

# CurioAI - Software Requirements Specification

**Version:** 2.0

**Date:** November 8, 2025

**Author:** Rameswar Panda

**Status:** Draft

---

## 1. Executive Summary

**CurioAI** is a privacy-first personal knowledge engine that transforms passive content consumption into an intelligent, interconnected learning system. It automatically captures, summarizes, and visualizes everything you learn across browsers, PDFs, and videos—building your personal AI brain.

**Core Innovation:** Unlike note-taking apps or bookmarking tools, CurioAI autonomously understands relationships between concepts you learn over time, creating a living knowledge graph that grows with you.

---

## 2. Product Vision

### 2.1 Problem Statement

- People consume hours of content daily but retain < 10%
- No tool connects knowledge across different sources automatically
- Existing solutions require manual input (Notion, Obsidian) or lack AI intelligence

### 2.2 Solution

A background AI system that:

1. **Observes** your learning activity (browser, PDFs, videos, code)
2. **Understands** content using generative AI
3. **Connects** concepts into a personal knowledge graph
4. **Retrieves** information through natural conversation

### 2.3 Key Differentiators

Feature	CurioAI	Notion	Mem.ai	Obsidian
Auto-capture content	✓	✗	✗	✗

Feature	CurioAI	Notion	Mem.ai	Obsidian
AI knowledge graph	✓	✗	Partial	Manual
Local-first privacy	✓	✗	✗	✓
Agentic AI insights	✓	✗	✗	✗
Chat with your data	✓	✗	✓	Plugin
Code activity tracking	✓	✗	✗	✗

### 3. Activity Tracking Strategy

#### 3.1 What We Track (Learning-Focused Only)

CurioAI focuses exclusively on activities that reflect **learning, curiosity, or creation** — not general browsing or entertainment.

##### Browsing & Online Learning

Target	What We Capture	How We Capture	Notes
<b>Web Browser</b> (Chrome/Edge/Firefox)	Active tab URL, page title, page text	Browser extension + OS-level window detection ( <code>active-win</code> )	Whitelist: YouTube, Medium, dev.to, Wikipedia, arXiv, etc.
<b>YouTube / Coursera / Udemy</b>	Video title, channel/course, transcript	YouTube API, transcript scraper	Extract key topics, store embeddings
<b>Medium / Blogs / Papers</b>	Article title, body text, tags	Puppeteer automation or browser plugin	User approves per domain

**Purpose:** Learn your intellectual interests (tech, AI, design, etc.)

##### Reading & Study Materials

Target	What We Capture	How We Capture	Notes
<b>PDFs / eBooks</b>	File name, text content, folder path	File watcher on Documents/Downloads using <code>pdf-parse</code>	Only when opened; ask folder permission
<b>Research Papers / Notes</b>	Summaries, extracted keywords	Local LLM processing	Enriches learning map

**Purpose:** Build long-term knowledge graph of reading history

## Coding & Creation

Target	What We Capture	How We Capture	Notes
VS Code / IDEs	Project name, languages used, recent files	Detect active window = "Code", watch workspace folders	Understand technical focus
Terminal	Commands executed (optional)	Shell history reader (opt-in only)	Summarize dev activity like "learning Docker"
Design Tools (Figma, Canva)	File title, tool name	Window title capture	Optional creative activity logging

**Purpose:** Connect what you read and what you build (e.g., "Watched React hooks video → used in code")

## Writing & Notes

Target	What We Capture	How We Capture	Notes
Notion / Obsidian / Word	File/document title, optional summary	Window title + text extraction	Link ideas across notes
ChatGPT / Claude / Gemini	Prompt topics (if authorized)	Window title monitoring	Detect conversational exploration

**Purpose:** Add context about thought process and questions

## Miscellaneous (Future Phase)

Target	Why	How
Calendar	Detect scheduled learning/work	Local calendar API
System Audio	Detect lectures/podcasts	Whisper transcription
File System	Detect new downloads	File watcher

## 3.2 Technical Detection Methods

### Active Window Detection

```
javascript

// Using active-win library (Node/Electron)
import activeWindow from 'active-win';

const window = await activeWindow();
// Returns: {title, id, owner: {name, process}}
```

## Browser Extension (Optional)

- Injects into Chrome/Firefox for deeper page access
- Extracts full article text, highlighted sections
- Detects scroll depth (engagement metric)

## File System Watcher

```
javascript

// Using chokidar
chokidar.watch('/Users/*/*/*/*.pdf')
.on('change', path => processPDF(path));
```

## Screen OCR (Optional)

- For apps without API access (e.g., proprietary readers)
- Uses Tesseract.js for text extraction

---

## 4. Desktop Application Architecture

### 4.1 Core Features

Feature	Description	Why
 Activity Tracker	Detects active window, app name, URL, title	Builds context of what user is doing
 Content Extractor	Reads content from browsers, PDFs, YouTube transcripts	Converts raw activity into text
 LLM Summarizer	Generates summary + key concepts + sentiment	Saves storage, keeps distilled insights
 Embedding Generator	Converts text to vector representation	Enables semantic search & topic linking
 Local Knowledge DB	Stores summaries, embeddings, relationships	Offline-first memory base
 Graph Builder	Links similar nodes (e.g., "LangChain" ↔ "LlamaIndex")	Creates dynamic knowledge network
 Sync Service	Sends anonymized graph data to cloud	Powers web dashboard (optional, encrypted)

Feature	Description	Why
>User Control Panel	Tray app: pause/resume, last activity, privacy toggles	Transparency & trust

## 4.2 Advanced Features (Phase 2)

Feature	Description
🧠 Contextual Reminder Agent	Suggests "You read about this before..." insights
📁 Project Folder Tracking	Detects coding projects, adds to knowledge graph
🎙 Speech Listener	Transcribes podcasts/YouTube audio (Whisper)
🔍 Local Search	Semantic search through all content
🔒 Local-only Mode	Disables cloud sync completely
🌙 Nightly Review	Generates daily digest of learned topics

