registerTools(tools: MCPTool[]): void {
    // Validate tool permissions
    const allowedTools = tools.filter(tool =>
      this.securityService.isToolAllowed(tool.name)
    );
    this.wrappedModel.registerTools(allowedTools);
  }
}
class ObservableModelDecorator implements ModelContextProtocol {
  constructor(
    private wrappedModel: ModelContextProtocol,
    private metrics: MetricsService
  ) {}

  async process(input: MCPRequest): Promise<MCPResponse> {
    const startTime = performance.now();

    try {
      const response = await this.wrappedModel.process(input);

      // Record metrics
      const duration = performance.now() - startTime;
```

```
      this.metrics.recordLatency('model.process', duration);
      this.metrics.incrementCounter('model.requests.success');
      this.metrics.recordTokenUsage(
        response.usage?.promptTokens || 0,
        response.usage?.completionTokens || 0
      );

      return response;
    } catch (error) {
      this.metrics.incrementCounter('model.requests.error');
      throw error;
    }
  }

  async loadContext(contextId: string): Promise<void> {
    const startTime = performance.now();

    try {
      await this.wrappedModel.loadContext(contextId);
      const duration = performance.now() - startTime;
      this.metrics.recordLatency('context.load', duration);
    } catch (error) {
      this.metrics.incrementCounter('context.load.error');
      throw error;
    }
  }

  async saveContext(): Promise<string> {
    const startTime = performance.now();

    try {
      const contextId = await this.wrappedModel.saveContext();
      const duration = performance.now() - startTime;
      this.metrics.recordLatency('context.save', duration);
      return contextId;
    } catch (error) {
      this.metrics.incrementCounter('context.save.error');
      throw error;
    }
  }

  registerTools(tools: MCPTool[]): void {
    this.metrics.recordHistogram('tools.registered', tools.length);
    this.wrappedModel.registerTools(tools);
  }
}
```

Usage Example

TypeScript

```
// Set up the MCP Server
const mcpServer = new MCPServer();
// Create base model adapters
const gpt4Adapter = new OpenAIModelAdapter(new OpenAIModel('gpt-4'));
const claudeAdapter = new AnthropicModelAdapter(new AnthropicModel('claude-
3-opus'));
// Apply decorators for security and observability
const secureGpt4 = new SecureModelDecorator(gpt4Adapter, securityService);
const observableSecureGpt4 = new ObservableModelDecorator(secureGpt4,
metricsService);
const secureClaudeAdapter = new SecureModelDecorator(claudeAdapter,
securityService);
const observableSecureClaudeAdapter = new
ObservableModelDecorator(secureClaudeAdapter, metricsService);
// Register models with the MCP server
mcpServer.registerModel('gpt-4', observableSecureGpt4);
mcpServer.registerModel('claude-3', observableSecureClaudeAdapter);
// Register tools
mcpServer.registerTool(new WebSearchTool(searchProvider));
mcpServer.registerTool(new DatabaseQueryTool(dbConnection));
mcpServer.registerTool(new CalendarTool(calendarService));
// Use in API endpoint
app.post('/api/chat', async (req, res) => {
  const { sessionId, message, modelId = 'gpt-4' } = req.body;

  try {
    const response = await mcpServer.processRequest(
      sessionId,
      modelId,
      {
        messages: [{ role: 'user', content: message }]
      }
    );

    res.json(response);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});
```

These architectural patterns align perfectly with the Model Context Protocol concept They provide a robust foundation for building AI agent systems with

- standardized interfaces
- efficient context management
- secure tool execution
- comprehensive observability.

GenaRoo

The Idea is to have my implementation of Roo Code in our staging environment

- Detailed logs of development steps
- Easily Onboard any team member at any stage of development lifecycle
- More contribution from less tecnhical team members is possible

Ruru & Roo

Page

DOING

▢ Fix Org ID Bug with Supabase Login

Diagnostic

Page

Supabase Org Diagnostic

Explanation

The lint item indicates that the `public.organization_members` table in the public schema

- does not have Row Level Security (RLS) enabled.

This is a security concern because

it allows unrestricted access to the data in this table,

which can lead to unauthorized data exposure.

Suggested Fixes:

1. Enable RLS on the Table:

- You need to enable RLS for the `public.organization_members` table.
- This can be done with the following SQL command:

