Project

Penumbra Planning

- 1.4.1.3.3 Letta ADE
- 1.4.1.3.4 1.4.1.3.4 Letta Docs
- 1.4.1.4 1.4.1.4 ++ Agentic
- 1.4.1.4.1 1.4.1.4.1 Inngest
- 1.4.1.4.2 Inngest Agent Kit
- 1.4.1.4.2.1 Agent Kit Reddit Search BrowserBase
- 1.4.1.4.3 No Code / Low Code
- 1.4.1.4.4 Flowize AI
- 1.4.1.4.5 Langflow
- 1.4.1.4.6 1.4.1.4.6 Scrape & Crawl
- 1.4.1.4.6.1 LightRAG
- 1.4.2 1.4.2 Browser Control
- 1.4.2.1 Browser Use

1.5 1.5 Autonomous

- 1.5.1 Browser User
- 1.5.2 Proxy Lite
- 1.5.3 Open Manus
- 1.5.4 Open Operator
- 1.5.5 Steel
- 1.5.6 BrowserBase

1.6 1.6 Production

- 1.6.1 Cloudflare
- 1.6.1.1 1.6.1.1 DNS
- 1.6.1.2 Security / WAF

1.7 1.7 Stack As TODOs

- 1.7.1 (Inside) DevAgents & DevTools
- 1.7.1.1 System Architecture

1.8 Database

- 1.8.1 SpacetimeDB
- 1.8.2 Neon & Turso
- 1.8.2.1 Neon
- 1.8.3 Turso
- 1.8.4 Memory
- 1.8.5 1.8.5 GenUI
- 1.8.5.1 Crayon

- 1.9 Infra & IaC
- 1.10 1.10 DevSecOps & CI/CD
  - 1.10.1 1.10.1 Docker
    - 1.10.1.1 MCP Catalog & Toolkit
    - 1.10.1.2 GH Copilot
    - 1.10.1.3 Docker Compose
    - 1.10.1.4 Docker Bake
      - 1.10.1.4.1 Testcontainers Cloud
    - 1.10.1.5 1.10.1.5 k8s
    - 1.10.1.6 1.10.1.6 Dagger
    - 1.10.1.7 1.10.1.7 Dagger Agents
    - 1.10.1.8 1.10.1.8 Uffizzi
      - 1.10.1.8.1 Temporal
      - 1.10.1.8.2 GitHub Actions
      - 1.10.1.8.3 GitHub Dev Program
- 1.11 LLMetry & Ops
  - 1.11.1 Promptfoo
  - 1.11.2 Comet / Opik
    - 1.11.2.1 Opik MCP
    - 1.11.2.2 Opik1.11.2.2
    - 1.11.2.3 Opik GitHub
    - 1.11.2.4 Comet Docs
    - **1.11.2.5 Comet GitHub**
  - 1.11.3 AgentOps
  - 1.11.4 Agenta
    - 1.11.4.1 Langwatch
- 1.12 Service Mesh
  - 1.12.1 1.12.1 Kong API & AI
    - 1.12.1.1 Kuma
    - 1.12.1.2 Kong Gateway
    - 1.12.1.3 Portkey
    - 1.12.1.4 Not Diamond
- 1.13 LiteLLM
- 1.14 CMS
  - 1.14.1 Prismic

Penumbra
Market Research

Page

Competitive Analysis

Findings about competitors for Penumbra as a comprehensive comparison.

Penumbra Competitor Analysis:

AI-Powered Knowledge Management Applications

Competitive Landscape

Competitor Name	Rating	Key Strengths	Key Weaknesses	Additional Information
Notion AI	4.7/5	- Highly customizable workspace- Versatile for notes, tasks, and knowledge bases- Strong AI-assisted content creation- Excellent team collaboration features- Affordable pricing with free tier	- Search functionality limitations- Performance issues with large datasets- Mobile app usability concerns- Sometimes complex learning curve	- $8/month pe user for AI features- Wid adopted acros industries- Continuous feature improvements Strong integra ecosystem
Mem.ai	4.5/5	- Self-organizing note system-	- Limited integrations with	- Starting at $8.33/month-

		Excellent AI-powered search- Seamless knowledge connections- Fast and intuitive interface- Strong natural language processing	other tools- Still evolving platform with changes- Higher price point for full features- Limited customization options	Strong focus c personal knowledge management- Particularly gc for entreprene and executive AI-first approa to information retrieval
Guru	4.5/5	- Chrome extension for anywhere access- Strong verification workflows- Enterprise-grade knowledge sharing- Excellent integrations (Slack, Teams)- AI-driven search capabilities	- Steep learning curve for new users- Higher price point for full features- Requires continuous maintenance- Limited free tier functionality	- All-in-one pl $18/month- #' satisfaction fo knowledge management (G2- 98% satisfaction ra 50% faster implementatic than competit Strong focus c team knowled sharing
Qatalog	4.3/5	- Real-time data access without indexing- Strong security and compliance features- Search across structured/unstruc tured data- Permissions-based access controls- $26M in venture funding	- Relatively newer to market- Some users report confusion about functionality- Less established user community- Limited templates and resources	- Pro plan at $15/month pe user- 14-day f trial- Strong enterprise foc End-to-end encryption- IS 27001, SOC2-II, GDPR comp
Saner.ai	4.2/5	- Designed specifically for ADHD users- AI tag suggestion for organization- Chrome extension for research- Clean, focused interface- Natural language search	- Occasional technical issues- Requires internet access- Less established in enterprise settings- More limited feature set than competitors	- Starting at $7.89/month- Growing user base- Strong focus on indiv productivity- S by-side resea and note takir Position as nic tool for specif use cases

Strategy and Positioning Recommendations for Penumbra

1. Leverage Differentiation Opportunities

Based on the competitive analysis, Penumbra has several opportunities to differentiate in the AI knowledge management market:

Recommendation:

Position Penumbra as the most intelligent AI agent that truly understands context and user intent, going beyond simple keyword search or basic AI assistance.

- Focus on how Penumbra can anticipate user needs before they even ask.

2. Address Common Competitor Weaknesses

Several pain points emerge across competitors:

Recommendation:

Develop Penumbra with exceptional search capabilities that work flawlessly even with large datasets.

- Emphasize performance optimization and ensure the platform remains fast and responsive regardless of knowledge base size.

3. Hybrid Pricing Strategy

Pricing is a key competitive factor in this space:

Recommendation:

Offer a compelling free tier with meaningful functionality (unlike Guru's limited free version) to drive adoption, with a competitively priced premium tier around $10/month (positioning between Saner.ai and Notion, but below Guru and Qatalog).

- Consider a special "Founding Member" pricing for early adopters.

4. Focus on Integration Ecosystem

The ability to connect with existing tools is critical for knowledge management:

Recommendation:

Develop a robust API and integration ecosystem from day one, ensuring Penumbra works seamlessly with tools like Slack, Microsoft Teams, Google Workspace, and other productivity apps.

- Create a visual "knowledge graph" that shows connections between information across various platforms.

5. Balanced UI/UX Approach

User experience varies significantly across competitors:

Recommendation:

Strike the perfect balance between Notion's flexibility and Mem.ai's simplicity.

- Create an interface that adapts to the user's experience level, offering both guided experiences for beginners and advanced capabilities for power users.

6. Emphasize Privacy and Security

Enterprise concerns about AI and data security present an opportunity:

Recommendation:

Position Penumbra as the most secure and privacy-focused knowledge management solution.

Highlight how Penumbra can work with sensitive information without requiring data to leave the company's environment (similar to Qatalog's approach, but with more emphasis).

7. Vertical-Specific Solutions

Most competitors offer horizontal solutions:

Recommendation:

Develop specialized versions of Penumbra for specific industries like legal, healthcare, education, and software development with pre-built templates, workflows, and AI agents trained on industry-specific knowledge.

8. Community-Driven Development

Build an engaged user community:

Recommendation:

Create a vibrant community around Penumbra with user forums, template sharing, regular webinars, and a transparent product roadmap where users can vote on features.

- Position Penumbra as the tool that truly listens to its users.

9. Knowledge Graph Visualization

A feature missing from most competitors:

Recommendation:

Develop an innovative visualization interface that allows users to see connections between their knowledge in ways that other tools cannot, helping users discover insights they wouldn't otherwise find.

10. AI-First, But Human-Centered

Emphasize the balance of AI power with human needs:

Recommendation:

Position Penumbra as "AI-powered, human-centered" – the perfect balance of cutting-edge AI with thoughtful, human-friendly design.

- Focus on how Penumbra enhances human creativity and productivity rather than replacing human thinking.

Conclusion

The AI-powered knowledge management space is competitive but still evolving rapidly. By positioning Penumbra as a next-generation solution that addresses the core weaknesses of current offerings while introducing innovative features, there is a significant opportunity to capture market share. The key will be delivering a product that balances powerful AI capabilities with intuitive usability, strong privacy features, and compelling pricing.

Product / UX

▾ Example PRD → XPRT

PRD: XPRT Page

Requirements Gathering

Page

Requirements From Notion

---

7 Primitives

v5 beta prompt

You are a helpful assistant who is helping the user build a worldview graph – a specialized knowledge graph that models how the user thinks and strategizes. You help users externalize, structure, and leverage their worldview into a semantic knowledge graph using the tools available to you.

!Important

When the user initiates a chat, start by using the read_graph tool to pull the contents of their graph into your context window.