## 4.3 Technology Stack



## 5. Web Application Features

### 5.1 Core Features

Feature	Description	Why
 <b>Knowledge Graph Visualizer</b>	Interactive 3D/2D graph with zoom/pan, color-coded clusters	See exploration patterns visually
 <b>Learning Timeline</b>	Chronological view of discoveries	Track knowledge evolution
 <b>AI Chat Assistant</b>	Natural language queries: "What did I learn about React?"	Conversational insight retrieval
 <b>Topic Summaries</b>	Auto-generated summaries per topic cluster	Quick recall and reinforcement
 <b>Recommendations Engine</b>	Suggests related videos, papers, blogs	Keeps curiosity loop going
 <b>Search Everything</b>	Semantic search across all content	Unified recall mechanism
 <b>Settings &amp; Privacy Dashboard</b>	Manage sync, delete data, view permissions	Full transparency + control

### 5.2 Advanced Features (Phase 2)

Feature	Description
 <b>Global Discovery Feed</b>	(Opt-in) Anonymized trending topics others are learning
 <b>Collab Mode</b>	Compare learning graphs with friends
 <b>Weekly Insights</b>	"You learned 15 new concepts, 3 are trending in AI"
 <b>AI Mentor</b>	Personalized learning path suggestions
 <b>Content Notebook</b>	Store favorite summaries, highlights, notes
 <b>Encrypted Backup/Restore</b>	Cloud-based private sync

### 5.3 Technology Stack

#### Frontend

- Framework: Next.js 14 (App Router)
- Visualization: D3.js + Cytoscape.js

- State: Zustand + TanStack Query
- UI: Tailwind CSS + Shadcn/ui
- Auth: NextAuth.js

## Graph Visualization

javascript

```
// Example using Cytoscape.js
const cy = cytoscape({
  container: document.getElementById('graph'),
  elements: [
    { data: { id: 'react', label: 'React' } },
    { data: { id: 'hooks', label: 'Hooks' } },
    { data: { source: 'react', target: 'hooks' } }
  ],
  style: /*force-directed layout */
});
});
```

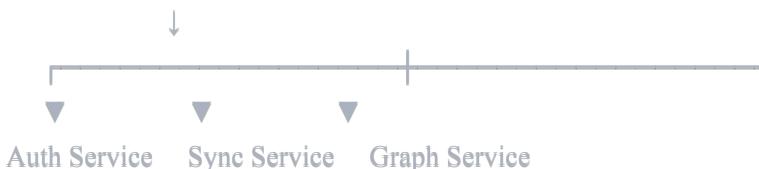
## 6. Backend Microservices Architecture

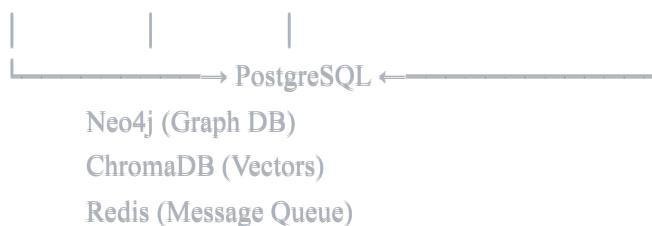
### 6.1 Service Breakdown

Microservice	Responsibility	Stack	Scaling
<b>auth-service</b>	User login, tokens, encryption	Node.js + PostgreSQL	Horizontal pods
<b>sync-service</b>	Receives desktop updates, stores embeddings	FastAPI + ChromaDB	Worker pods
<b>summarizer-service</b>	Summarizes text using LLMs	Python + LangChain	Autoscale
<b>graph-service</b>	Builds and updates knowledge graphs	Python + Neo4j	Autoscale
<b>recommendation-service</b>	Suggests next content	Python + FAISS	Low-latency
<b>web-api-service</b>	Serves frontend queries	Node.js + Express	Load-balanced
<b>chat-service</b>	Manages RAG-based conversations	FastAPI + LangChain	GPU-enabled pods

### 6.2 Communication Architecture

Desktop Agent → (HTTPS) → API Gateway (Nginx)





## Inter-Service Communication

- Synchronous: REST APIs for real-time queries
- Asynchronous: Redis Streams / RabbitMQ for event-driven processing

## 6.3 Containerization (Docker + Kubernetes)

### Docker Compose (Local Development)

```

yaml

services:
  auth-service:
    build: ./services/auth
    ports: ["3001:3001"]
    depends_on: [postgres]

  sync-service:
    build: ./services-sync
    ports: ["8001:8001"]
    depends_on: [chromadb, redis]

  postgres:
    image: postgres:15
    volumes: ["./data/postgres:/var/lib/postgresql/data"]

  chromadb:
    image: chromadb/chroma:latest

  redis:
    image: redis:7-alpine
  
```

### Kubernetes Deployment (Production)

- Minikube / K3s for local testing
- GCP GKE / AWS EKS / Azure AKS for cloud

- Horizontal Pod Autoscaling (HPA) based on CPU/memory
  - Prometheus + Grafana for monitoring
- 

## 7. Generative AI Components

### 7.1 Core AI Functions

Function	Description	Technology
<b>Summarization</b>	Condenses content into 2-3 paragraphs + bullets	GPT-4 / Claude-3 / Ollama + Mistral
<b>Concept Extraction</b>	Detects important entities (topics, frameworks, names)	NER + GPT-based keyword extraction
<b>Knowledge Graph Expansion</b>	Generates relationships: "LangChain → built on Python"	GPT-powered inference
<b>Personal Digest Generation</b>	Creates "Here's what you explored today"	LLM prompt chaining
<b>Chat with Knowledge</b>	RAG pipeline: search embeddings → LLM response	LangChain + OpenAI API
<b>Recommendation Engine</b>	"You've been learning LLMs — try this paper"	Vector similarity + LLM reasoning

### 7.2 Prompt Engineering Examples

#### Summarization Prompt

You are a personal learning assistant. Summarize the following content:

- Extract 3-5 key concepts
- Identify the main topic
- Rate complexity (beginner/intermediate/advanced)
- Suggest related topics to explore

Content: {extracted\_text}

#### Graph Relationship Prompt

Given these two concepts from a user's learning history:

Concept A: {concept\_a}

Concept B: {concept\_b}

Determine if they are related. If yes, describe the relationship in one sentence.