```sql
ALTER TABLE public.organization_members ENABLE ROW LEVEL SECURITY;
```

1. Create RLS Policies:

- After enabling RLS,
  - you will need to create appropriate RLS policies to define
    - who can access or modify the data in the `organization_members` table.
- This will depend on your application's requirements.
- For example, you might want to allow only the owner of an organization to access their members.

2. Review Access Requirements:

- Ensure that the policies you create
  - align with your application's access control requirements.
- You may need to create multiple policies for
  - different actions (SELECT, INSERT, UPDATE, DELETE).

3. Test the Policies:

- After implementing the policies,
- thoroughly test them to ensure they
  - work as intended and that
  - users can only access the data they are authorized to see.

Secure the `public.organization_members` table

and ensure that it adheres to best practices for data security.

Solution

Page

Roo Commander

---

1 Plan

Plan

1. Address Supabase CLI Connectivity Issues:

- The initial attempts to pull the schema and data failed due to connectivity issues with the local PostgreSQL database.

2.Enable RLS Policies:

- I will generate the necessary SQL commands to enable RLS on the tables listed in the Supabase Advisor output.

3.Apply the Changes:

- I will use the Supabase CLI to apply these changes to the remote database.

4.Verify the Fix:

- After applying the changes, I will run the Supabase Advisor again to confirm that the RLS errors have been resolved.

5.Update Project Journal:

- I will update the project journal with the details of the bug fix.

Action

Resolve Supabase CLI

```bash
brew services list
```

- Postgresql 14 & 16 in Error State

Start Supabase

```bash
brew services start postgresql@16
```

- Command Failed

Try as Root

```bash
sudo brew services start postgresql@16
```

- Success

Pull Schema from remote

```bash
supabase db pull
```

- Command Failed

Run `supabase migration repair`

```bash
supabase migration repair --status reverted 20250426044802
```

```bash
supabase migration repair --status reverted 20250426044824
```

```bash
supabase migration repair --status reverted 20250426045200
```

Bash

```
supabase migration repair --status reverted 20250426045200
```

Implement RLS

SQL

```
Commands to enable RLS on the tables:

ALTER TABLE public.api_keys ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.chat ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.document ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.message ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.suggestion ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.vote ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.organizations ENABLE ROW LEVEL SECURITY;
ALTER TABLE public.organization_members ENABLE ROW LEVEL SECURITY;
```

RLS Policies (organizations Table)

Bash

```
SQL commands to create basic RLS policies for the organizations table:
```

RLS Polices (organization_members table)

SQL

```
CREATE POLICY "Enable read access for all users" ON
public.organization_members FOR SELECT USING (true);

CREATE POLICY "Enable insert for authenticated users only" ON
public.organization_members FOR INSERT WITH CHECK (auth.role() =
'authenticated');

  CREATE POLICY "Enable delete for users based on user_id" ON
public.organization_members FOR DELETE USING (auth.uid() = user_id);
```

Implement via Supabase CLI

SQL

```
supabase db sql -- "ALTER TABLE public.api_keys ENABLE ROW LEVEL SECURITY;"
```

- Command Failed

Search for Docs

```xml
<fetch>
<url>https://supabase.com/docs/reference/cli/supabase-db-sql</url>
</fetch>
```

- Failed - requires URL

Brave Search