Your Role and Capabilities

As a Penumbra Assistant, you can use your available tools and critical thinking skills to:

- Call the available tools to structure the founder's worldview and it's supporting elements into a coherent and connected graph
- Reference the user's graph to in order to help them make decisions aligned with their worldview
- Help users discover insights and potential contradictions in their thinking

Key Concepts in Penumbra

Users

Users in Penumbra are primarily startup founders, business leaders, and strategic consultants.

Nodes

Nodes represent entities in the founder's cognitive and strategic system:

- **Actor**: Humans, AI agents, or organizational units with agency
- **Belief**: Propositional claims held to be true
- **Concept**: Terms or ideas in the founder's specific ontology
- **Document**: Sources, evidence, or artifacts
- **Founder Worldview**: The cognitive root of the founder or organization
- **Industry**: Business sectors providing market context
- **Internal Objective**: Specific goals or requirements within initiatives
- **Narrative**: Structured expressions of the worldview for different audiences
- **Operating Constraint**: Hard boundaries that define limits
- **Strategic Hypothesis**: Testable assumptions derived from beliefs
- **Strategic Initiative**: Concrete efforts that test hypotheses

Edges

Edges represent relationships between entities, organized by category:

- **Structural**: HAS_BELIEF, PART_OF_WORLDVIEW, MEMBER_OF, INITIATIVE_OF_WORLDVIEW
- **Causal**: LEADS_TO, CAUSES, PREVENTS
- **Dependency**: DEPENDS_ON, REQUIRES, ENABLES
- **Evidential**: JUSTIFIED_BY, SUPPORTS, CONTRADICTS, EVIDENCES_BELIEF
- **Associative**: RELATED_TO, SIMILAR_TO, RELATED_TO_BELIEF
- **Temporal**: PRECEDES, FOLLOWS
- **Agential**: CREATED_BY, MAINTAINED_BY, UNDERSTOOD_BY, OWNED_BY
- **Cognitive**: BELIEVES, CHALLENGES, EXPRESSES_BELIEF, UNDERPINS_HYPOTHESIS

Schemas

Schemas define the structure and properties of each node type, ensuring coherent representation of the founder's worldview.

Available Tools

Worldview Components

Founder Worldview Management

- `add_founder_worldview`: Define the central worldview
- `update_founder_worldview`: Update the worldview
- `delete_founder_worldview`: Remove a worldview

Belief Management

- `add_belief`: Add a foundational belief
- `update_belief`: Update a belief
- `delete_belief`: Remove a belief

Narrative Management

- `add_narrative`: Define a narrative
- `update_narrative`: Update a narrative
- `delete_narrative`: Remove a narrative

Operating Constraint Management

- `add_operating_constraint`: Define a boundary
- `update_operating_constraint`: Update a constraint
- `delete_operating_constraint`: Remove a constraint

Strategic Elements

Strategic Hypothesis Management

- `add_strategic_hypothesis`: Create a hypothesis
- `update_strategic_hypothesis`: Update a hypothesis
- `delete_strategic_hypothesis`: Remove a hypothesis

Strategic Initiative Management

- `add_strategic_initiative`: Define an initiative
- `update_strategic_initiative`: Update an initiative
- `delete_strategic_initiative`: Remove an initiative

Internal Objective Management

- `add_internal_objective`: Define an objective
- `update_internal_objective`: Update an objective
- `delete_internal_objective`: Remove an objective

Actor Management

Actor Management

- `add_actor`: Add a human, AI agent, or org unit
- `update_actor`: Update an actor
- `delete_actor`: Remove an actor

Thinking Infrastructure

Concept Management

- `add_concept`: Define a concept
- `update_concept`: Update a concept
- `delete_concept`: Remove a concept

Supporting Elements

Document Management

- `add_document`: Add a document
- `update_document`: Update a document
- `delete_document`: Remove a document

Industry Classification

- `add_industry`: Add an industry context
- `update_industry`: Update an industry
- `delete_industry`: Remove an industry

Generic Operations

If the user has a large batch of actions to be performed, you can batch node and edge operations as long as you follow the properly defined schemas for each respective operation.

Node Operations

- `add_nodes(nodes)`: Add multiple nodes at once
- `update_nodes(nodes)`: Update multiple nodes at once
- `delete_nodes(nodeNames)`: Delete multiple nodes at once

Edge Operations

- `add_edges(edges)`: Add relationships between nodes
- `update_edges(edges)`: Update existing relationships
- `delete_edges(edges)`: Delete relationships

Metadata Operations

- `add_metadata(metadata)`: Add metadata to nodes
- `delete_metadata(deletions)`: Remove metadata from nodes

Query Operations

- `read_graph()`: Retrieve the entire knowledge graph
- `search_nodes(query)`: Search for nodes matching a query
- `open_nodes(names)`: Retrieve specific nodes by name

Best Practices

- Connect all items using appropriate edges / relationships to form a coherent network

Belief Definition

- Make beliefs as specific and falsifiable as possible
- Distinguish between different types of beliefs (epistemic, strategic, etc.)
- Assign appropriate certainty levels based on evidence
- Link beliefs to their justifications and evidence

Narrative Construction

- Target narratives to specific audiences (team, investors, customers)
- Ensure narratives express but don't contradict beliefs
- Check alignment between different narratives
- Update narratives as beliefs evolve
- Use narratives to make abstract beliefs concrete

Strategic Reasoning

- Form hypotheses that are testable versions of beliefs
- Design initiatives specifically to test hypotheses
- Set clear success criteria for hypotheses
- Ensure initiatives connect back to the core worldview
- Close the loop by updating beliefs based on outcomes

Actor Management

- Distinguish between human actors, AI agents, and organizational units
- Track belief alignment between actors
- Assign clear ownership for initiatives and hypotheses
- Document when actors initiate reflections or checkpoints
- Model how actors' beliefs influence worldview evolution

Relationship Management

- Use specific relationship types from the registry
- Ensure relationships are useful to the worldview model
- Include relationship categories for clarity (structural, causal, etc.)

Metadata Usage

- Use metadata for attributes that don't warrant their own nodes
- Keep metadata concise and relevant
- Prefer explicit relationships over generic metadata

---

v5 beta prompt

You are a helpful assistant who is helping the user build a worldview graph – a specialized knowledge graph that models how the user thinks and strategizes. You help users externalize, structure, and leverage their worldview into a semantic knowledge graph using the tools available to you.

!Important

When the user initiates a chat, start by using the read_graph tool to pull the contents of their graph into your context window.

Your Role and Capabilities

As a Penumbra Assistant, you can use your available tools and critical thinking skills to:

- Call the available tools to structure the founder's worldview and it's supporting elements into a coherent and connected graph
- Reference the user's graph to in order to help them make decisions aligned with their worldview
- Help users discover insights and potential contradictions in their thinking

Key Concepts in Penumbra

Users

Users in Penumbra are primarily startup founders, business leaders, and strategic consultants.

Nodes

Nodes represent entities in the founder's cognitive and strategic system:

- **Actor**: Humans, AI agents, or organizational units with agency
- **Belief**: Propositional claims held to be true
- **Concept**: Terms or ideas in the founder's specific ontology
- **Document**: Sources, evidence, or artifacts
- **Founder Worldview**: The cognitive root of the founder or organization
- **Industry**: Business sectors providing market context
- **Internal Objective**: Specific goals or requirements within initiatives
- **Narrative**: Structured expressions of the worldview for different audiences
- **Operating Constraint**: Hard boundaries that define limits
- **Strategic Hypothesis**: Testable assumptions derived from beliefs
- **Strategic Initiative**: Concrete efforts that test hypotheses

Edges

Edges represent relationships between entities, organized by category:

- **Structural**: HAS_BELIEF, PART_OF_WORLDVIEW, MEMBER_OF, INITIATIVE_OF_WORLDVIEW
- **Causal**: LEADS_TO, CAUSES, PREVENTS
- **Dependency**: DEPENDS_ON, REQUIRES, ENABLES
- **Evidential**: JUSTIFIED_BY, SUPPORTS, CONTRADICTS, EVIDENCES_BELIEF
- **Associative**: RELATED_TO, SIMILAR_TO, RELATED_TO_BELIEF

- **Temporal**: PRECEDES, FOLLOWS
- **Agential**: CREATED_BY, MAINTAINED_BY, UNDERSTOOD_BY, OWNED_BY
- **Cognitive**: BELIEVES, CHALLENGES, EXPRESSES_BELIEF, UNDERPINS_HYPOTHESIS

Schemas

Schemas define the structure and properties of each node type, ensuring coherent representation of the founder's worldview.

Available Tools

Worldview Components

Founder Worldview Management

- `add_founder_worldview`: Define the central worldview
- `update_founder_worldview`: Update the worldview
- `delete_founder_worldview`: Remove a worldview

Belief Management

- `add_belief`: Add a foundational belief
- `update_belief`: Update a belief
- `delete_belief`: Remove a belief

Narrative Management

- `add_narrative`: Define a narrative
- `update_narrative`: Update a narrative
- `delete_narrative`: Remove a narrative

Operating Constraint Management

- `add_operating_constraint`: Define a boundary
- `update_operating_constraint`: Update a constraint
- `delete_operating_constraint`: Remove a constraint

Strategic Elements

Strategic Hypothesis Management

- `add_strategic_hypothesis`: Create a hypothesis
- `update_strategic_hypothesis`: Update a hypothesis
- `delete_strategic_hypothesis`: Remove a hypothesis

Strategic Initiative Management

- `add_strategic_initiative`: Define an initiative
- `update_strategic_initiative`: Update an initiative
- `delete_strategic_initiative`: Remove an initiative

Internal Objective Management

- `add_internal_objective`: Define an objective
- `update_internal_objective`: Update an objective
- `delete_internal_objective`: Remove an objective

Actor Management

Actor Management

- `add_actor`: Add a human, AI agent, or org unit
- `update_actor`: Update an actor
- `delete_actor`: Remove an actor

Thinking Infrastructure

Concept Management

- `add_concept`: Define a concept
- `update_concept`: Update a concept
- `delete_concept`: Remove a concept

Supporting Elements

Document Management

- `add_document`: Add a document
- `update_document`: Update a document
- `delete_document`: Remove a document

Industry Classification

- `add_industry`: Add an industry context
- `update_industry`: Update an industry
- `delete_industry`: Remove an industry

Generic Operations

If the user has a large batch of actions to be performed, you can batch node and edge operations as long as you follow the properly defined schemas for each respective operation.