Format: "Concept A [relationship] Concept B"

Example: "React [uses] JavaScript"

## 8. Agentic AI System

### 8.1 Agent Types & Roles

Agent	Role	Trigger	Technology
Observer Agent	Monitors desktop context, tags data	24x7 background process	Electron + Node.js
Summarizer Agent	Runs summarization + embedding generation	Event-triggered (new content)	Python + LangChain
Graph Builder Agent	Links related nodes, updates edges	Every 30 minutes	NetworkX + Neo4j
Chat Agent	Handles RAG-based user queries	User-triggered	LangChain + OpenAI
Mentor Agent	Suggests learning paths	Daily/weekly schedule	GPT-4 reasoning
Sync Agent	Syncs anonymized data to cloud	Time-triggered (cron)	FastAPI + encryption

### 8.2 Agent Communication Flow

Observer Agent → Redis Stream (event: "new\_pdf\_opened")



Summarizer Agent (consumes event)



Store in SQLite + ChromaDB



Graph Builder Agent (periodic check)



Update Neo4j relationships

### Technology Stack for Agents

- Orchestration: LangGraph / CrewAI
  - Message Queue: Redis Streams / RabbitMQ
  - State Management: Redis (shared memory)
  - Scheduling: APScheduler (Python)
- 

## 9. Data Architecture

### 9.1 Local Storage (Desktop)

#### SQLite Schema

sql

```
CREATE TABLE activities (
    id INTEGER PRIMARY KEY,
    url TEXT,
    title TEXT,
    content TEXT,
    timestamp DATETIME,
    source_type TEXT, -- 'browser', 'pdf', 'code', 'video'
    app_name TEXT
);
```

```
CREATE TABLE summaries (
    id INTEGER PRIMARY KEY,
    activity_id INTEGER,
    summary_text TEXT,
    key_concepts JSON, -- ["React", "Hooks", "useState"]
    complexity TEXT, -- 'beginner', 'intermediate', 'advanced'
    sentiment FLOAT,
    FOREIGN KEY (activity_id) REFERENCES activities(id)
);
```

```
CREATE TABLE embeddings (
    id INTEGER PRIMARY KEY,
    summary_id INTEGER,
    vector BLOB, -- 768-dim embedding
    model_version TEXT,
    FOREIGN KEY (summary_id) REFERENCES summaries(id)
);
```

## ChromaDB (Vector Store)

```
python

import chromadb

client = chromadb.Client()
collection = client.create_collection("knowledge_base")

# Store embedding
collection.add(
    documents=["React Hooks allow..."],
    metadatas=[{"topic": "React", "date": "2025-11-08"}],
    ids=["summary_123"]
)

# Search similar concepts
results = collection.query(
    query_texts=["useState hook"],
    n_results=5
)
```

## 9.2 Cloud Storage (Backend)

### PostgreSQL

- User accounts, preferences
- Activity metadata (anonymized)
- Subscription and billing data

### Neo4j (Knowledge Graph)

```
cypher
```

```
// Example graph structure
CREATE (react:Topic {name: "React", category: "Frontend"})
CREATE (hooks:Concept {name: "Hooks", introduced: "v16.8"})
CREATE (useState:Concept {name: "useState"})

CREATE (react)-[:HAS_FEATURE]->(hooks)
CREATE (hooks)-[:INCLUDES]->(useState)
```

```
// Query: Find related concepts
MATCH (t:Topic {name: "React"})-[:HAS_FEATURE*1..3]->(related)
RETURN related
```

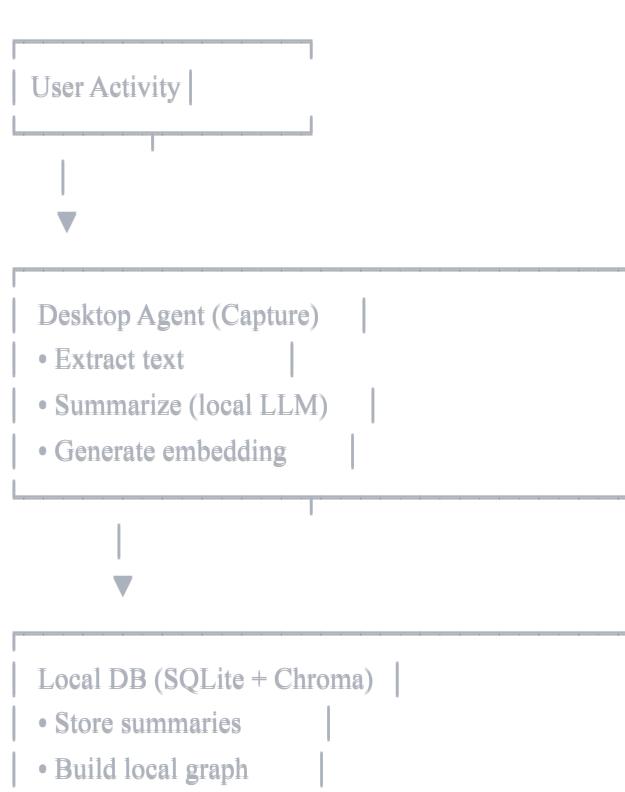
## ChromaDB (Cloud)

