Memoir.ai — Technical Architecture Manual

System Design, Component Boundaries, and Execution Model

Document Version: 1.0

Status: Engineering Reference

Owner: Platform Architecture Team

Last Updated: YYYY-MM-DD

----

1. Purpose

This Technical Architecture Manual defines the structural, logical, and operational architecture of Memoir.ai. It provides engineers and system operators with a comprehensive understanding of system components, data movement, processing boundaries, and execution flows required to build, maintain, and evolve the platform.

----

2. Scope

This manual covers:

• System architecture and component boundaries

• Multi-process execution model

• Data ingestion and processing flows

• AI snapshot generation architecture

• Storage and encryption model

• Background job execution

• Search and indexing architecture

• Performance strategies and budgets

• Export and data portability model

• Cloud metadata synchronization layer

• API and IPC communication model

UI design and end-user workflows are out of scope.

----

3. Architectural Philosophy

Memoir.ai is designed as a local-first system prioritizing:

• User data ownership

• On-device processing

• Encrypted storage by default

• Deterministic processing pipelines

• Verifiable AI outputs

• Minimal cloud dependency

Cloud services manage metadata, billing, and sync state only. Personal content remains local.

----

4. System Overview

Memoir.ai operates as a desktop application using Electron and React, with a local backend running inside the Electron environment. Heavy computation executes in background workers to keep the interface responsive.

Primary components:

• Electron main process

• Renderer UI process

• Background worker processes

• Encrypted vault storage

• Job runner and queue manager

• AI narrative engine

• Search and indexing services

• Optional cloud metadata layer

----

5. Multi-Process Execution Model

Main Process:

• Controls application lifecycle

• Manages IPC communication

• Handles filesystem access

Renderer Process:

• Runs React UI

• Displays timeline and narratives

• Communicates via IPC bridge

Worker Processes:

• Perform imports

• Execute AI generation

• Build indexes

• Process media

Workers run at lower priority to preserve UI responsiveness.

----

6. Storage Architecture

Vault Partition:

• SQLCipher encrypted database

• Media storage directories

• Thumbnails and caches

Config Partition:

• Non-sensitive settings

• UI preferences

• Window configuration

Encryption keys exist only in memory while unlocked.

----

7. Data Ingestion Architecture

Import flow stages:

1. User selects archive.
2. Extraction worker parses raw data.
3. Validator checks schema integrity.
4. Normalization converts records to canonical format.
5. Events stored in encrypted database.
6. Errors logged and surfaced.

Ingestion runs via resumable background jobs.

----

8. Background Job Architecture

Jobs execute through a SQLite-backed queue.

Lifecycle:

1. Enqueue
2. Worker dispatch
3. Active execution
4. Completion or failure

Features:

• Priority scheduling

• Retry and checkpointing

• Crash recovery

• Memory throttling

----

9. AI Snapshot Generation Architecture

Narrative pipeline:

1. Evidence sampling from database.
2. Prompt context construction.
3. Local model inference.
4. Citation mapping.
5. Verification checks.
6. Narrative storage with version metadata.

All outputs must map to source evidence.

----

10. Provenance & Versioning

Each event and narrative stores provenance metadata.

Versioning rules:

• Regeneration creates new versions.

• User edits preserved.

• Published versions remain immutable.

• Version browsing supported.

Citation integrity maintained across versions.

----

11. Search & Indexing Architecture

Hybrid search combines:

• SQLite FTS keyword search

• Vector-based semantic search

Workflow:

1. Events normalized.
2. Indices built in workers.
3. Queries combine lexical and semantic scores.

All indices remain encrypted inside vault.

----

12. Performance Architecture

Performance optimizations include:

• Chunked ingestion streams

• Worker thread throttling

• Database page caching

• Prepared SQL statements reuse

• Thumbnail caching

Targets include:

• Search <250ms

• Snapshot generation <8s

• Smooth timeline scrolling

----

13. Export & Data Portability

Exports include:

• Events

• Narratives

• Participants

• Media

• Provenance metadata

Data packaged in structured JSON and organized ZIP bundles. Media indexed by hashes for reconstruction.

Exports work offline without vendor dependency.

----

14. Cloud Metadata Layer

Cloud services manage:

• Subscription entitlements

• Device/workspace metadata

• Job sync health

No personal content stored remotely.

Row-level security ensures tenant isolation.

----

15. Communication Architecture

Communication channels include:

IPC Bridge:

• Secure renderer-to-backend communication

Local REST APIs:

• Non-sensitive metadata operations

Webhooks:

• Stripe billing updates

• Auth workspace creation events

All sensitive operations require vault unlock state.

----

16. Security Boundaries

Security controls include:

• Process isolation

• Encrypted vault storage

• Input validation layers

• Sanitized IPC messaging

• Memory key clearing

Renderer processes have no direct filesystem access.

----

17. Failure Recovery Architecture

System supports:

• Import resumption

• Job retries

• Database integrity repair

• Backup restoration

Failures generate logs and user recovery guidance.

----

18. Operational Best Practices

Recommended practices:

• Maintain backward compatibility in schemas.

• Monitor worker memory usage.

• Test ingestion on large archives.

• Validate search performance across releases.

• Audit encryption and key handling regularly.

----

19. Conclusion

The Memoir.ai architecture balances performance, privacy, and verifiability while supporting scalable ingestion and AI-assisted narrative generation. This manual provides engineering teams with a structural reference for platform development and operational reliability.