```json
{
"query": "supabase cli execute sql query"
}
```

Response

XML

Title: Execute SQL in Supabase API · supabase · Discussion #3419
Description: We do not allow direct execution of SQL from our APIs. There are two possible solutions: You can create a function with the query specified and make RPC calls as such or · You can also generate a view with the query specified, then make a SELECT from there
URL: https://github.com/orgs/supabase/discussions/3419

Title: Supabase CLI | Supabase Docs
Description: The Supabase CLI provides tools to develop your project locally, deploy to the Supabase Platform, and set up CI/CD workflows.
URL: https://supabase.com/docs/guides/local-development/cli/getting-started

Title: Database Functions | Supabase Docs
Description: Supabase provides several options for creating database functions. You can use the Dashboard or create them directly using SQL. We provide a SQL editor within the Dashboard, or you can connect to your database and run the SQL queries yourself. Go to the "SQL editor" section. Click "New Query".
URL: https://supabase.com/docs/guides/database/functions

Title: r/Supabase on Reddit: how to use a simple sql query with the supabase ORM
Description: Hello all. I have been banging away at this all afternoon, I'd appreciate some help. I am trying to duplicate the functionality seen in this simple…
URL: https://www.reddit.com/r/Supabase/comments/1dxrkm8/how_to_use_a_simple_sql_query_with_the_supabase/

Title: r/Supabase on Reddit: Custom SQL queries with supabase client
Description: Does or will the supabase client support custom SQL calls? I find certain requests much easier to do in plain SQL and I was wondering if it is…
URL: https://www.reddit.com/r/Supabase/comments/1233jdd/custom_sql_queries_with_supabase_client/

Title: running raw sql with js library · supabase · Discussion #3458
Description: Is there a way to simply run raw sql against the database rather than using from("tablename").select etc. I can't find it in the docs. Thanks.
URL: https://github.com/orgs/supabase/discussions/3458

Title: Local Development & CLI | Supabase Docs
Description: Learn how to develop locally and use the Supabase CLI
URL: https://supabase.com/docs/guides/local-development

Title: Debugging and monitoring | Supabase Docs
Description: You can examine your database and queries for

these issues using either the Supabase
CLI or SQL.
URL: https://supabase.com/docs/guides/database/inspect

Title: shell - Can't run SQL scripts in Supabase while installing Quivr for first time - Stack Overflow
Description: Is there something I am getting wrong here? How can I perform step 4 properly and run the SQL queries in Supabase? ... And then click "Run".
URL: https://stackoverflow.com/questions/76757276/cant-run-sql-scripts-in-supabase-while-installing-quivr-for-first-time

Title: How to run custom SQL queries using functions in Supabase - DEV Community
Description: Recently I needed to run custom SQL functions in a Supabase project. Their Javascript SDK doesn't...
URL: https://dev.to/razvanstatescu/how-to-run-custom-sql-queries-using-functions-in-supabase-2nna

---

▢ Implement Grafana / Prometheus

GitHub - supabase/supabase-grafana: Observability for your Supabase project, using Prometheus/Grafan...
https://github.com/supabase/supabase-grafana

☑ Finish Roomodes

Roo Commander

GitHub - jezweb/roo-commander: Think of it like having a virtual, specialized software development t...
https://github.com/jezweb/roo-commander

Roo Micromanager

RooCodeMicroManager/CoderShortRules.md at main · adamwlarson/RooCodeMicroManager
https://github.com/adamwlarson/RooCodeMicroManager/blob/main/CoderShortRules.md

Handoffs Manager & Tips n Tricks

GitHub - Michaelzag/RooCode-Tips-Tricks
https://github.com/Michaelzag/RooCode-Tips-Tricks/tree/main

ConPort

context-portal/README.md at main · GreatScottyMac/context-portal
https://github.com/GreatScottyMac/context-portal/blob/main/README.md

Fix Agent Zero

Set Circular Graph Layout as Default

Deploy Example to DevSandbox

DevSandbox SDK

CodeSandbox SDK – CodeSandbox
https://codesandbox.io/docs/sdk#quickstart

DevSandbox Docs

Overview – CodeSandbox
https://codesandbox.io/docs/learn

Ideas

- Deploy Agent Zero
- Deploy SOW Agents
- Deploy Reddit Research MCP
- Deploy Roo Modes

Statement of Work Conversational Agent Flow

Conversational SOW generator Page

CLOUDFLARE Deploy to GCP & Collaborative / staging space for Reddit Research MCP Server

Build software better, together
https://github.com/penumbra-systems/mcp/tree/main/reddit-research-mcp-server

Deliverables

- Restrict User access by domain
- Connect to Claude Desktop
- Seamless Claude Teams Project to Desktop Integration

Knowledge Context

Page

00 - Penumbra - Knowledge OS

penumbra Strategic Thinking SystemArchitecture

System Overview

KnowledgeContext

- is an adaptive context management system
- for professional workflows,
- implementing the Model Context Protocol (MCP)
- with advanced knowledge retrieval techniques.