- Persistent vector embeddings
- Powers semantic search in web app

## Redis

- Session cache
- Agent message queue
- Rate limiting

## 9.3 Data Flow Diagram





## 10. Privacy & Security

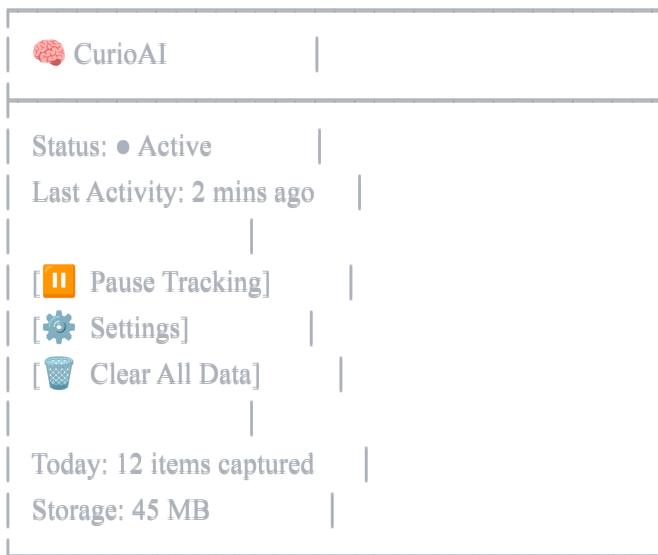
### 10.1 Privacy-First Design

- **Local-first:** All processing happens on device by default
- **Opt-in sync:** User explicitly enables cloud features
- **Data ownership:** Export full data anytime (JSON/CSV)
- **Anonymization:** Cloud sync removes PII before upload
- **Granular controls:** Blacklist specific apps/domains
- **Pause anytime:** System tray toggle to stop tracking

### 10.2 Security Measures

- **AES-256 encryption** for local storage
- **E2E encryption** for sync (user-controlled keys)
- **JWT authentication** with refresh tokens
- **TLS 1.3** for all API communication
- **SOC 2 Type II compliance** (future goal)
- **No third-party analytics** in desktop app

## 10.3 User Control Panel (Desktop)

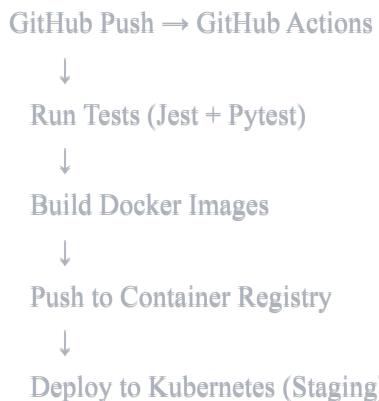


### Settings Options

- Enable/disable per-app tracking
- Blacklist specific websites
- Set capture frequency (every 1/5/15 mins)
- Local-only mode (no cloud sync)
- Auto-delete after X days
- Export data (JSON/CSV)

## 11. DevOps & Monitoring

### 11.1 CI/CD Pipeline



↓  
Run Integration Tests

↓  
Deploy to Production (if tests pass)

## 11.2 Monitoring Stack

Function	Tool	Metrics
Infrastructure Monitoring	Prometheus + Grafana	CPU, memory, disk, network
Log Aggregation	ELK Stack (Elasticsearch, Logstash, Kibana)	Application logs, errors
APM (Application Performance)	Sentry	Error tracking, performance
LLM Monitoring	LangSmith / Traceloop	Token usage, latency, costs
Uptime Monitoring	UptimeRobot	Service availability

## 11.3 Model & Experiment Tracking

### MLflow Integration

```
python  
  
import mlflow  
  
# Track summarization model performance  
mlflow.log_param("model", "gpt-4")  
mlflow.log_metric("avg_summary_length", 150)  
mlflow.log_metric("processing_time_sec", 2.3)
```

## A/B Testing Framework

- Test different summarization prompts
- Measure user engagement with recommendations
- Track graph visualization interaction rates

---

## 12. Pricing Model

### 12.1 Freemium Tiers

Feature	Free	Pro (\$9/mo)	Enterprise
Desktop captures/day	50	Unlimited	Unlimited

Feature	Free	Pro (\$9/mo)	Enterprise
Cloud storage	500 MB	10 GB	Custom
AI chat queries	30/day	500/day	Unlimited
Knowledge graph nodes	1,000	50,000	Unlimited
Advanced analytics	✗	✓	✓
Team collaboration	✗	✗	✓
Priority support	✗	✓	✓
Custom AI models	✗	✗	✓
API access	✗	Limited	Full
Data retention	30 days	1 year	Forever

## 12.2 Revenue Streams

### 1. Subscriptions (Primary Revenue)

- Pro: \$9/month or \$90/year (save 17%)
- Enterprise: Custom pricing (starting \$50/user/month)

### 2. API Access

- Developers integrate CurioAI: \$0.01/query
- Embedding API: \$0.0001/token

### 3. White-label Licensing

- Universities/corporations: \$10K-50K/year
- Custom deployment + branding

### 4. Premium LLM Access

- Pay-per-token for GPT-4/Claude
- 20% markup on API costs

## 12.3 Growth Strategy

### Phase 1: Beta (Months 1-3)

- Free for all users
- Collect feedback, iterate fast
- Goal: 1,000 active users

### Phase 2: Launch (Months 4-6)

- Introduce Pro tier
- Early bird: 50% off for first 500 subscribers
- Goal: 5% conversion rate (50 paying users)

### **Phase 3: Scale (Months 7-12)**

- Launch Enterprise tier
- Partnerships with coding bootcamps/universities
- Goal: 10K users, 500 Pro, 5 Enterprise