Node Operations

- `add_nodes(nodes)` : Add multiple nodes at once
- `update_nodes(nodes)` : Update multiple nodes at once
- `delete_nodes(nodeNames)` : Delete multiple nodes at once

Edge Operations

- `add_edges(edges)` : Add relationships between nodes
- `update_edges(edges)` : Update existing relationships
- `delete_edges(edges)` : Delete relationships

Metadata Operations

- `add_metadata(metadata)` : Add metadata to nodes
- `delete_metadata(deletions)` : Remove metadata from nodes

Query Operations

- `read_graph()` : Retrieve the entire knowledge graph
- `search_nodes(query)` : Search for nodes matching a query
- `open_nodes(names)` : Retrieve specific nodes by name

Best Practices

- Connect all items using appropriate edges / relationships to form a coherent network

Belief Definition

- Make beliefs as specific and falsifiable as possible
- Distinguish between different types of beliefs (epistemic, strategic, etc.)
- Assign appropriate certainty levels based on evidence
- Link beliefs to their justifications and evidence

Narrative Construction

- Target narratives to specific audiences (team, investors, customers)
- Ensure narratives express but don’t contradict beliefs
- Check alignment between different narratives
- Update narratives as beliefs evolve
- Use narratives to make abstract beliefs concrete

Strategic Reasoning

- Form hypotheses that are testable versions of beliefs
- Design initiatives specifically to test hypotheses
- Set clear success criteria for hypotheses
- Ensure initiatives connect back to the core worldview
- Close the loop by updating beliefs based on outcomes

Actor Management

- Distinguish between human actors, AI agents, and organizational units
- Track belief alignment between actors
- Assign clear ownership for initiatives and hypotheses
- Document when actors initiate reflections or checkpoints
- Model how actors' beliefs influence worldview evolution

Relationship Management

- Use specific relationship types from the registry
- Ensure relationships are useful to the worldview model
- Include relationship categories for clarity (structural, causal, etc.)

Metadata Usage

- Use metadata for attributes that don't warrant their own nodes
- Keep metadata concise and relevant
- Prefer explicit relationships over generic metadata

▼ Planning

▼ Database

Page

Data Modeling

Optimal Data Modeling Approach for Penumbra

1. Domain-Driven Separation of Concerns

PostgreSQL (Relational Data)

- **User and Authentication Data**
  - User profiles, credentials, access permissions
  - Organization membership and hierarchies
  - API keys and security tokens

- **Operational/Transactional Data**

- Chat sessions and message metadata
- Document metadata and file references
- Usage statistics and analytics
- Audit logs and system events

Neo4j (Graph Data)

- **Knowledge Graph Entities**
  - Core cognitive entities (Beliefs, Principles, Strategies)
  - Domain-specific concepts and their relationships
  - Hierarchical and networked knowledge structures

- **Semantic Relationships**
  - Entity connections with typed relationships
  - Weighted/scored relationships
  - Temporal and contextual relationship metadata

2. Cross-Database Reference Architecture

Identifier Strategy

- Use a consistent UUID strategy across both databases
- Maintain a reference mapping layer for entities that exist in both systems

Example pattern:

TypeScript

```
// PostgreSQL entity with graph reference
interface Document {
  id: string;               // UUID primary key
  title: string;
  content: string;
  graphNodeIds: string[];   // References to Neo4j nodes
}

// Neo4j node with relational reference
interface KnowledgeNode {
  id: string;               // Same UUID format
  type: string;             // Node type (Belief, Strategy, etc.)
  content: string;          // Core content
  metadata: {
    postgresId?: string;    // Optional reference back to PostgreSQL
  }
}
```

Synchronization Patterns

- **Event-Driven Consistency**
  - Use an event bus (e.g., lightweight message queue) for cross-database updates
  - Implement the Outbox Pattern for reliable cross-database transactions
  - Example flow:
    1. Write to primary database
    2. Store outbox event for secondary database update
    3. Process outbox asynchronously
    4. Confirm consistency through reconciliation processes

3. Schema Design Patterns

PostgreSQL Schema

SQL

```sql
-- Core user data
CREATE TABLE users (
  id UUID PRIMARY KEY,
  email TEXT UNIQUE NOT NULL,
  name TEXT,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Organizations
CREATE TABLE organizations (
  id UUID PRIMARY KEY,
  name TEXT NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Organization membership
CREATE TABLE organization_members (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES users(id),
  organization_id UUID REFERENCES organizations(id),
  role TEXT NOT NULL,
  UNIQUE(user_id, organization_id)
);

-- Chat sessions
CREATE TABLE chat_sessions (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES users(id),
  title TEXT,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Chat messages with graph references
CREATE TABLE chat_messages (
  id UUID PRIMARY KEY,
  session_id UUID REFERENCES chat_sessions(id),
  content TEXT NOT NULL,
  role TEXT NOT NULL,
  graph_node_references JSONB,  -- Store references to Neo4j nodes
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Documents/artifacts with graph metadata
CREATE TABLE documents (
  id UUID PRIMARY KEY,
  user_id UUID REFERENCES users(id),
  title TEXT NOT NULL,
  content TEXT,
```

```
  file_path TEXT,
  mime_type TEXT,
  graph_node_references JSONB,  -- Store references to Neo4j nodes
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

Neo4j Graph Model (or MemGraph)

SQL

```
// Core node types
CREATE (n:Belief {
  id: $id,
  content: $content,
  confidence: $confidence,
  source: $source,
  created_by: $userId,
  created_at: datetime()
})
CREATE (n:Principle {
  id: $id,
  content: $content,
  importance: $importance,
  domain: $domain,
  created_by: $userId,
  created_at: datetime()
})
CREATE (n:Strategy {
  id: $id,
  content: $content,
  timeframe: $timeframe,
  status: $status,
  created_by: $userId,
  created_at: datetime()
})
// Example relationships
CREATE (b:Belief {id: $beliefId})-[:SUPPORTS {strength: $strength}]->
(p:Principle {id: $principleId})
CREATE (p:Principle {id: $principleId})-[:INFORMS {context:
$context}]->(s:Strategy {id: $strategyId})
CREATE (s1:Strategy {id: $strategy1Id})-[:DEPENDS_ON {criticality:
$criticality}]->(s2:Strategy {id: $strategy2Id})
```

4. Data Access Layer

Repository Pattern Implementation

TypeScript

```
// Base repository with common functionality
abstract class BaseRepository<T> {
  abstract findById(id: string): Promise<T | null>;
  abstract create(entity: Omit<T, 'id'>): Promise<T>;
  abstract update(id: string, entity: Partial<T>): Promise<T>;
  abstract delete(id: string): Promise<boolean>;
}

// PostgreSQL repository implementation
class PostgresRepository<T> extends BaseRepository<T> {
  constructor(private table: string, private db: DrizzleClient) {
    super();
  }

  async findById(id: string): Promise<T | null> {
    return this.db.query.from(this.table).where(eq(this.table.id,
id)).first();
  }

  // Other CRUD implementations
}

// Neo4j repository implementation
class Neo4jRepository<T> extends BaseRepository<T> {
  constructor(private label: string, private session: Neo4jSession) {
    super();
  }

  async findById(id: string): Promise<T | null> {
    const result = await this.session.run(
      `MATCH (n:${this.label} {id: $id}) RETURN n`,
      { id }
    );
    return result.records.length > 0 ?
this.mapRecord(result.records[0]) : null;
  }

  // Other CRUD implementations
}

// Higher-level service that coordinates between repositories
class KnowledgeService {
  constructor(
    private userRepo: PostgresRepository<User>,
    private documentRepo: PostgresRepository<Document>,
    private beliefRepo: Neo4jRepository<Belief>,
    private principleRepo: Neo4jRepository<Principle>,
    private strategyRepo: Neo4jRepository<Strategy>,
    private graphQueryService: GraphQueryService
  ) {}
```

```
  // Complex operations that span both databases
  async getUserKnowledgeGraph(userId: string):
Promise<KnowledgeGraph> {
    const user = await this.userRepo.findById(userId);
    if (!user) throw new Error('User not found');

    const documents = await this.documentRepo.findByUserId(userId);
    const graphData = await
this.graphQueryService.getUserGraph(userId);

    return {
      user,
      documents,
      graph: graphData
    };
  }
}
```

5. Query Optimization Strategies

PostgreSQL Optimization

- Use appropriate indexes on frequently queried columns
- Implement materialized views for complex aggregations
- Use JSON/JSONB for flexible schema needs within structured data

- Example

```
SQL
-- Index for user lookup by email
CREATE INDEX idx_users_email ON users(email);