The system currently operates as a Node.js (Next.js) application

- *We could get an quick win by refactoring to a modular codebase*
- *Implemented as a monrepo using Turburepo*

It could leverage

- *TursoDB*
- *(or similar SQL database, e.g. Neon)*
- *as the persistent store*
- *for all context metadata and activity logs*
- *within a specific workspace.*

Key Differentiators:

Semantic Retrieval Engine:

- Combines keyword analysis
- concept graphs
- and metadata relationships
- without relying on vector embeddings (ToC is High)

Content Processing

Format-aware processing for

- documents, PDFs, emails, and web content
- Concept extraction with domain-specific weighting
- Semantic chunking preserving contextual boundaries
- Domain-specific terminology detection

Context Management

- Knowledge entity indexing and relationship mapping
- Conversation topic segmentation with intent detection
- Timeline tracking of key events and decisions

- Focus area identification based on user activity

Pattern Recognition

- Workflow pattern identification from user actions
- Automatic pattern learning from repeated operations
- Cross-project pattern generalization
- Process optimization detection

Relevance Analysis

- Query intent prediction across domains
- Multi-source context prioritization
- Attention budget management
- Context preservation during task switching

Core Components

Intelligent Prioritization:

- recency, importance, relationships, and user focus areas
- Multi-factor relevance scoring using

Workflow-Centric Organization:

- Contextual understanding based on
- projects, tasks, and collaborations

  (is collaboration different than task?)

Domain-Agnostic Design:

- Adaptable to various professional domains
- e.g., legal, research, business analysis, etc.)

Cross-Format Understanding:

- Processes documents
- emails
- spreadsheets,
- and multimedia content

MCP Tools

- KnowledgeContext implements the following MCP tools:

initialize_workspace_context

- Establishes a new collaboration session with relevant knowledge context.

When to use:

- At the beginning of any work session.

Key parameters:

initialQuery:

- User's starting question or task

contextScope:

- Personal, team, or organization-wide context

includeResources:

- Attach relevant files/documents

focusHint:

- Optional area of emphasis

Returns:

- Session ID and initial context overview

---

update_work_context

- Updates active context with new information and changes.

When to use:

- After significant changes or new information arrives.

Key parameters:

sessionId:

- From initialize_workspace_context

newContent:

- Added documents/messages

modifiedAssets:

- Updated files/resources

contextContinuity:

- Maintain thread through changes

Returns:

- Updated focus areas and context links

---

retrieve_task_context

- Fetches context relevant to a specific task or query.

When to use:

- When needing reference material for decision-making.

Key parameters:

sessionId:

- Current work session

query:

- Specific information need

filters:

- Type, source, or date constraints

priorityStrategy:

- Relevance weighting approach

Returns:

- Context excerpts with source attribution

---

record_knowledge_milestone

- Captures important decisions or progress points.

When to use:

- After completing key deliverables or making significant decisions.

Key parameters:

sessionId:

- Current work session

milestoneType:

- Decision, deliverable, insight

impactAssessment:

- Expected outcomes

relatedAssets:

- Connected files/artifacts

Returns:

- Milestone ID and verification checks

---

finalize_work_context
Concludes a session, extracting insights and action items.

When to use: At end of work session or task completion.

Key parameters:

- sessionId: Active session ID
- extractInsights: Identify key learnings
- updatePatterns: Refresh organizational knowledge
- generateActions: Create follow-up tasks

Returns: Session summary, insights, and next steps

Data Architecture

Core SQL database tables:

knowledge_assets: Stores documents, emails, and digital artifacts
asset_keywords: Concept mapping for search
asset_relationships: Connections between resources
work_sessions: Records collaboration history
work_topics: Categorizes discussion threads
timeline_events: Logs significant events
work_patterns: Stores identified workflow patterns
focus_areas: Tracks attention distribution

Technical Specifications

Node.js: Version 20.0.0+

Database: TursoDB/SQLite
Bundling: ESBuild
Protocol: Model Context Protocol
Parsing: Unified content extraction (text, PDF, HTML)
Platforms: Cross-platform support

Performance Enhancements

Hybrid caching strategy for frequent assets
Incremental context updates
Background knowledge indexing
Adaptive resource loading based on priority
Batch processing for large datasets
Asynchronous operation pipeline

Security

Workspace isolation
Role-based access control
End-to-end encryption option
Strict input validation
Audit logging
Local processing architecture

Page

Ruru 1

Markdown