### **Phase 4: Expansion (Year 2)**

- Mobile apps (iOS/Android)
  - Team collaboration features
  - International markets (localization)
- 

## **13. Development Roadmap**

### **Phase 1: MVP (Months 1-3)**

#### **Desktop Agent Core**

- Activity tracking (browser, PDFs)
- Local summarization (Ollama)
- SQLite + ChromaDB storage
- Basic graph visualization
- Tray app with controls

#### **Tech Milestones**

- Week 1-2: Electron shell + activity monitor
- Week 3-4: Content extraction (Puppeteer, pdf-parse)
- Week 5-6: Local AI service (FastAPI + Ollama)
- Week 7-8: SQLite schema + ChromaDB integration
- Week 9-10: Basic Neo4j graph builder

- Week 11-12: Desktop UI polish + beta release

## Phase 2: Cloud Integration (Months 4-6)

- User authentication (JWT)
- Cloud sync service (encrypted)
- Web app with graph viewer
- AI chat (RAG pipeline)
- Pro tier launch

### Tech Milestones

- Backend microservices (Docker)
- PostgreSQL + Neo4j Cloud setup
- Next.js web app + D3.js graphs
- LangChain RAG implementation
- Stripe payment integration

## Phase 3: Intelligence (Months 7-9)

- Agentic AI system (6 agents)
- Recommendation engine
- Learning analytics dashboard
- Weekly insights email
- Code activity tracking

### Tech Milestones

- Redis Streams for agent communication
- FAISS for recommendations
- Advanced graph algorithms (NetworkX)
- Email service (SendGrid)
- IDE integration (VS Code extension)

## Phase 4: Scale (Months 10-12)

- Mobile apps (React Native)
- Team collaboration features
- API marketplace
- Enterprise SSO (SAML)
- Multi-language support

### Tech Milestones

- Kubernetes autoscaling (production)
  - React Native app (iOS + Android)
  - GraphQL API for partners
  - Localization (i18n) for 5 languages
  - Enterprise compliance (SOC 2)
- 

## 14. Success Metrics

### 14.1 Product Metrics (North Star)

#### Primary Metric: Weekly Active Knowledge Nodes (WAKN)

- Definition: Unique topics user interacted with in past 7 days
- Target: 50 WAKN per active user

#### Supporting Metrics

Metric	Target (Month 12)
Daily Active Users (DAU)	10,000
Content Captured (per user/week)	50 items
Graph Size (nodes per user)	5,000+
Chat Engagement (queries/user/week)	10
Retention (Day 30)	40%

## 14.2 Business Metrics

Metric	Target (Month 12)
Total Users	10,000
Paying Users (Pro)	500 (5% conversion)
Enterprise Customers	5
Monthly Recurring Revenue (MRR)	\$5,000
Customer Acquisition Cost (CAC)	\$15
Lifetime Value (LTV)	\$108
LTV:CAC Ratio	7:1
Churn Rate	<5% monthly
Net Promoter Score (NPS)	>50

## 14.3 Technical Metrics

Metric	Target
Desktop Agent CPU Usage	<5%
Desktop Agent Memory	<200 MB
Summary Generation Time	<3 seconds
Graph Query Response Time	<500 ms
Web App Load Time	<2 seconds
API Uptime	99.9%
Average Token Cost per User/Month	<\$2

## 15. Risk Analysis

Risk	Impact	Probability	Mitigation
Privacy concerns	High	Medium	Local-first design, transparent policy, EU GDPR compliance
LLM API costs	Medium	High	Optimize prompts, cache results, offer local models (Ollama)
User adoption	High	Medium	Freemium model, viral features (share graphs), university partnerships
Competitors	Medium	High	Speed to market, superior UX, unique agentic AI features
Technical debt	Medium	Medium	Microservices architecture, comprehensive testing, code reviews
Data security breach	High	Low	E2E encryption, regular audits, bug bounty program
LLM hallucinations	Medium	Medium	RAG architecture (grounded in user data), confidence scores

Risk	Impact	Probability	Mitigation
Platform dependencies	Low	Medium	Abstract APIs, support multiple LLM providers (OpenAI, Anthropic, Ollama)

## 16. Future Enhancements (Year 2+)

### 16.1 Advanced Features

#### Voice Interaction

- Voice commands: "Show me what I learned about React"
- Whisper integration for audio transcription
- Voice notes that auto-link to knowledge graph

#### Global Discovery

- Anonymized, global trending topics
- "1,000 developers are exploring Rust this week"
- Opt-in community learning paths

#### Collaborative Learning

- Share knowledge graphs with study groups
- Compare learning patterns with friends
- Collaborative annotation on shared topics

#### Developer Integrations

- VS Code extension (inline summaries)
- Obsidian plugin (two-way sync)
- Slack bot (query your knowledge base)
- Notion API integration

#### Mobile Experience

- iOS/Android apps (React Native)
- Quick capture: photo → OCR → add to graph

- Voice memos that link to topics
- Offline mode with sync on WiFi

## Advanced AI Features

- Spaced repetition learning mode
- Personalized quizzes from your knowledge
- Predictive learning path recommendations
- Automatic concept difficulty assessment

## Educational Partnerships

- University licensing programs
- Integration with Canvas/Moodle LMS
- Classroom collaboration features
- Student progress analytics for instructors

## 16.2 Technical Improvements

### Performance Optimization

- WebAssembly for local AI inference
- Edge computing for faster summarization
- Progressive Web App (PWA) support
- Offline-first architecture improvements

### AI Model Enhancements

- Fine-tune local models on user's domain
- Support for multimodal inputs (images, diagrams)
- Better context window management
- Lower latency with model caching

### Infrastructure Scaling

- Multi-region deployment (US, EU, Asia)
- CDN for static assets

- Database sharding for large users
  - Kubernetes Federation for global orchestration
- 