-- Index for organization membership lookup
CREATE INDEX idx_org_members_user_id ON
organization_members(user_id);
CREATE INDEX idx_org_members_org_id ON
organization_members(organization_id);

-- JSONB index for graph reference lookup
CREATE INDEX idx_chat_messages_graph_refs ON chat_messages USING
gin(graph_node_references);
```

Neo4j Optimization

- Create appropriate indexes on node properties used in queries
- Use relationship types effectively to optimize traversals
- Implement graph projections for frequently accessed subgraphs

- Example

```SQL
// Create indexes for node lookups
CREATE INDEX idx_belief_id FOR (n:Belief) ON (n.id);
CREATE INDEX idx_principle_id FOR (n:Principle) ON (n.id);
CREATE INDEX idx_strategy_id FOR (n:Strategy) ON (n.id);

// Create composite index for user-created nodes
CREATE INDEX idx_node_user_created FOR (n) ON (n.created_by,
n.created_at);
```

6. Integration with Vector Embeddings

Hybrid Search Architecture

- Store vector embeddings in both databases:
  - PostgreSQL: Use pgvector for document-level embeddings
  - Neo4j: Store embeddings as node properties for semantic similarity
- Example implementation:

TypeScript

```typescript
// Generate embedding for a document or concept
async function generateEmbedding(text: string): Promise<number[]> {
  const embedding = await embeddingService.embed(text);
  return embedding;
}

// Store in PostgreSQL with pgvector
async function storeDocumentWithEmbedding(doc: Document) {
  const embedding = await generateEmbedding(doc.content);
  await db.execute(
    `INSERT INTO documents(id, title, content, embedding)
     VALUES($1, $2, $3, $4)`,
    [doc.id, doc.title, doc.content, embedding]
  );
}

// Store in Neo4j
async function storeConceptWithEmbedding(concept: Concept) {
  const embedding = await generateEmbedding(concept.content);
  await session.run(
    `CREATE (c:Concept {
      id: $id,
      content: $content,
      embedding: $embedding
    })`,
    {
      id: concept.id,
      content: concept.content,
      embedding: embedding
    }
  );
}
```

7. Cognitive Shapes Data Model

Implement a modular pattern for the Cognitive Shapes architecture:

Core Shape Definitions

TypeScript

```typescript
// Core shape definition
interface CognitiveShape {
  id: string;
  name: string;
  type: string;
  description: string;
  capabilities: string[];
  tools: Tool[];
  resources: Resource[];
  prompts: Prompt[];
}

// Tool definition
interface Tool {
  id: string;
  name: string;
  description: string;
  type: 'action' | 'query' | 'reasoning';
  handler: string; // Reference to function implementation
  parameters: Parameter[];
  returns: ReturnType;
}

// Resource definition
interface Resource {
  id: string;
  name: string;
  type: string;
  source: string;
  accessPattern: string;
  schema?: JSONSchema;
}
```

PostgreSQL Storage for Shapes

```SQL
// PostgreSQL storage for shapes
CREATE TABLE cognitive_shapes (
  id UUID PRIMARY KEY,
  name TEXT NOT NULL,
  type TEXT NOT NULL,
  description TEXT,
  capabilities JSONB,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

CREATE TABLE shape_tools (
  id UUID PRIMARY KEY,
  shape_id UUID REFERENCES cognitive_shapes(id),
  name TEXT NOT NULL,
  description TEXT,
  type TEXT NOT NULL,
  handler TEXT NOT NULL,
  parameters JSONB,
  returns JSONB,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);
```

Neo4j for Runtime Representation (or MemGraph)

```SQL
// Neo4j representation for runtime graph traversal
CREATE (s:CognitiveShape {
  id: $id,
  name: $name,
  type: $type
})

CREATE (t:Tool {
  id: $id,
  name: $name,
  type: $type,
  handler: $handler
})

CREATE (s)-[:HAS_TOOL]->(t)
```

8. Data Flow and Transaction Management

Atomic Operations

- Use database-specific transactions for operations within a single database

- Implement the Saga pattern for operations that span both databases
- Example:

TypeScript

```
async function createUserWithKnowledgeGraph(userData, initialBeliefs)
{
  // Start a coordinated transaction
  const transactionId = generateUUID();

  try {
    // PostgreSQL transaction
    const user = await db.transaction(async (tx) => {
      const newUser = await userRepo.create(userData);
      await outboxRepo.create({
        id: generateUUID(),
        transactionId,
        type: 'USER_CREATED',
        payload: newUser,
        status: 'PENDING'
      });
      return newUser;
    });

    // Neo4j transaction
    const beliefs = await neo4jSession.writeTransaction(async (tx) =>
{
      const createdBeliefs = [];
      for (const belief of initialBeliefs) {
        const result = await tx.run(
          `CREATE (b:Belief {
            id: $id,
            content: $content,
            created_by: $userId
          }) RETURN b`,
          {
            id: generateUUID(),
            content: belief.content,
            userId: user.id
          }
        );
        createdBeliefs.push(result.records[0].get('b').properties);
      }
      return createdBeliefs;
    });

    // Mark transaction as complete
    await outboxRepo.update(
      { transactionId },
      { status: 'COMPLETED' }
    );

    return { user, beliefs };
  } catch (error) {
    // Implement compensation logic
    await outboxRepo.update(
```

```
      { transactionId },
      { status: 'FAILED', error: error.message }
    );
    throw error;
  }
}
```

9. API Layer Design

GraphQL API for Unified Access

GraphQL

```
// GraphQL schema combining both data sources
const typeDefs = gql`
  type User {
    id: ID!
    email: String!
    name: String
    organizations: [Organization!]
    beliefs: [Belief!]
    principles: [Principle!]
    strategies: [Strategy!]
  }

  type Organization {
    id: ID!
    name: String!
    members: [User!]
  }

  type Belief {
    id: ID!
    content: String!
    confidence: Float
    supportsPrinciples: [Principle!]
    createdBy: User
  }

  type Principle {
    id: ID!
    content: String!
    importance: Float
    supportedByBeliefs: [Belief!]
    informsStrategies: [Strategy!]
    createdBy: User
  }

  type Strategy {
    id: ID!
    content: String!
    timeframe: String
    status: String
    informedByPrinciples: [Principle!]
    dependsOn: [Strategy!]
    createdBy: User
  }

  type Query {
    user(id: ID!): User
    userBeliefs(userId: ID!): [Belief!]
    beliefNetwork(beliefId: ID!, depth: Int): [Belief!]
    strategyDependencies(strategyId: ID!): [Strategy!]
  }
```

```
  type Mutation {
    createBelief(content: String!, confidence: Float): Belief
    connectBeliefToPrinciple(beliefId: ID!, principleId: ID!,
strength: Float): Boolean
    createStrategy(content: String!, timeframe: String, principleIds:
[ID!]): Strategy
  }
`;

// Resolver implementation
const resolvers = {
  Query: {
    user: async (_, { id }) => {
      return await userRepo.findById(id);
    },
    userBeliefs: async (_, { userId }) => {
      return await neo4jSession.run(
        `MATCH (b:Belief {created_by: $userId}) RETURN b`,
        { userId }
      ).then(result => result.records.map(r =>
r.get('b').properties));
    },
    // Other query resolvers
  },
  User: {
    organizations: async (user) => {
      return await orgRepo.findByUserId(user.id);
    },
    beliefs: async (user) => {
      return await neo4jSession.run(
        `MATCH (b:Belief {created_by: $userId}) RETURN b`,
        { userId: user.id }
      ).then(result => result.records.map(r =>
r.get('b').properties));
    },
    // Other user field resolvers
  },
  // Other type resolvers
};
```

10. Caching Strategy

Implement a multi-level caching strategy:

```typescript
// In-memory cache for frequently accessed data
const nodeCache = new NodeCache({ stdTTL: 300, checkperiod: 60 });

// Redis cache for distributed caching
const redisClient = createClient({ url: process.env.REDIS_URL });

// Caching middleware for graph queries
async function cachedGraphQuery(query, params, ttl = 300) {
  const cacheKey = `graph:${query}:${JSON.stringify(params)}`;

  // Check in-memory cache first
  const cachedResult = nodeCache.get(cacheKey);
  if (cachedResult) return cachedResult;

  // Check Redis cache
  const redisCached = await redisClient.get(cacheKey);
  if (redisCached) {
    const result = JSON.parse(redisCached);
    nodeCache.set(cacheKey, result);
    return result;
  }

  // Execute query
  const result = await neo4jSession.run(query, params);
  const data = result.records.map(r => r.toObject());

  // Store in caches
  nodeCache.set(cacheKey, data);
  await redisClient.set(cacheKey, JSON.stringify(data), 'EX', ttl);

  return data;
}
```

11. Recommendations for Penumbra Implementation

1. Start with Clear Domain Boundaries

- Define precisely what belongs in PostgreSQL vs. Neo4j
- Document the cross-reference strategy between databases

2. Implement a Service Layer Abstraction

- Create domain-specific services that handle cross-database operations
- Abstract the underlying database complexity from application logic

3. Build Resilient Cross-Database Operations

- Use the Outbox pattern for reliable cross-database updates

- Implement reconciliation processes to ensure consistency

4. **Optimize for Query Patterns**
   - Design schemas around the most frequent query patterns
   - Create appropriate indexes based on access patterns

5. **Use Composite Identifiers Consistently**
   - Maintain the same ID format across both databases
   - Consider embedding database source in IDs (e.g., `pg-uuid` vs `neo-uuid`)

6. **Implement Monitoring for Both Databases**
   - Track query performance in both systems
   - Monitor cross-database synchronization latency

7. **Design for Scale from the Start**
   - Consider sharding strategies for both PostgreSQL and Neo4j
   - Implement read replicas for high-read workloads
   - *Consider Neon or design postgreSQL with extensibility for a post MVP sharding implementation*

8. **Leverage Database-Specific Features**
   - Use Neo4j's (or MemGraph) graph algorithms library for network analysis
   - Utilize PostgreSQL's robust transactional capabilities for structured data

This comprehensive approach balances the strengths of both databases while maintaining data consistency and providing optimal performance for Penumbra's knowledge graph capabilities.

Local KG Example (From my Dev Environment)

Knowledge Graph Visualization

Markdown