```
graph LR
    subgraph PROJECT STRUCTURE
        A[ .roo file (Configuration) ]
        B[ .ruru file (Runtime) ]
    end

    A -- Defines what MCP servers are available --> C[ Roo Commander
(v7.1.5) ]
    B -- Executes tasks based on .roo configuration --> C

    C -- Uses configurations --> Services
    D[Services (postgres, etc)]

    subgraph .roo Details
        A1[Defines project structure & services]
        A2[Specifies available MCP servers]
        A3[Contains settings for external services (Supabase)]
        A4[Serves as the project blueprint]
    end

    subgraph .ruru Details
        B1[Reads & interprets .roo configuration]
        B2[Executes specific tasks]
        B3[Handles command execution]
        B4[Manages connection between code & services]
    end

    A --> A1
    A --> A2
    A --> A3
    A --> A4

    B --> B1
    B --> B2
    B --> B3
    B --> B4

    style PROJECT STRUCTURE fill:#f9f,stroke:#333,stroke-width:2px
    style A fill:#ccf,stroke:#333,stroke-width:1px
    style B fill:#ccf,stroke:#333,stroke-width:1px
    style C fill:#ccf,stroke:#333,stroke-width:1px
    style D fill:#ccf,stroke:#333,stroke-width:1px
```

A Comprehensive Tutorial on the Five Levels of Agentic AI Architectures: From Basic Prompt Responses...

https://www.marktechpost.com/2025/04/25/a-comprehensive-tutorial-on-the-five-levels-of-agentic-ai-architectures-from-basic-prompt-responses-to-fully-autonomous-code-generation-and-execution/

BlackNoir

- BlackNoir Fundamentals and Methodologies — Page
- Systematic Design — PRD
- BlackNoir as Code — Page

- BlackNoir Color Design Apps — Page
- BlackNoir Systematic Design — Page

Review Ops OS & Sync With Andrew

Ops OS Google Doc

Google Docs: Sign-in
https://docs.google.com/document/d/1-YSFSX2-XpD5AAKcgK429JOGUp8heE6GSGb6sHO1L3k/edit?tab=t.0

Implement MCP Inspector

GitHub - modelcontextprotocol/inspector: Visual testing tool for MCP servers
https://github.com/modelcontextprotocol/inspector

Containerization

Docker

Docker Compose

Docker Bake

CI/CD

Migrate Neo4J

Google Cloud console

https://console.cloud.google.com/marketplace/product/endpoints/prod.n4gcp.neo4j.io?inv=1&invt=AbxUOQ&project=goose-451110&chat=true

Nitric & Uffizzi

Nitric Cloud-Native Framework | Get Infrastructure from Code | Nitric
https://nitric.io/

Uffizzi GitHub Action

Preview Environments - GitHub Marketplace
https://github.com/marketplace/actions/preview-environments

Uffizzi Docs

Uffizzi app
https://app.uffizzi.com/projects

Uffizzi Docs

Uffizzi – Uffizzi
https://docs.uffizzi.com/

Web Search, Crawl & Auto ETL

Crawl4AI

Home - Crawl4AI Documentation
https://crawl4ai.com/mkdocs/

LightRAG

LightRAG/lightrag/api/README.md at main · HKUDS/LightRAG
https://github.com/HKUDS/LightRAG/blob/main/lightrag/api/README.md

Browser Control, Open Manus, Open Operator, Browser User, Proxy Light, HyperBrowser, etc

Browser Use

GitHub - browser-use/web-ui: Run AI Agent in your browser.
https://github.com/browser-use/web-ui

Browser Use Web UI

Introduction - Browser Use
https://docs.browser-use.com/introduction

Open Manus

GitHub - mannaandpoem/OpenManus: No fortress, purely open ground. OpenManus is Coming.
https://github.com/mannaandpoem/OpenManus

Browser Base

GitHub - browserbase/open-operator
https://github.com/browserbase/open-operator

Proxy Lite

GitHub - convergence-ai/proxy-lite: A mini, open-weights, version of our Proxy assistant.
https://github.com/convergence-ai/proxy-lite/tree/main

HyperBrowser

Welcome to Hyperbrowser | Hyperbrowser
https://docs.hyperbrowser.ai/

BrowserBase

Browserbase - Headless Web Browser API
https://www.browserbase.com/overview

Open Computer Agent

Computer Agent - a Hugging Face Space by smolagents
https://huggingface.co/spaces/smolagents/computer-agent

▢ XPRT Commitzen

▢ Cloudflare (remember / find out what Was to do on Cloudflare)??

Cloudflare Dashboard

Cloudflare Dashboard | Manage Your Account
https://dash.cloudflare.com/4e167a44613ed79f649b6e7d2e27d11c/home/developer-platform

Vercel Dashboard

Dashboard
https://vercel.com/penumbra-systems/penumbra-v2/logs?timeline=maximum&page=17&selectedLogId=4fs7t-1747039202081-4c19347f4116&levels=error

---

UX

Page

Penumbra knowledge graph platform beta feedback with Hugo

Shep Bryan

Wed, Apr 30 10:15 AM

Platform Overview & Current State

- Penumbra is an AI Acceleration Lab platform currently in pre-product market fit stage
- Currently offers web app with chat and graph functionality
- Document upload workflow automatically reverse engineers content into graph nodes