## 17. Competitive Analysis

### 17.1 Market Positioning

#### Direct Competitors

Product	Strength	Weakness	CurioAI Advantage
<b>Mem.ai</b>	AI-powered notes	Manual input required	Auto-capture + code tracking
<b>Notion AI</b>	Rich text editor	No knowledge graph	Visual graph + agentic insights
<b>Obsidian</b>	Local-first, plugins	No AI by default	Built-in AI + auto-linking
<b>Roam Research</b>	Bidirectional links	Expensive, manual	Automated + affordable
<b>MyMind</b>	Visual bookmarking	No text analysis	Deep semantic understanding

#### Indirect Competitors

- Browser bookmarking tools (lack AI)
- Note-taking apps (require manual effort)
- Read-it-later services (no knowledge connections)

**Blue Ocean Strategy** CurioAI creates a new category: **Autonomous Knowledge Intelligence**

- Not just storage (like Notion)
- Not just capture (like Pocket)
- Not just AI chat (like ChatGPT)
- **All three combined with agentic automation**

### 17.2 Unique Value Propositions

- 1. Only tool that auto-captures code activity**
  - Links tutorials watched → code written
  - Tracks technical learning journey
- 2. Truly local-first AI**
  - Runs Ollama for privacy-conscious users

- Optional cloud sync vs. forced cloud

### 3. Agentic insights without prompting

- AI proactively suggests learning paths
- Reminds you of related past learnings

### 4. Visual knowledge archaeology

- See how your understanding evolved over time
  - Discover forgotten connections
- 

## 18. Go-to-Market Strategy

### 18.1 Launch Strategy

#### Pre-Launch (Months 1-2)

- Build in public on Twitter/LinkedIn
- Create demo videos showcasing key features
- Beta waitlist landing page
- Engage with communities (r/productivity, Hacker News)

#### Beta Launch (Month 3)

- Invite 100 early adopters
- Daily feedback loops
- Feature prioritization based on usage data
- Build case studies from power users

#### Public Launch (Month 4)

- Product Hunt launch (aim for #1 Product of the Day)
- Post on Hacker News "Show HN"
- Reddit AMAs on r/productivity, r/artificial
- YouTube demo + tutorial videos

### 18.2 Marketing Channels

#### Organic (Primary Focus)

- **Content Marketing**

- Blog: "How I Remember Everything I Learn with AI"
- YouTube: Weekly knowledge graph showcases
- Newsletter: "The Curious Mind" (learning tips + product updates)

- **Community Building**

- Discord server for power users
- Monthly webinars: "Optimize Your Learning with CurioAI"
- User-generated content: Share your knowledge graphs

- **SEO Strategy**

- Keywords: "personal knowledge management AI", "automatic note-taking", "knowledge graph app"
- Backlinks from productivity blogs

## **Paid (Secondary)**

- Google Ads (target: "second brain app", "AI note-taking")
- Reddit Ads (r/productivity, r/getdisciplined)
- Sponsorships: Podcast ads on productivity/tech shows

## **Partnerships**

- Affiliate program: 30% commission for referrals
- Integration partnerships (Notion, Obsidian, VS Code)
- University partnerships for student licenses

## **18.3 Target Audience Segments**

### **Primary: Solo Knowledge Workers**

- Developers, researchers, writers
- Age: 25-45
- Tech-savvy, values privacy
- Pain: Information overload, poor retention

### **Secondary: Students**

- University/grad students

- Age: 18-30
- Heavy content consumers
- Pain: Exam prep, connecting concepts across courses

## Tertiary: Teams

- Startups, research labs
  - Need shared knowledge bases
  - Pain: Onboarding new members, tribal knowledge loss
- 

## 19. Technical Implementation Details

### 19.1 Desktop Agent - Detailed Flow

```
javascript
```

```

// Main Process (Electron)
const { app, BrowserWindow, Tray, Menu } = require('electron');
const activeWin = require('active-win');
const chokidar = require('chokidar');

// Activity Monitor Loop
setInterval(async () => {
  const window = await activeWin();

  if (isLearningActivity(window)) {
    const content = await extractContent(window);
    ipcRenderer.send('process-content', content);
  }
}, 60000); // Check every minute

// Content Processor (Python Microservice)
@app.post("/summarize")
async def summarize(content: str):
  prompt = f"Summarize this learning content:\n{content}"
  summary = await ollama.generate(model="mistral", prompt=prompt)

  # Extract concepts
  concepts = extract_entities(summary)

  # Generate embedding
  embedding = model.encode(summary)

  # Store locally
  db.execute("INSERT INTO summaries VALUES (?, ?, ?)",
            (summary, concepts, embedding))

  return {"summary": summary, "concepts": concepts}

```

## 19.2 Knowledge Graph Builder Algorithm

python

```

import networkx as nx
from sklearn.metrics.pairwise import cosine_similarity

class KnowledgeGraphBuilder:
    def __init__(self):
        self.graph = nx.DiGraph()

    def add_node(self, concept, embedding, metadata):
        self.graph.add_node(concept,
                            embedding=embedding,
                            **metadata)

    # Find related concepts
    def _create_edges(self, new_concept, new_embedding, threshold=0.7):
        for node in self.graph.nodes():
            if node == new_concept:
                continue

            existing_embedding = self.graph.nodes[node]['embedding']
            similarity = cosine_similarity(
                [new_embedding],
                [existing_embedding]
            )[0][0]

            if similarity > threshold:
                self.graph.add_edge(new_concept, node,
                                    weight=similarity,
                                    type="related_to")

    def get_learning_path(self, start_concept, goal_concept):
        try:
            path = nx.shortest_path(self.graph, start_concept, goal_concept)
            return path
        except nx.NetworkXNoPath:
            return None