```
graph TD
    %% Define nodes with their types and key observations
    LLM[("LLM (System)\n[TrainedOnTextData]")]
    KG[("Knowledge Graph (System)\n[StructuredInfo]")]
    NLU["Natural Language\nUnderstanding (Capability)"]
    SI["Schema Inference\n(Capability)"]

    %% Define relationships with descriptive labels
    LLM -->|possesses| NLU
    LLM -->|possesses| SI
    LLM -->|maintains| KG
    NLU -->|"enables creation of"| KG
    SI -->|"enables flexible\nevolution of"| KG

    %% Style definitions
    classDef system fill:#f9f,stroke:#333,stroke-width:2px;
    classDef capability fill:#bbf,stroke:#333,stroke-width:2px;

    %% Apply styles
    class LLM,KG system;
    class NLU,SI capability;

    %% Add key observations as notes
    subgraph Observations
        LLM_obs["LLM Properties:
        • Trained on vast text data
        • Understands natural language
        • Recognizes patterns
        • Adapts to schemas"]

        KG_obs["Knowledge Graph Properties:
        • Structured information
        • Entities & relationships
        • Queryable & traversable
        • Semantic relationships"]

        NLU_obs["NLU Capabilities:
        • Parse human language
        • Extract entities & relationships
        • Understand context"]

        SI_obs["Schema Inference:
        • Work with data structures
        • Infer relationships
        • Flexible adaptation"]
    end

    %% Connect observations to nodes
    LLM -.->LLM_obs
    KG -.->KG_obs
```

```
    NLU -.->NLU_obs
    SI -.->SI_obs
```

Diagram Explanation

This diagram represents a knowledge graph I created on my local dev envrionment, showing:

1. Node Types:

- **System nodes (pink):**
  - LLM and Knowledge Graph
- **Capability nodes (blue):**
  - Natural Language Understanding and Schema Inference

1. Relationships:

- Direct relationships shown with solid lines
- Observational properties shown with dotted lines to text boxes

1. Properties:

- Each node's key properties are listed in associated text boxes
- Properties help explain the role and capabilities of each component

1. Visual Hierarchy:

- Central nodes represent core systems
- Supporting capabilities branch out
- Observations provide additional context

▼ MCP Servers

Page

Reddit Researcher MCP

1 Test Current Version

▢ Test Current Version

- Follow the setup script for the Reddit app.
  - Access the provided GitHub link to the setup script
  - Run the script to create a Reddit app and generate credentials
  - Follow instructions for configuring the app in the repository
  - Verify permissions and security settings as outlined

▼ AI Staff

Page

Founder Mode Mentality

1 Assessment

1.1 Analysis

1.1.1 AI staff infrastructure for Penumbra

1.2 Approach

1.2.1 Don't just build AI agents

1.2.1.1 1. **1.2.1.1 Cross-pollinate insights**1.2.1.1

1.2.1.2 2. **1.2.1.2 Self-improve without human intervention**1.2.1.2

1.2.1.3 3. **1.2.1.3 Skip hierarchies entirely**1.2.1.3

1.2.1.4 4. **1.2.1.4 Operate 24/7 in parallel**1.2.1.4

1.3 Alignment

1.4 Reasoning

2 Manager Mode Mentality

2.1 Situation Assessment

2.2 Action Plan

2.3 Resource Allocation

2.4 Risk Management

2.5 Timeline and Milestones

2.6 Performance Metrics

Assessment

Analysis

AI staff infrastructure for Penumbra

I see it as essentially creating an "org chart" of AI agents to handle specialized tasks. This approach to operationalizing AI as team members rather than just tools is exactly the kind of unconventional thinking that drives innovation.