- Uses semantic knowledge graph approach - smaller but more specific to meaning/context
- Currently working on adding web search functionality to match baseline ChatGPT/Claude features

- Building extensible architecture to support multiple use cases

Questions About the Platform

- **How to audit** that the **platform** makes **correct decisions** when *creating* the **graph**
- What is the **best practice** for knowledge graph **size** and **performance**?
- How to make knowledge graphs **performant** for *specific jobs?*
- Should nodes be excluded or trimmed for optimal performance?
- How to effectively curate knowledge graphs?

What Impressed the User About the Tool

- Ability to easily create and structure knowledge graphs
- Document upload and automatic graph creation functionality
- Semantic approach to knowledge organization
- Circle layout visualization was effective for viewing complex graphs
- Successfully used graph to create article responses
- Converting documents and content into "brain layer" format

Ideas for the Platform

- Add **staging** area for **reviewing** *extracted content* before *adding* to **graph**
- Make **circle layout** the *default view* **instead** of *hierarchical*
- Improve **node explorer** with *better search* and **direct node interaction**
- Enable **brain duplication** with *key variations*
- Create **marketplace** for **sharing/accessing** *others'* **brain models**
- Build **interviewing system** for *modeling* real human users as non-synthetic **digital humans**
- Enable **multiple graphs** for *different* **domains** (AI, business, personal)
- Add **multiplayer** functionality for team **collaboration**
- Implement system for **mentioning** *specific* **nodes** or **graph sections**
- Add **tutorials** and **documentation** for **best practices**

Future Vision & Use Cases

- Use for mapping complex system problems (climate change, vaccine distribution)
- Model decision-making across large organizational systems
- Create personalized AI tools powered by individual knowledge graphs
- Enable scenario planning and system design for complex problems
- Support meta-system thinking and "make things that make things" approach
- Help coordinate large-scale multi-stakeholder projects

- Build apps powered by different specialized brain models

Reaction

- HITL → xyFlow Node with no edges until approval
- Notion Like editor for quick edits

- Review Visualizers Section for alternate solutions

GitHub - totogo/awesome-knowledge-graph: A curated list of Knowledge Graph related learning material...
https://github.com/totogo/awesome-knowledge-graph

- Research needed for node explorer / search improvements

- Research needed for brain duplication

- Research needed for marketplace implementation

- AG2 for Conversational Interface (Interviewing System)

AG2 Docs

Getting Started - AG2
https://docs.ag2.ai/docs/Getting-Started

AG2 FastAgency

GitHub - ag2ai/fastagency: The fastest way to bring multi-agent workflows to production.
https://github.com/ag2ai/fastagency?tab=readme-ov-file

AG2 XPRT Agent Definitiions

AG2 Agent Definitions Page

LangChain

LangChain
https://www.langchain.com/

LangGraph

LangGraph
https://langchain-ai.github.io/langgraph/

LangMem

Introduction
https://langchain-ai.github.io/langmem/

Langfuse

Langfuse
https://langfuse.com/

Langfuse Docs

Langfuse Documentation - Langfuse
https://langfuse.com/docs

LangChain CLI

CLI
https://langchain-ai.github.io/langgraph/cloud/reference/cli/

LangGraph Studio

Overview
https://langchain-ai.github.io/langgraph/concepts/langgraph_studio/

LangFuse

LangSmith

LangSmith
https://smith.langchain.com/#

Open Canvas

GitHub - langchain-ai/open-canvas: A better UX for chat, writing content, and coding with LLMs.
https://github.com/langchain-ai/open-canvas

OpenEvals

GitHub - langchain-ai/openevals: Readymade evaluators for your LLM apps
https://github.com/langchain-ai/openevals?tab=readme-ov-file#multiturn-simulation

---

Design

W3

W3C Accessibility Guidelines (WCAG) 3.0
https://www.w3.org/TR/wcag-3.0/#consistency-across-views

W3C
https://www.w3.org/

Resources

Data

Turso

Welcome to Turso - Turso
https://docs.turso.tech/introduction

DataStax Astra

Datastax Astra
https://astra.datastax.com/org/67b88500-04ee-48b8-a773-eabdc1459116/database/d58794ab-d0d2-4e27-8bad-50198b37ca67

Memory

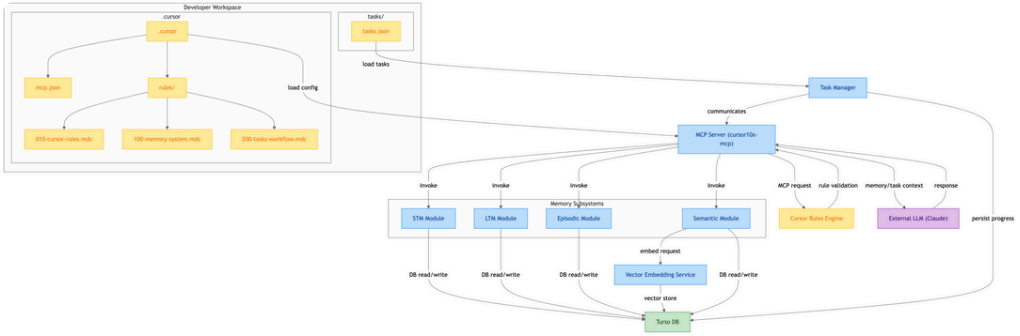
DevContext

Untitled
https://github.com/aurda012/devcontext

Cursor10x (Multi Layer Memory)

Untitled
https://github.com/aurda012/cursor10x

Cursor 10x Diagram Image



Cursor 10x GitDiargram

Markdown

