Second Brain AI Companion – System Design

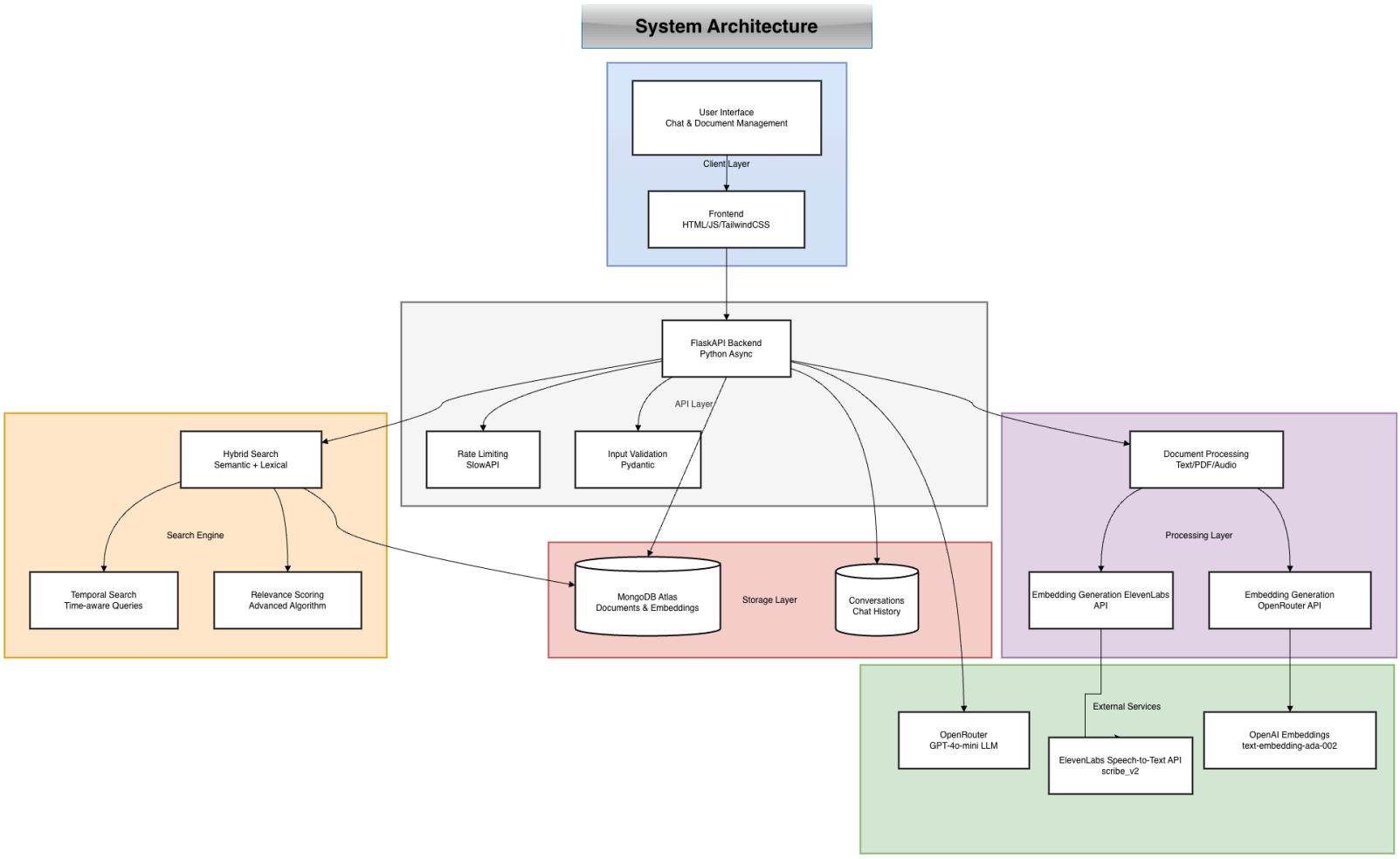
Table of Contents

Summary:

A personal knowledge management system that ingests multi-modal data (audio, documents, text) and provides intelligent conversational access through natural language queries. The system emphasizes scalability, operational simplicity, and intelligent information retrieval.
Live Application: https://ai-companion-proj.vercel.app

System Architecture:

High-Level Architecture - Frontend (HTML/JS, TailwindCSS) working with a Flask-based serverless API deployed on Vercel. The API integrates with OpenRouter for LLM responses (GPT-4o-mini) as well as using OpenAI's text-embedding-ada-002 ML model and ElevenLabs for speech-to-text transcription. A hybrid search engine processes and retrieves content stored in MongoDB Atlas, which serves as the system's primary cloud storage layer.
Text-embedding-ada-002 - converts text into high-dimensional vectors (works exceptionally well with semantic searching)
scribe_v2 - Whisper-based model, convert speech to text



Core Components:

Frontend - A single-page application supporting real-time chat, document upload, and knowledge exploration
API Layer - Flask-based serverless functions deployed on Vercel, responsible for ingestion, querying, and orchestration of AI services
Search Engine - A hybrid retrieval system combining semantic, lexical, and temporal signals for accurate information retrieval
Storage - MongoDB Atlas with flexible schema design and full-text indexing, optimized for global scalability
AI Services - OpenRouter for language model inference and ElevenLabs Scribe v2 for audio transcription

Multi-Modal Data Ingestion Pipeline:

The ingestion pipeline processes multiple data modalities through a unified, asynchronous workflow optimized for immediate availability and searchability.
Pipeline Flow:

- Input Sources → Validation → Processing → Storage → Indexing → Search Ready

Audio Processing:

- Supported formats include MP3, WAV, M4A, OGG, FLAC, AAC, AIFF, and AIF
- Processing steps include file validation (50MB limit), transcription via ElevenLabs Scribe v2, metadata extraction (format, size, estimated duration), and structured transcription output with source attribution
- Output consists of searchable text paired with transcription metadata

Document Processing:

- Supported formats include PDF, TXT, MD, DOC, and DOCX
- The pipeline performs file type detection, text extraction (UTF-8 decoding or OCR for PDFs), content normalization, and metadata preservation (filename, size, creation date)

Web Content Processing:

- User-provided URLs are validated for accessibility, scraped via HTTP requests or a

headless browser, parsed for text content, and cleaned to remove navigation, ads, and boilerplate

- Metadata such as title, description, and publication date is extracted and stored alongside the content
- Implementation uses BeautifulSoup and requests for static content, and Playwright for dynamic pages

Plain Text Processing:

- Raw text and markdown inputs undergo UTF-8 validation, basic formatting preservation, and content length checks before indexing

Image Processing:

- Images are stored in cloud storage (e.g., AWS S3) with multiple generated formats (thumbnail, web-optimized, original). EXIF metadata is extracted, including date, location, and device information
- Searchability is enabled through OCR text extraction, AI-generated visual descriptions using vision-language models, and indexing of EXIF metadata. Future enhancements include visual similarity search via image embeddings

Information Retrieval and Querying Strategy:

Hybrid Search Architecture:

- The system uses a multi-signal hybrid search strategy combining semantic similarity, lexical matching, and temporal relevance
- This approach avoids the limitations of single-method search systems by capturing context, exact phrasing, and time-based intent

Search Components:

- Semantic Search: Uses contextual word coverage and meaning-based similarity to surface conceptually related content
- Lexical Search: Leverages MongoDB full-text search with TF-IDF scoring for exact phrase and keyword matching
- Temporal Search: Incorporates natural language date parsing and time-based proximity scoring to support time-aware queries

Fusion Scoring Algorithm:

- Search results are ranked using a multi-stage scoring process that prioritizes exact phrase matches, evaluates word coverage, applies semantic relevance boosts, incorporates temporal proximity, and weights content types such as titles versus body text
- Final results are normalized into confidence scores from 0 to 100 percent

Data Indexing and Storage Model:

Data Lifecycle:

- Raw data progresses through ingestion, processing, chunking, indexing, storage, and retrieval
- Text documents are split into overlapping semantic chunks of approximately 500 words. Audio transcriptions are stored as single documents with timestamp markers. Web content is chunked at the article or section level. Images store metadata and extracted text as searchable content

Database Schema Design:

- Documents Collection: Each document includes a unique identifier, title, full content, source type (audio, PDF, text, web, or image), source path or URL, ingestion timestamp, original creation timestamp when available, file size, processing status, and type-specific metadata. Large documents include an array of content chunks with positional information
- Indexing Strategy: MongoDB full-text indexes are applied to titles and content. Temporal indexes support time-based queries, compound indexes enable filtered searches by content type and date, and metadata fields are indexed for advanced filtering
- Storage Trade-offs: MongoDB Atlas was selected for its schema flexibility, built-in search capabilities, managed scaling, global distribution, and cost efficiency compared to multi-database architectures or dedicated vector databases

Temporal Querying Support:

- All ingested content includes ingestion time, original creation time when available, last

modified timestamps, and access timestamps for analytics

- Natural language temporal expressions such as “last week” or “before the deadline” are parsed using pattern recognition and contextual resolution, converted into absolute date ranges, and integrated into query filters
- Temporal relevance scoring applies boosts for exact range matches, decay for near-miss results, and slight recency bias for ambiguous queries

Key Architectural Decisions:

Serverless-First Architecture:

- ElevenLabs Scribe v2 was chosen over local Whisper deployment to maintain serverless compatibility, ensure reliability, and avoid large model dependencies
- MongoDB Atlas Over Specialized Vector Databases
- A unified storage system was preferred to reduce operational complexity while still supporting full-text search, metadata storage, and future vector embeddings

Hybrid Search Strategy:

- Combining lexical, semantic, and temporal signals provides higher retrieval accuracy, transparency through confidence scores, and flexibility across query types

Scalability and Performance:

The current system performance averages approximately 1.2 seconds per query with over 90 percent relevance on test queries. The system scales automatically via serverless infrastructure and MongoDB Atlas. Future scaling includes caching, query optimization, and CDN usage for static assets. Primary bottlenecks stem from external API latency, mitigated through asynchronous processing and optimization.

Security and Privacy:

Secrets are stored using environment variables. Input validation enforces file size and type restrictions. CORS policies restrict access, and error handling prevents information leakage.

Users maintain control over their data and API keys. The system clearly documents external service usage and balances functionality with privacy trade-offs.

Technology Stack Justification:

Frontend - Vanilla JavaScript and HTML provide simplicity, performance, and ease of deployment.
Backend - Flask on Vercel enables fast cold starts, serverless compatibility, and access to the Python AI ecosystem.
Database - MongoDB Atlas offers flexible schemas, built-in search, global scalability, and minimal operational overhead.

Future Enhancements:

Short-term improvements include vector embeddings, analytics, multi-user support, and enhanced privacy controls. Long-term plans involve enterprise collaboration, proactive AI insights, mobile applications, browser extensions, and third-party integrations.

Success Metrics:

Technical goals include sub-2-second response times for 95 percent of queries, over 99 percent availability, and linear scalability. User experience success is measured through high query usefulness, feature adoption, and minimal error rates.

Conclusion:

This system balances scalability, simplicity, and intelligent retrieval by leveraging serverless infrastructure and managed services. The hybrid search engine and multi-modal ingestion pipeline enable powerful personal knowledge management while remaining extensible for future enterprise use. The design reflects real-world engineering trade-offs that prioritize reliability, clarity, and user value.