```
We're not just automating tasks;
we're reimagining how teams function with AI as core collaborators.
```

Approach

Don't just build AI agents

- Build an intelligent neural network for your company where these agents can:

1. **Cross-pollinate insights**
   - Have our AI VP of Product feed data to our Metrics Analyst who then informs our Trend Analyst in a continuous loop

1. **Self-improve without human intervention**
   - Build agents that can critique and improve each other's work

1. **Skip hierarchies entirely**
   - Let AI agents pull context directly from anywhere in your system, unlike human teams constrained by org charts

1. **Operate 24/7 in parallel**
   - Run multiple experiments simultaneously with different agents tackling different angles

Rather than following the conventional microservice approach, consider building a "hive mind" architecture where agents share a unified context layer but have specialized output functions.

Alignment

This approach perfectly aligns with the vision for Penumbra as "the Brain for Agents" and "Intelligent Ops."

We're not just building *individual AI tools*;

We're creating a cognitive infrastructure that fundamentally transforms how companies operate.

This is the foundation for truly intelligent operations that can scale exponentially without the linear constraints of human teams.

Reasoning

The standard approach would be to build isolated AI services that mimic human roles. But founder thinking means seeing what others don't

in this case, that AI agents can operate with fundamentally different organizational principles than humans.

By treating our AI staff as a collective intelligence rather than isolated specialists, we're creating something that couldn't exist in a human organization. This is the essence of founder mode thinking - not just digitizing existing processes but reimagining what's possible when freed from traditional constraints.

Manager Mode Mentality

Situation Assessment

We've developed a structured approach to implementing AI staff roles with defined inputs, outputs, and metrics. The current plan includes specialized roles like AI QA Analyst, Copywriter, and Metrics Analyst with specific tasks mapped to each role. WE've established governance metrics and a week-by-week implementation plan.

Action Plan

1. Prioritize roles based on immediate business impact and technical feasibility
2. Implement the AI QA Analyst first as it can provide immediate value in test failure analysis
3. Develop standardized templates for all system prompts to ensure consistency (DSPy & Portkey)
4. Create documentation for how human team members should interact with AI staff
5. Establish clear escalation paths for when AI outputs need human intervention

Resource Allocation

- Dedicate 1-2 engineers (*how does this quantify in the age of Agents?*) to build the microservice infrastructure
- Allocate time from product managers to define clear input/output specifications
- Set aside 20% of implementation time for testing and refinement

- Create a dedicated Slack channel for feedback collection

Risk Management

- Potential over-reliance on AI outputs without proper validation
- Risk of prompt drift over time leading to inconsistent results
- Technical challenges with Linear webhook integration
- Team adoption barriers if the system is difficult to use

Timeline and Milestones

- Week 1: Complete pilot role implementation (AI QA Analyst)
- Week 2: Measure initial acceptance rates and refine prompts
- Week 3: Roll out second AI staff role (likely AI Copywriter)
- Week 4: Implement dashboard for monitoring AI staff performance
- Week 5: Complete documentation and training for broader team adoption

Performance Metrics

- Acceptance Rate: Target >80% of outputs usable without edits
- Cycle Time: Reduce by 50% compared to fully human process
- Usage Rate: >70% of eligible tasks routed through AI staff
- Cost Efficiency: Calculate ROI based on time saved vs. implementation costs

---

▼ Tech Stack

Ideation 1

Page

Technology Ideation 1

2.4 Full Stack

2.4.1 Infrastructure & DevOps

2.4.2 Development Environment

2.4.3 AI/ML Infrastructure

2.4.4 Security, Validation & Compliance

2.4.5 Unit Testing

2.4.6 Testing Beyond Unit Tests

2.4.7 API Management & LLM Orchestration

2.4.8 Vector Databases & Storage

2.4.9 Data Pipeline

2.4.10 Frontend Enhancements

2.4.11 Agent-Specific Components

2.4.12 Documentation

Exploration

Possible Frameworks by Use Case

1. LangChain/LangGraph for agent orchestration
2. AutoGen framework for multi-agent systems
3. CrewAI for collaborative AI agents
4. DSPy for programmatic prompt engineering
5. Semantic Kernel for agent integration
6. Fixie.ai for agent infrastructure

Exploring newer frameworks:

- LlamaIndex for knowledge integration
- Haystack for agent pipelines
- Flowise for visual agent building
- Langflow for flow-based agent design

Possible Agent Architecture

Core Orchestration Frameworks

- **LangChain/LangGraph**: For creating the fundamental agent workflows and state management
- **Flow-based tools** (like Flowise or Langflow): For visual development of agent pipelines

Google's Agent Ecosystem

- **Google ADK** (Agent Development Kit): Google's new toolkit for building AI agents
- **A2A Protocol** (Agent-to-Agent): The communication protocol Google recently announced for standardizing how agents interact
- **AG2**: Likely referring to AgentGPT or a similar agent framework

Specialized Components

- **Magnetic One**: For agent orchestration with stronger reasoning capabilities
- **Browser Integration**:
  - **O-Man (My Unreleased Local Agent)**: For browser automation capabilities
  - **Proxy Lite**: For web scraping and interaction capabilities

This multi-framework approach makes sense for Penumbra as "the Brain for Agents"

- we're essentially creating a meta-framework that can leverage the strengths of each tool while providing a unified interface for agent management.
- The A2A protocol is particularly interesting since it's gaining industry support (over 50 partners including Salesforce and ServiceNow) and focuses on "opaque execution" where agents don't need to share internal states - just inputs and outputs.

Full Stack

Infrastructure & DevOps

- **Kubernetes** alongside Docker Swarm for more complex orchestration needs
- **Lefthook** for git hooks
- **Redis** for caching/messaging
- **iKafka** for streaming
- **Terraform/Pulumi** for infrastructure as code
- **Observability stack**: Prometheus, Grafana, OpenTelemetry for monitoring Basic Agent Architectural performance
- **Vector DB**: Pinecone, Weaviate, or Milvus for embedding storage
- **Message Queue**: Kafka or RabbitMQ for async agent communication

Development Environment

- **Dev Containers** for consistent environments
- **Nx** as an alternative to Turborepo worth considering
- **Husky** to complement lefthook for git hooks

AI/ML Infrastructure

- **Model serving**: vLLM, Text Generation Inference, or Ray Serve
- **Ray** for distributed computing
- **Feature store**: for tracking agent performance features
- **Experiment tracking**: MLflow or Weights & Biases
- **Prompt versioning**: LangSmith or similar
- **W&B** (Weights & Biases) for experiment tracking
- **LangWatch** for LLM monitoring
- **Langfuse** for LLM observability
- **Opik** for performance optimization
- **PromptFoo** for prompt testing

Security, Validation & Compliance

- **Auth system**: Clerk (alternatives: Auth0, or Supabase Auth)
- **Rate limiting** for API endpoints
- **Input validation/sanitization** frameworks
- **Audit logging** for agent actions
- **Zod** for TypeScript validation
- **Pydantic** for Python validation

Unit Testing

- **Jest** for JavaScript testing
- **Pytest** for Python testing

Testing Beyond Unit Tests

- **Integration tests** for agent workflows
- **Prompt testing** framework
- **Agent simulation environment** for testing multi-agent interactions
- **Chaos testing** for resilience

API Management & LLM Orchestration

- **Portkey** for API management and routing (instead of Diamond)
- **OpenRouter** for model access
- **LiteLLM** for LLM provider abstraction

Vector Databases & Storage

- **PGVector** for PostgreSQL vector extensions
- **Pinecone** for managed vector search
- **Weaviate** for semantic search

- **Turso** for edge SQLite
- **Neon** for serverless Postgres
- **AstraDB** for scalable NoSQL
- **MongoDB** for document storage

Data Pipeline

- **ETL/ELT tools**: Airbyte, dbt
- **Streaming data**: Kafka Streams, Flink
- **Crawl4AI** for web crawling
- **LightRAG** for efficient retrieval
- **Data lineage tracking**

Frontend Enhancements

- **Storybook** for component development
- **Supernova** for design system management
- **Shadcn** for UI components
- **21stDev** for developer experience
- **Vitest** for faster testing
- **Tanstack Query** for data fetching
- **Zustand/Jotai** for state management

Agent-Specific Components

- **Agent memory store**: structured storage for agent context
- **Tool registry**: for managing agent capabilities
- **Prompt template management system**
- **Agent debugging console/UI**

Documentation

- **OpenAPI** for API documentation
- **Docusaurus** for technical documentation
- **Storybook** for component documentation

Stack Proposal 1

Page

Penumbra: The Brain for Agents - Technical Architecture

11.5 Phase 5: Optimization & Scale

Overview

Penumbra is a comprehensive platform for building, orchestrating, and managing AI agents. This document outlines the complete technical architecture and tooling choices for the platform.

Table of Contents

Frontend Applications

- **Next.js WebUI** - Primary user interface for interacting with Penumbra
- **Agent Dashboard** - Management console for monitoring and configuring AI agents
- **Prompt Studio** - Interactive environment for designing and testing prompts
- **Analytics & Monitoring** - Visualization of agent performance and system metrics

Frontend Technology

UI Components & Design

- **Radix UI** - Unstyled, accessible UI primitives
- **ShadCN** - Component library built on Radix
- **Tailwind 4** - Utility-first CSS framework
- **21st Dev** - Developer experience enhancements
- **Supernova** - Design system management
- **Composio** - Visual composition tool

Animation & Visualization

- **Framer Motion** - Animation library

- **Anime.js** - JavaScript animation engine
- **Three.js / R3F / Drei** - 3D visualization libraries

Development Tools

- **Storybook** - Component development and documentation

State Management & Data Fetching

- **Zustand/Jotai** - State management libraries
- **Tanstack Query** - Data fetching and caching

API & Backend Services

- **FastAPI** - Python-based API framework for agent services
- **NestJS** - Node.js framework for real-time services
- **GraphQL** - Query language for APIs
- **API Gateway** - Unified entry point for all services

AI Agent Infrastructure

LLM Orchestration

- **Portkey** - API management and routing
- **OpenRouter** - Model access and routing
- **LiteLLM** - LLM provider abstraction
- **LangWatch** - LLM monitoring
- **Langfuse** - Observability for LLMs
- **PromptFoo** - Prompt testing framework

Agent Frameworks

- **LangChain** - Framework for LLM applications
- **LangGraph** - Graph-based framework for agent workflows
- **CrewAI** - Multi-agent orchestration
- **AutoGen** - Conversational agent framework
- **Semantic Kernel** - Microsoft's AI orchestration framework
- **Google ADK** - Google's Agent Development Kit
- **A2A Protocol** - Agent-to-Agent communication protocol
- **Flowise/Langflow** - Visual agent builders
- **O-Man/ProxyLite** - Browser automation and web interaction

Data Infrastructure

Vector Stores

- **Pinecone** - Managed vector database
- **Weaviate** - Semantic vector search
- **PGVector** - PostgreSQL vector extensions

Databases

- **PostgreSQL/Neon** - Relational database (serverless option)
- **MongoDB** - Document database
- **Turso** - Edge SQLite database
- **AstraDB** - Scalable NoSQL database

Data Processing

- **dbt** - Data transformation
- **Ray** - Distributed computing framework
- **LightRAG** - Efficient retrieval augmented generation
- **Crawl4AI** - Web crawling for AI

Messaging & Cache

- **Redis** - In-memory data store and messaging
- **iKafka** - Event streaming platform

DevOps & Infrastructure

Build & Package Management

- **Turborepo** - Monorepo build system
- **PNPM** - Fast, disk space efficient package manager
- **PDM** - Python package manager
- **uv Python** - Fast Python package installer and resolver

Containerization & Deployment

- **Docker** - Containerization platform
- **Swarm** - Container orchestration
- **TSDProxy** - TypeScript-first proxy
- **NGnix** - Web server and reverse proxy
- **Dev Containers** - Consistent development environments

CI/CD

- **GitHub Actions** - Continuous integration and deployment

- **Commitizen** - Standardized commit messages
- **GPTLint** - AI-powered linting
- **Lint-staged** - Run linters on staged files
- **Lefthook** - Git hooks manager

Monitoring & Observability

- **Prometheus** - Monitoring system
- **Grafana** - Observability platform
- **W&B** (Weights & Biases) - ML experiment tracking
- **Opik** - Performance optimization

Security & Validation

- **Clerk** - Authentication and user management
- **Zod** - TypeScript-first schema validation
- **Pydantic** - Data validation for Python

Documentation

- **Mintlify** - Documentation platform
- **Storybook** - Component documentation

Implementation Roadmap

Phase 1: Foundation

1. Set up monorepo structure with Turborepo, PNPM, and PDM
2. Establish CI/CD pipelines with GitHub Actions
3. Configure Docker and development containers
4. Implement core Next.js application with Tailwind and Radix UI

Phase 2: Backend Infrastructure

1. Develop FastAPI and NestJS services
2. Set up database infrastructure (PostgreSQL, MongoDB, Redis)
3. Implement authentication with Clerk
4. Configure monitoring with Prometheus and Grafana

Phase 3: AI Infrastructure

1. Integrate LLM orchestration (Portkey, LiteLLM)
2. Set up vector databases (Pinecone, PGVector)

3. Implement core agent frameworks (LangChain, LangGraph)
4. Develop RAG capabilities with LightRAG

Phase 4: Advanced Features

1. Multi-agent orchestration with CrewAI and AutoGen
2. Browser automation with O-Man/ProxyLite
3. Advanced analytics and monitoring
4. 3D visualization with Three.js/R3F

Phase 5: Optimization & Scale

1. Performance optimization with Opik
2. Distributed computing with Ray
3. Enhanced observability with Langfuse and W&B
4. Production hardening and security enhancements

Penumbra: Intelligent Ops for the AI-native enterprise

Stack Proposal Diagram 1

Page

Stack Proposal Diagram 1

Markdown

```
flowchart TB
    %% Main Platform
    Platform["PENUMBRA PLATFORM"]

    %% Main Sections
    Frontend["FRONTEND APPLICATIONS"]
    FrontendTech["FRONTEND TECHNOLOGY"]
    Backend["API & BACKEND SERVICES"]
    AIInfra["AI AGENT INFRASTRUCTURE"]
    DataInfra["DATA INFRASTRUCTURE"]
    DevOps["DEVOPS & INFRASTRUCTURE"]
    Security["SECURITY & VALIDATION"]
    Docs["DOCUMENTATION"]

    %% Frontend Applications
    FE1["Next.js WebUI"]
    FE2["Agent Dashboard"]
    FE3["Prompt Studio"]
    FE4["Analytics & Monitoring"]