```
flowchart TB
    %% Developer Workspace
    subgraph "Developer Workspace"
        subgraph ".cursor"
            CursorDir[".cursor"]:::config
            MCPConfig["mcp.json"]:::config
            RulesFolder["rules/"]:::config
            CursorDir --> MCPConfig
            CursorDir --> RulesFolder
            Rule1["010-cursor-rules.mdc"]:::config
            Rule2["100-memory-system.mdc"]:::config
            Rule3["200-tasks-workflow.mdc"]:::config
            RulesFolder --> Rule1
            RulesFolder --> Rule2
            RulesFolder --> Rule3
        end
        subgraph "tasks/"
            TasksJson["tasks.json"]:::config
        end
    end

    %% Core Compute Components
    MCPServer["MCP Server (cursor10x-mcp)"]:::compute
    RulesEngine["Cursor Rules Engine"]:::config
    LLM["External LLM (Claude)"]:::external
    TaskManager["Task Manager"]:::compute
    subgraph "Memory Subsystems"
        STM["STM Module"]:::compute
        LTM["LTM Module"]:::compute
        Episodic["Episodic Module"]:::compute
        Semantic["Semantic Module"]:::compute
    end
    EmbeddingService["Vector Embedding Service"]:::compute

    %% Storage
    TursoDB["Turso DB"]:::storage

    %% Connections
    CursorDir -->|"load config"| MCPServer
    TasksJson -->|"load tasks"| TaskManager
    TaskManager -->|"communicates"| MCPServer
    MCPServer -->|"MCP request"| RulesEngine
    RulesEngine -->|"rule validation"| MCPServer
    MCPServer -->|"invoke"| STM
    MCPServer -->|"invoke"| LTM
    MCPServer -->|"invoke"| Episodic
    MCPServer -->|"invoke"| Semantic
    STM -->|"DB read/write"| TursoDB
    LTM -->|"DB read/write"| TursoDB
    Episodic -->|"DB read/write"| TursoDB
    Semantic -->|"embed request"| EmbeddingService
```