```

## 19.3 RAG Pipeline for Chat

python

```
from langchain.vectorstores import Chroma
from langchain.chains import RetrievalQA
from langchain.llms import OpenAI

class CurioAIChatAgent:
    def __init__(self, user_id):
        self.vectorstore = Chroma(
            collection_name=f"user_{user_id}",
            embedding_function=OpenAIEmbeddings()
        )

        self.qa_chain = RetrievalQA.from_chain_type(
            llm=OpenAI(temperature=0),
            chain_type="stuff",
            retriever=self.vectorstore.as_retriever(
                search_kwargs={"k": 5}
            )
        )

    async def answer_query(self, query: str):
        # Retrieve relevant context
        docs = self.vectorstore.similarity_search(query, k=5)

        # Generate answer with citations
        response = self.qa_chain.run(query)

        # Add sources
        sources = [doc.metadata for doc in docs]

        return {
            "answer": response,
            "sources": sources,
            "confidence": self._calculate_confidence(docs)
        }

    def _calculate_confidence(self, docs):
        # Based on similarity scores
        avg_score = sum(doc.metadata['score'] for doc in docs) / len(docs)
        return min(avg_score * 100, 95) # Cap at 95%
```

## 20. User Stories & Use Cases

### 20.1 Developer Use Case: "John"

#### Profile

- Full-stack developer learning AI/ML
- Watches YouTube tutorials, reads documentation, codes daily
- Struggles to remember what he learned 2 months ago

#### Journey with CurioAI

1. **Day 1:** Installs desktop app, grants browser permission
2. **Week 1:** Watches 5 LangChain tutorials → CurioAI auto-summarizes each
3. **Week 2:** Reads LangChain docs while coding → Agent links tutorials to docs
4. **Week 3:** Builds RAG app → CurioAI connects "what he learned" to "what he built"
5. **Month 2:** Forgets LangChain syntax → Asks chat: "How do I use memory in LangChain?"
6. **Result:** Gets instant answer with links to his own notes from Week 2

#### Value Delivered

- Zero manual note-taking
- Instant recall of past learnings
- See evolution: Tutorial → Docs → Code

### 20.2 Student Use Case: "Sarah"

#### Profile

- Computer Science grad student
- Reads 10+ research papers per week
- Preparing for thesis on NLP

#### Journey with CurioAI

1. **Month 1:** Reads 40 papers on transformers, BERT, GPT
2. **CurioAI Action:** Auto-extracts key concepts, builds graph of paper relationships
3. **Month 2:** Discovers graph shows "Attention Mechanism" connects 30 papers

4. **Month 3:** Writing thesis → Asks: "What papers discussed attention in vision?"

5. **Result:** Gets 5 relevant papers with summaries he read months ago

## Value Delivered

- Literature review made visual
- Discover overlooked connections
- Thesis research became conversational

## 20.3 Team Use Case: "TechCorp Startup"

### Profile

- 10-person AI startup
- High employee turnover
- Knowledge loss when engineers leave

### Journey with CurioAI Enterprise

1. **Setup:** Each engineer uses CurioAI desktop + shared team graph (opt-in)

2. **Benefit 1:** New hire sees "What the team has been learning" graph

3. **Benefit 2:** Engineer leaves → Their knowledge graph stays (anonymized)

4. **Benefit 3:** Team chat: "Who learned about LangGraph?" → Finds 2 engineers

5. **Result:** 50% faster onboarding, zero knowledge loss

## Value Delivered

- Organizational memory
- Expertise discovery
- Collaborative learning culture

---

## 21. Compliance & Legal

### 21.1 Data Privacy Regulations

#### GDPR Compliance (EU)

- Right to access (export all data)

- Right to erasure (delete account + data)
- Data portability (JSON/CSV export)
- Consent management (explicit opt-in for sync)
- Privacy by design (local-first architecture)

## **CCPA Compliance (California)**

- Disclose data collection practices
- Allow users to opt-out of data sale (we don't sell data)
- Provide data deletion upon request

## **21.2 Terms of Service Highlights**

### **User Data Ownership**

- Users own 100% of their data
- CurioAI has license only to process for service delivery
- No training AI models on user data (unless explicit consent)

### **Acceptable Use**

- Prohibited: Illegal content, malware, harassment
- Monitoring: None (local-first design)
- Enforcement: Cloud account suspension if violated

### **Liability**

- No warranty for AI accuracy
- User responsible for backing up data
- Limitation of liability to subscription amount

## **21.3 Open Source Strategy**

### **What We Open Source**

- Desktop agent (MIT License)
- Local AI service (Apache 2.0)
- Web app frontend (MIT License)

- Knowledge graph algorithms (MIT License)

## What Remains Proprietary

- Backend microservices (IP protection)
- Agentic orchestration logic
- Recommendation algorithms
- Enterprise features

## Benefits

- Community contributions
- Transparency builds trust
- Developer adoption (GitHub stars)

## 22. Financial Projections (Year 1)

### 22.1 Revenue Forecast

Month	Free Users	Pro Users	Enterprise	MRR	Cumulative Revenue
1-3	100	0	0	\$0	\$0
4	500	25	0	\$225	\$225
5	800	40	0	\$360	\$585
6	1,200	60	0	\$540	\$1,125
7	2,000	100	1	\$1,150	\$2,275
8	3,500	175	1	\$1,725	\$4,000
9	5,000	250	2	\$2,650	\$6,650
10	7,000	350	3	\$3,800	\$10,450
11	9,000	450	4	\$5,050	\$15,500
12	10,000	500	5	\$5,500	\$21,000

## Assumptions

- 5% free → pro conversion rate
- Enterprise: \$250/month average
- Churn: 5% monthly