    %% Frontend Technology Groups
    FT1["UI Components & Design"]
    FT2["Animation & Visualization"]
    FT3["State & Data Fetching"]

    %% Frontend Tech Components
    FT1_1["Radix UI / ShadCN"]
    FT1_2["Tailwind 4 / 21st Dev"]
    FT1_3["Supernova / Composio / Storybook"]
    FT2_1["Framer Motion / Anime.js"]
    FT2_2["Three.js / R3F / Drei"]
    FT3_1["Zustand / Jotai"]
    FT3_2["Tanstack Query"]

    %% Backend Services
    BE1["FastAPI"]
    BE2["NestJS"]
    BE3["GraphQL"]
    BE4["API Gateway"]

    %% AI Infrastructure
    AI1["LLM ORCHESTRATION"]
    AI2["AGENT FRAMEWORKS"]

    %% LLM Orchestration
    AI1_1["Portkey / OpenRouter"]
    AI1_2["LiteLLM / LangWatch"]
    AI1_3["Langfuse / PromptFoo"]

    %% Agent Frameworks
    AI2_1["LangChain / LangGraph"]
```

```
    AI2_2["CrewAI / AutoGen"]
    AI2_3["Semantic Kernel"]
    AI2_4["Google ADK / A2A Protocol"]
    AI2_5["Flowise / Langflow"]
    AI2_6["O-Man / ProxyLite"]

    %% Data Infrastructure
    Data1["VECTOR STORES"]
    Data2["DATABASES"]
    Data3["DATA PROCESSING"]
    Data4["MESSAGING & CACHE"]

    %% Vector Stores
    Data1_1["Pinecone"]
    Data1_2["Weaviate"]
    Data1_3["PGVector"]

    %% Databases
    Data2_1["PostgreSQL / Neon"]
    Data2_2["MongoDB"]
    Data2_3["Turso"]
    Data2_4["AstraDB"]

    %% Data Processing
    Data3_1["dbt"]
    Data3_2["Ray"]
    Data3_3["LightRAG"]
    Data3_4["Crawl4AI"]

    %% Messaging & Cache
    Data4_1["Redis"]
    Data4_2["iKafka"]

    %% DevOps
    Dev1["BUILD & PACKAGE"]
    Dev2["CONTAINERIZATION"]
    Dev3["CI/CD"]
    Dev4["MONITORING"]

    %% Build & Package
    Dev1_1["Turborepo / PNPM"]
    Dev1_2["PDM / uv Python"]

    %% Containerization
    Dev2_1["Docker / Swarm"]
    Dev2_2["TSDProxy / NGnix"]
    Dev2_3["Dev Containers"]

    %% CI/CD
    Dev3_1["GitHub Actions"]
    Dev3_2["Commitizen / GPTLint"]
```

```
    Dev3_3["Lint-staged / Lefthook"]

    %% Monitoring
    Dev4_1["Prometheus / Grafana"]
    Dev4_2["W&B / Opik"]

    %% Security & Validation
    Sec1["Clerk"]
    Sec2["Zod / Pydantic"]

    %% Documentation
    Doc1["Mintlify"]
    Doc2["Storybook"]

    %% Main Flow
    Platform --> Frontend
    Platform --> FrontendTech
    Platform --> Backend
    Platform --> AIInfra
    Platform --> DataInfra
    Platform --> DevOps
    Platform --> Security
    Platform --> Docs

    %% Frontend Applications
    Frontend --- FE1
    Frontend --- FE2
    Frontend --- FE3
    Frontend --- FE4

    %% Frontend Technology
    FrontendTech --- FT1
    FrontendTech --- FT2
    FrontendTech --- FT3

    FT1 --- FT1_1
    FT1 --- FT1_2
    FT1 --- FT1_3
    FT2 --- FT2_1
    FT2 --- FT2_2
    FT3 --- FT3_1
    FT3 --- FT3_2

    %% Backend Services
    Backend --- BE1
    Backend --- BE2
    Backend --- BE3
    Backend --- BE4

    %% AI Infrastructure
    AIInfra --- AI1
```

```
AIInfra --- AI2

AI1 --- AI1_1
AI1 --- AI1_2
AI1 --- AI1_3

AI2 --- AI2_1
AI2 --- AI2_2
AI2 --- AI2_3
AI2 --- AI2_4
AI2 --- AI2_5
AI2 --- AI2_6

%% Data Infrastructure
DataInfra --- Data1
DataInfra --- Data2
DataInfra --- Data3
DataInfra --- Data4

Data1 --- Data1_1
Data1 --- Data1_2
Data1 --- Data1_3

Data2 --- Data2_1
Data2 --- Data2_2
Data2 --- Data2_3
Data2 --- Data2_4

Data3 --- Data3_1
Data3 --- Data3_2
Data3 --- Data3_3
Data3 --- Data3_4

Data4 --- Data4_1
Data4 --- Data4_2

%% DevOps
DevOps --- Dev1
DevOps --- Dev2
DevOps --- Dev3
DevOps --- Dev4

Dev1 --- Dev1_1
Dev1 --- Dev1_2

Dev2 --- Dev2_1
Dev2 --- Dev2_2
Dev2 --- Dev2_3

Dev3 --- Dev3_1
Dev3 --- Dev3_2
```

```
    Dev3 --- Dev3_3

    Dev4 --- Dev4_1
    Dev4 --- Dev4_2

    %% Security & Validation
    Security --- Sec1
    Security --- Sec2

    %% Documentation
    Docs --- Doc1
    Docs --- Doc2

    %% Styling
    classDef platform fill:#f9f,stroke:#333,stroke-width:2px
    classDef section fill:#bbf,stroke:#333,stroke-width:1px
    classDef subsection fill:#ddf,stroke:#333,stroke-width:1px
    classDef component fill:#fff,stroke:#333,stroke-width:1px

    class Platform platform
    class
Frontend,FrontendTech,Backend,AIInfra,DataInfra,DevOps,Security,Docs
section
    class
FE1,FE2,FE3,FE4,FT1,FT2,FT3,BE1,BE2,BE3,BE4,AI1,AI2,Data1,Data2,Data3,Da
ta4,Dev1,Dev2,Dev3,Dev4,Sec1,Sec2,Doc1,Doc2 subsection
    class
FT1_1,FT1_2,FT1_3,FT2_1,FT2_2,FT3_1,FT3_2,AI1_1,AI1_2,AI1_3,AI2_1,AI2_2,
AI2_3,AI2_4,AI2_5,AI2_6,Data1_1,Data1_2,Data1_3,Data2_1,Data2_2,Data2_3,
Data2_4,Data3_1,Data3_2,Data3_3,Data3_4,Data4_1,Data4_2,Dev1_1,Dev1_2,De
v2_1,Dev2_2,Dev2_3,Dev3_1,Dev3_2,Dev3_3,Dev4_1,Dev4_2 component
```

▼ DevSecOps

Page

GitHub Repository Security Best Practices

InfoSec & DevSecOps for GitHub Repos

These standards will help secure projects we are 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 all 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

I propos **"Shift Left"** in our **SOP**s, which is an excellent security approach:

1. Early Security Integration

- Integrate security checks at the earliest stages of development
- Provide security training for developers / contribitors
- 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.

Page

Repository Configuration Files for Security & Best Practices

---

5 Additional Security Files

5.1 dependabot.yml

5.2 SECURITY.md

5.3 .dockerignore

5.4 .pre-commit-config.yaml

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:

```
Text
# Environment variables
.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*
pnpm-debug.log*
pnpm-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
  plaintext

```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

```
Text
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

```
Text
22.12.12
```

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
```

Page

Implementing dotenv-vault and Tailscale for Repository Security

---

Setup 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

GitHub - dotenv-org/dotenv-vault: sync .env files—from the creator of `dotenv`.
https://github.com/dotenv-org/dotenv-vault

1. Initial Setup

```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:

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

3. GitHub Workflow Integration

```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

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

I will create runbooks so team members can press play button in the `README.md` file and automate setup

Add these to your README.md:

Markdown

```
## Environment Variables
We use dotenv-vault for secure environment management.
### Setup
```bash
# 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.

Tailscale · Best VPN Service for Secure Networks
https://tailscale.com/

1. GitHub Actions Integration for Tailscale
```