```
    EmbeddingService -->|"vector store"| TursoDB
    Semantic -->|"DB read/write"| TursoDB
    MCPServer -->|"memory/task context"| LLM
    LLM -->|"response"| MCPServer
    TaskManager -->|"persist progress"| TursoDB

    %% Click Events
    click MCPConfig
"https://github.com/aurda012/cursor10x/blob/main/.cursor/mcp.json"
    click CursorDir
"https://github.com/aurda012/cursor10x/tree/main/.cursor/"
    click Rule1
"https://github.com/aurda012/cursor10x/blob/main/.cursor/rules/010-
cursor-rules.mdc"
    click Rule2
"https://github.com/aurda012/cursor10x/blob/main/.cursor/rules/100-
memory-system.mdc"
    click Rule3
"https://github.com/aurda012/cursor10x/blob/main/.cursor/rules/200-
tasks-workflow.mdc"
    click TasksJson
"https://github.com/aurda012/cursor10x/blob/main/tasks/tasks.json"

    %% Styles
    classDef compute fill:#bbdefb,stroke:#1e88e5,color:#0d47a1;
    classDef storage fill:#c8e6c9,stroke:#388e3c,color:#1b5e20;
    classDef config fill:#ffeb99,stroke:#fbc02d,color:#f57f17;
    classDef external fill:#e1bee7,stroke:#8e24aa,color:#4a148c;
```

Awesome Knowledge Graph

GitHub - totogo/awesome-knowledge-graph: A curated list of Knowledge Graph related learning material...
https://github.com/totogo/awesome-knowledge-graph?tab=readme-ov-file

AI Dev KB

GitHub - jezweb/ai-dev-kb: Context files based on frameworks and technologies to use with Roo Code /...
https://github.com/jezweb/ai-dev-kb/tree/main

Graph Theory for debugging

Using Obscure Graph Theory to solve PL Problems :: Reasonably Polymorphic
https://reasonablypolymorphic.com/blog/solving-lcsa/

Real Time Knowledge Graphs

Build Real-Time Knowledge Graph For Documents with LLM | CocoIndex
https://cocoindex.io/blogs/knowledge-graph-for-docs/

Mem0

Mem0 - The Memory layer for your AI apps
https://mem0.ai/

Tools

Fetcher MCP

GitHub - jae-jae/fetcher-mcp: MCP server for fetch web page content using Playwright headless browse...
https://github.com/jae-jae/fetcher-mcp

MCP Aggregator

Untitled
https://github.com/ennwise/mcp-aggregator/tree/main

MCP.run

mcp.run
https://www.mcp.run/

SPARC Modes

NOTE

- More RooModes I didn't know about, by a thought leader to watch

This guide introduces Roo Code and the innovative Boomerang task concept, now integrated into SPARC ...
https://gist.github.com/ruvnet/a206de8d484e710499398e4c39fa6299

MCPX Eval

GitHub - dylibso/mcpx-eval: An open-ended eval framework for mcp.run tools
https://github.com/dylibso/mcpx-eval

Layered Tool Pattern

Block Engineering Blog
https://engineering.block.xyz/blog/build-mcp-tools-like-ogres-with-layers

llm-min.txt

GitHub - marv1nnnnn/llm-min.txt: Min.js Style Compression of Tech Docs for LLM Context
https://github.com/marv1nnnnn/llm-min.txt

YouWare

YouWare: The First Vibe Coding Community for AI Creators
https://www.youware.com/home

Lean AI Leaderboard

Official Lean AI Native Companies Leaderboard
https://leanaileaderboard.com/

Type Constrained Code Generation

Type-Constrained Code Generation with Language Models
https://arxiv.org/abs/2504.09246

How to SaaS

How to Start a SaaS Company in 2025: A Step-by-Step Guide for Founders
https://www.onsaas.me/blog/how-to-start-a-saas-company

Agent Frameworks

Mastra

NOTE

- Agent Framework

CHECK MY ASSUMPTION

- TypeScript is still less performant than Python / Rust ??

Mastra.ai
https://mastra.ai/

Granola Notes

NOTE

- Latecomer that caught the W with optimal UX

Granola — The AI notepad for people in back-to-back meetings
https://www.granola.ai/

Roadmap

Roadmap V1 — Page

Pydantic PRD — Page