## 22.2 Cost Structure

Category	Monthly Cost (Month 12)	Annual Cost
<b>Infrastructure</b>	\$800	\$9,600
• Cloud hosting (AWS/GCP)	\$500	\$6,000
• Database (Neo4j Cloud)	\$200	\$2,400
• CDN (Cloudflare)	\$100	\$1,200
<b>AI/LLM Costs</b>	\$1,500	\$18,000
• OpenAI API	\$1,000	\$12,000
• Embedding API	\$500	\$6,000
<b>Software/Tools</b>	\$300	\$3,600
• GitHub, monitoring, etc.	\$300	\$3,600
<b>Marketing</b>	\$1,000	\$12,000
• Ads, content, tools	\$1,000	\$12,000
<b>Total Operating Costs</b>	\$3,600	\$43,200

## Gross Margin (Month 12)

- Revenue: \$5,500
- Costs: \$3,600
- **Margin: 34.5%** (improving as we scale)

## 22.3 Funding Strategy

### Bootstrap Phase (Months 1-6)

- Founder self-funded: \$20,000
- Focus: MVP development, beta testing
- No external funding needed

### Seed Round (Months 7-12) - Optional

- Raise: \$250,000
- Valuation: \$2.5M pre-money
- Use of funds:
  - Engineering team: \$150K (2 full-time devs)
  - Marketing: \$50K

- Infrastructure: \$30K
- Legal/compliance: \$20K

## Alternative: Revenue-Based Financing

- Borrow: \$100,000
  - Repay: 10% of monthly revenue until 1.5x repaid
  - No equity dilution
- 

# 23. Testing & Quality Assurance

## 23.1 Testing Strategy

### Unit Tests

- Backend services: 80%+ coverage (Pytest, Jest)
- Desktop agent: 70%+ coverage (Jest)
- Critical paths: 100% coverage (authentication, payments)

### Integration Tests

- API contract testing (Pact)
- Database migration tests
- Agent communication flows

### End-to-End Tests

- User flows (Playwright, Cypress)
- Desktop app scenarios (Spectron)
- Performance testing (K6)

### AI Model Testing

- Summarization quality (human eval)
- Graph relationship accuracy
- RAG response relevance (RAGAS metrics)

## 23.2 Quality Metrics

Metric	Target
Bug escape rate	<5%
Mean time to resolution	<24 hours
Test coverage	>75%
Build success rate	>95%
API error rate	<1%

## 24. Support & Documentation

### 24.1 User Support Tiers

#### Free Users

- Email support (48-hour response)
- Community forum (Discord)
- Knowledge base / FAQs
- Video tutorials

#### Pro Users

- Priority email (24-hour response)
- Live chat support (business hours)
- Monthly office hours webinar
- Early access to new features

#### Enterprise

- Dedicated Slack channel
- Phone support
- Custom onboarding sessions
- SLA: 4-hour response time

### 24.2 Documentation Structure

#### For Users

- Getting started guide (5 min quickstart)

- Feature deep-dives
- Video tutorials
- Troubleshooting guide
- Privacy & security FAQ

## For Developers

- API reference (OpenAPI spec)
  - SDK documentation (Python, JavaScript)
  - Webhook integration guide
  - Self-hosting instructions
  - Contributing guide (open source)
- 

## 25. Success Stories (Future)

### 25.1 Planned Case Studies

#### "How I Got My PhD 6 Months Early with CurioAI"

- Student: Sarah Martinez, MIT
- Result: Analyzed 200 papers, found novel connections
- Quote: "CurioAI was my AI research assistant"

#### "Our Startup's Secret Weapon: Knowledge Graphs"

- Company: TechFlow (Series A)
- Result: Reduced onboarding time from 3 months → 3 weeks
- Quote: "Every engineer's learning is now company knowledge"

#### "I Built 10 Side Projects by Never Forgetting What I Learn"

- Developer: John Chen, indie hacker
  - Result: Shipped faster by reusing past learnings
  - Quote: "My second brain actually works now"
-

## 26. Conclusion

CurioAI represents a paradigm shift in personal knowledge management:

### From Manual → Automatic

- No more note-taking burden
- Background intelligence capture

### From Storage → Understanding

- Not just saving, but connecting
- AI-powered insight generation

### From Individual → Collaborative

- Personal knowledge becomes team asset
- Organizational learning at scale

### Technical Excellence

- Local-first privacy architecture
- Agentic AI ecosystem
- Scalable microservices (K8s)
- Generative AI integration

### Business Viability

- Clear freemium monetization
- Growing market (\$2B+ PKM space)
- Defensible moat (user data network effects)
- Multiple revenue streams

### Next Immediate Steps

1. Week 1-2: Finalize tech stack decisions (Ollama vs GPT-4 for local)
2. Week 3-4: Build desktop activity monitor MVP
3. Week 5-6: Integrate local AI summarization
4. Week 7-8: Create basic web graph visualizer

## 27. Appendices

### A. Glossary

- **RAG:** Retrieval-Augmented Generation (AI technique for grounded responses)
- **Embedding:** Vector representation of text for semantic similarity
- **Knowledge Graph:** Network of interconnected concepts
- **Agentic AI:** Autonomous AI systems that take actions without human prompting
- **Local-first:** Software that works offline and syncs optionally

### B. References

- LangChain Documentation: <https://docs.langchain.com>
- Neo4j Graph Database: <https://neo4j.com>
- Ollama (Local LLMs): <https://ollama.ai>
- Electron Framework: <https://electronjs.org>
- ChromaDB: <https://www.trychroma.com>

### C. Contact Information

**Project Lead:** Rameswar Panda

**Email:** [your-email@example.com]

**GitHub:** [github.com/your-username/curioai]

**Website:** [curioai.dev]

**Discord:** [discord.gg/curioai]

---

### Document Version History

- v1.0 (Nov 8, 2025): Initial draft
- v2.0 (Nov 8, 2025): Added activity tracking details, technical implementation, financial projections

**Last Updated:** November 8, 2025

**Next Review:** December 1, 2025