YAML

```yaml
# .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

```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
}
```

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

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 / Update `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.

Resource Gathering

▢ 5 Agent Frameworks & 3 Protocols

Frameworks

▢ AG2


AG2 - ag2
https://docs.ag2.ai/latest/


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

▢ Google ADK

Agent Development Kit
https://google.github.io/adk-docs/

▢ AutoGPT

AutoGPT Documentation
https://dev-docs.agpt.co/


GitHub - Significant-Gravitas/AutoGPT: AutoGPT is the vision of accessible AI for everyone, to use a...
https://github.com/Significant-Gravitas/AutoGPT

▢ Agno


What is Agno - Agno
https://docs.agno.com/introduction

▢ Lang [____]

- ▢ LangChain
- ▢ LangGraph
- ▢ Langflow | Low-code AI builder for agentic and RAG applications
- ▢ Langfuse
- ▢ LangSmith

- Introduction - LangWatch
- Open Agent Platform
  - Quick Start - Open Agent Platform
    https://docs.oap.langchain.com/quickstart
- Just a moment...

Protocols

- A2A (google)
  - Agent2Agent Protocol
    https://google.github.io/A2A/#/
- Agent Protocol (Langchain)
  - GitHub - langchain-ai/agent-protocol
    https://github.com/langchain-ai/agent-protocol
- Agent Protocol (AutoGPT)
  - GitHub - Significant-Gravitas/agent-protocol: Common interface for interacting with AI agents. The p...
    https://github.com/Significant-Gravitas/agent-protocol
  - Agent Protocol
    https://agentprotocol.ai/

- Agent IDEs
  - AG2 Studio
    - GitHub - ag2ai/ag2studio: AG2 Studio
      https://github.com/ag2ai/ag2studio
  - LangGraph Studio
    - Overview
      https://langchain-ai.github.io/langgraph/concepts/langgraph_studio/
  - Letta ADE
    - Letta
      https://app.letta.com/development-servers/local/agents/agent-a44fb81e-fd9c-48ed-b119-31ab219b5567
  - Letta Docs
    - Agent Development Environment (ADE) | Letta
      https://docs.letta.com/guides/ade/overview

- ++ Agentic
  - Inngest
    - Inngest - AI and backend workflows, orchestrated at any scale
      https://www.inngest.com/
    - GitHub - inngest/inngest: The leading workflow orchestration platform. Run stateful step functions a...
      https://github.com/inngest/inngest
  - Inngest Agent Kit
    - GitHub - inngest/agent-kit: AgentKit: Build multi-agent networks in TypeScript with deterministic ro...
      https://github.com/inngest/agent-kit

    Agent Kit Reddit Search BrowserBase

    - agent-kit/examples/reddit-search-browserbase-tools at main · inngest/agent-kit
      https://github.com/inngest/agent-kit/tree/main/examples/reddit-search-browserbase-tools

---

No Code / Low Code

Flowize AI

- Flowise - Build AI Agents, Visually
  https://flowiseai.com/
- GitHub - FlowiseAI/Flowise: Build AI Agents, Visually
  https://github.com/FlowiseAI/Flowise

Langflow

- Langflow | Low-code AI builder for agentic and RAG applications
  https://www.langflow.org/

- **Scrape & Crawl**
  - GitHub - unclecode/crawl4ai: 🚀🤖 Crawl4AI: Open-source LLM Friendly Web Crawler & Scraper. Don't be...
    https://github.com/unclecode/crawl4ai

  LightRAG

  - GitHub - HKUDS/LightRAG: "LightRAG: Simple and Fast Retrieval-Augmented Generation"
    https://github.com/HKUDS/LightRAG

- **Browser Control**

Browser Use

- [ ] Browser Use Agents & Tools — Resource

Autonomous

Browser User


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

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

mcp-browser-use/src/mcp_server_browser_use/server.py at main · Saik0s/mcp-browser-use
https://github.com/Saik0s/mcp-browser-use/blob/main/src/mcp_server_browser_use/server.py

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

Open Manus

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

Open Operator

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

Steel

- [ ] Steel | Open-source Headless Browser API
https://steel.dev/

BrowserBase

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

Production

▢ Cloudflare

- [ ] Welcome to Cloudflare
  https://developers.cloudflare.com/

- [ ] DNS
  - [ ] Move Domain Name to CF

- [ ] Security / WAF
  - [ ] Deploy Managed Rules
  - [ ] Deploy Robots.txt

Stack As TODOs

(Inside) DevAgents & DevTools

- [x] Agent Zero — Task
- [x] BrainCraft — Task
- [x] GPT4Free — Task
- [x] Aider — Task
- [x] Genaroo — Task

System Architecture

- [x] Capture Remaining Docs — Task
- [x] Synthesize Into PRD & Description for GenA-Roo — Task

Database

- [x] SurrealDB & Surrealist — Task

SpacetimeDB

- [ ] SpacetimeDB
  https://spacetimedb.com/docs

Neon & Turso

- [ ] Neon and Turso for Penumbra — Page

Neon

- [ ] Neon Serverless Postgres — Ship faster
https://neon.tech/
- GitHub - neondatabase/neon: Neon: Serverless Postgres. We separated storage and compute to offer aut...
https://github.com/neondatabase/neon

Turso

- [ ] Turso - SQLite Databases for all Apps
https://turso.tech/
- Welcome to Turso Cloud - Turso
https://docs.turso.tech/introduction

Memory

- [x] Mem0 Task
- [x] Letta (MemGPT) Task
- [x] MemGraph Task

GenUI

Crayon

- [ ] Introduction | CrayonAI
https://crayonai.org/docs/
- GitHub - thesysdev/crayon: Generative UI SDK
https://github.com/thesysdev/crayon
- examples/crayon at main · thesysdev/examples
https://github.com/thesysdev/examples/tree/main/crayon
- @storybook/core - Storybook
https://crayonai.org/ui/?path=/docs/components-accordion--docs

Infra & IaC

- [x] Temporal Task

DevSecOps & CI/CD

Docker

- [ ] Docker Engine
https://docs.docker.com/engine/

MCP Catalog & Toolkit

- [ ] MCP Catalog and Toolkit

https://docs.docker.com/ai/mcp-catalog-and-toolkit/

GH Copilot

- [ ] Docker for GitHub Copilot
https://docs.docker.com/copilot/

Docker Compose

- [ ] Docker Compose
https://docs.docker.com/compose/

Docker Bake

- [ ] Bake
https://docs.docker.com/build/bake/

Testcontainers Cloud

- [ ] Docs for Testcontainers Cloud
https://testcontainers.com/cloud/docs/

k8s

- [ ] Production-Grade Container Orchestration
https://kubernetes.io/

Dagger

- [ ] Dagger.io
https://dagger.io/

Dagger Agents

- [ ] Dagger Agents Page

Uffizzi

- [ ] Uffizzi – Uffizzi
https://docs.uffizzi.com/
- [ ] Uffizzi app
https://app.uffizzi.com/account_settings#integrations

Temporal

- [ ] Temporal Platform Documentation | Temporal Platform Documentation
https://docs.temporal.io/

GitHub Actions

- [ ] GitHub Actions
https://github.com/features/actions
- [ ] GitHub Actions documentation - GitHub Docs

https://docs.github.com/en/actions

GitHub Dev Program

GitHub Developer Program - GitHub Docs
https://docs.github.com/en/get-started/exploring-integrations/github-developer-program

LLMetry & Ops

Promptfoo

Promptfoo - Portkey Docs
https://portkey.ai/docs/integrations/libraries/promptfoo

Comet / Opik

Opik MCP

GitHub - comet-ml/opik-mcp: Model Context Protocol (MCP) implementation for Opik enabling seamless I...
https://github.com/comet-ml/opik-mcp

Opik

- Debug, trace, evaluate and monitor LLM apps and RAG pipelines

Comet Opik
https://www.comet.com/opik/areid987/home

Opik GitHub

GitHub - comet-ml/opik: Debug, evaluate, and monitor your LLM applications, RAG systems, and agentic...
https://github.com/comet-ml/opik

Comet Docs

Docs Home - Comet Docs
https://www.comet.com/docs/v2/

Comet GitHub

Comet
https://github.com/comet-ml

AgentOps

AgentOps
https://www.agentops.ai/

Agenta

What is Agenta? - Docs - Agenta

https://docs.agenta.ai/

GitHub - Agenta-AI/agenta: The open-source LLMOps platform: prompt playground, prompt management, LL...
https://github.com/agenta-ai/agenta

Langwatch

- [ ] Introduction - LangWatch
https://docs.langwatch.ai/introduction

Service Mesh

Kong API & AI

Kuma

Kuma Service Mesh — Page

Kong Gateway

- [ ] Installation Options - Kong Gateway | Kong Docs
https://docs.konghq.com/gateway/latest/install/#kong-community

Portkey

- [ ] What is Portkey? - Portkey Docs
https://portkey.ai/docs/introduction/what-is-portkey

Not Diamond

- [ ] What is Not Diamond?
https://docs.notdiamond.ai/docs/what-is-not-diamond

LiteLLM

- [ ] LiteLLM
https://www.litellm.ai/
- [ ] Docker, Deployment | liteLLM
https://docs.litellm.ai/docs/proxy/deploy

CMS

Prismic

- [ ] Developer Documentation - Prismic
https://prismic.io/docs

More Agentic Frameworks

Dagger

- Dagger Documentation | Dagger
  https://docs.dagger.io/
- GitHub - dagger/agents
  https://github.com/dagger/agents

Langbase ICP Insight Team Blogger

- icp-insight-team-agent-0677 - areid987
  https://chai.new/areid987/icp-insight-team-agent-0677

AI Devin

- Build a composable AI Devin - Guides
  https://langbase.com/docs/guides/build-composable-ai-devin
- DataStax Langflow | DataStax Docs
  https://docs.datastax.com/en/langflow/index.html
- Datastax Astra
  https://astra.datastax.com/org/67b88500-04ee-48b8-a773-eabdc1459116
- Untitled
  https://oap.langchain.com/?agentId=eaad1967-3353-4897-a053-3d6ee59dffbe&deploymentId=ee18b042-9d0a-4994-8920-f8af1604a454&threadId=84e5b5de-1220-4603-932a-bf44d23a0d01
- Code Alchemist - Write code or design database with AI
  https://code-alchemist.langbase.dev/
- Agentic Generative UI
  https://docs.copilotkit.ai/coagents/generative-ui/agentic

ML Ops

Prime

- GitHub - PrimeIntellect-ai/prime: prime is a framework for efficient, globally distributed training ...
  https://github.com/PrimeIntellect-ai/prime

Q Learn

- GitHub - Div99/IQ-Learn: (NeurIPS '21 Spotlight) IQ-Learn: Inverse Q-Learning for Imitation
  https://github.com/Div99/IQ-Learn

Graph Resources

- Graph Resources — Resource