

AI Research Assistant (AIRA): Search, Summarize and Cite

CSCI 6180 Software Design and Development

Final Project - Requirement Gathering and Analysis

Team Members:

Ahmeed Yinusa

Shang Chen

Dimitri Nanmejo

October 4, 2025



Figure 1: AIRA Logo

Contents

AI Disclaimer	4
1 Scope Statement	5
1.1 Finding Papers (Discovery)	5
1.2 Writing with AI (Essay / Notes Generation)	5
1.3 Chatting with PDFs (Knowledge Extraction)	5
2 Stakeholders / Users	5
2.1 Primary Users	5
2.2 Secondary Users	5
2.3 Project Stakeholders	5
2.4 Influencers & Constraints	6
3 Functional Requirements	6
3.1 Authentication & Profiles	6
3.2 Paper Discovery (Semantic Scholar)	6
3.3 General Chat & Summarization	6
3.4 PDF Chat (Parse → Index → RAG)	6
3.5 AI Writer Workspace & History	7
3.6 Admin / Operations	7
4 Non-Functional Requirements (NFR)	7
4.1 Performance	7
4.2 Scalability	8
4.3 Security & Privacy	8
4.4 Reliability & Availability	8
4.5 Usability & Accessibility	8
4.6 Maintainability & Observability	8
4.7 Portability & Cost	8
5 Risks & Mitigations	8
6 Data Entities & Attributes	9
6.1 Core Entities	9
6.2 Inputs / Outputs	9
7 User Stories	9
7.1 Example User Workflow	10
7.1.1 Step 1: User Login / Onboarding	10
7.1.2 Step 2: Search & Save Papers	10
7.1.3 Step 3: Batch Abstract Summaries	11
7.1.4 Step 4: General Chat	11
7.1.5 Step 5: PDF Chat with Citations	11
7.1.6 Step 6: Insert Evidence in Writing	11
7.1.7 Step 7: Provider Switching	11
7.1.8 Step 8: Delete My Data	12
7.1.9 Step 9: Logout	12

8 System Architectural	12
9 Acceptance Criteria	13
10 Conclusion	14

AI Disclaimer

Work for this assignment was completed with the aid of AI tools and further edited manually to meet assessment requirements.

1 Scope Statement

The project, **AIRA**, is a web-based research assistant designed mainly for students and researchers. Its purpose is to make three things easier:

1.1 Finding Papers (Discovery)

Instead of searching multiple websites, AIRA connects to scholarly databases like **OpenAlex**, **Crossref**, and **Semantic Scholar**. Users can type in a topic or keywords, and the system shows a clean list of results including titles, authors, abstracts, and links. This ensures access to relevant academic material, even if one source is down or rate-limited.

1.2 Writing with AI (Essay / Notes Generation)

Once references are gathered, users can leverage AIRA’s AI assistant to generate structured academic writing. The assistant creates sections such as *Introduction*, *Applications*, *Limitations*, *Future Work*, and *Conclusion*. The generated content includes proper in-text citations in Author–Year style (e.g., “LeCun, 2015”), with all references clickable in a side panel. The AI provider is **pluggable**: it defaults to DeepSeek but can be switched seamlessly to OpenAI, Anthropic Claude, Gemini, or even open-source LLMs without redesigning the system.

1.3 Chatting with PDFs (Knowledge Extraction)

Users can upload research papers in PDF format, which the system parses into text, splits into chunks, and embeds into a vector database (**Supabase + pgvector**). This enables interactive Q&A with the document: for example, a user may ask, “What methodology did the authors use?” and AIRA will retrieve the relevant chunks, summarize them, and return an answer with citations pointing back to the exact section of the paper.

2 Stakeholders / Users

2.1 Primary Users

Undergraduate and graduate students, research assistants, and faculty. They rely on AIRA to search academic databases, generate structured notes, and support research or teaching tasks.

2.2 Secondary Users

Industry professionals and independent learners. They use the platform to quickly extract reliable, citable insights from scholarly sources without deep manual research.

2.3 Project Stakeholders

Instructor/Teaching Assistant (TA), development team, hosting providers, API providers (Semantic Scholar, OpenRouter), and Supabase. Each plays a role in evaluation, delivery, uptime, or supplying essential data and AI infrastructure.

2.4 Influencers & Constraints

Academic integrity policies, API limits, free-tier restrictions, and privacy requirements. These factors shape system design, performance, scalability, and compliance.

3 Functional Requirements

3.1 Authentication & Profiles

- Users can register and log in securely using Supabase Auth with email/password.
- Each user has a profile storing key details like ID, email, display name, and preferences (e.g., preferred AI provider/model), enabling a personalized experience.

3.2 Paper Discovery (Semantic Scholar)

- Users can search Semantic Scholar with filters (keywords, authors, year, venue) and view detailed paper results.
- Selected papers can be saved to a personal library for later reference.
- The system can automatically generate short summaries of multiple abstracts to help users quickly assess a batch of results.

3.3 General Chat & Summarization

- Users can ask open-ended research or study questions, with AI providing contextual answers.
- Uploaded text (e.g., notes or abstracts) can be condensed into bullet summaries and a one-line key takeaway.
- The AI backend is configurable, defaulting to DeepSeek via OpenRouter, but allowing switching to providers such as OpenAI, Anthropic, Gemini, or open-source LLMs.

3.4 PDF Chat (Parse → Index → RAG)

- Users upload PDFs, which are stored in Supabase Storage and parsed using tools like LlamaParse or Unstructured.
- The parsed text is chunked (with overlap), embedded using MiniLM, and stored in a vector database (pgvector).
- On query, the system retrieves the most relevant chunks, provides inline citations, and allows viewing source snippets and metadata for accurate RAG-style Q&A.

3.5 AI Writer Workspace & History

- A dedicated writer workspace includes Notes, Editor, and AI Assistant tabs; the assistant can insert evidence directly into drafts with sources.
- User activity (chat history, saved searches, saved papers, indexed documents) is persisted for continuity.
- Users can manage their data by deleting papers, documents, or chats to maintain control over their workspace.

3.6 Admin / Operations

- The system includes health checks and structured error logging for all gateway and API calls. This allows administrators and developers to quickly identify issues, monitor system performance, and ensure reliability during peak usage or demo sessions.
- Per-user and per-session rate limiting is enforced to prevent misuse, accidental overloading, or exceeding API quotas. This also protects backend resources, ensures fair usage among all users, and maintains system stability under concurrent access.
- Basic monitoring dashboards provide real-time metrics such as active users, request counts, and error rates, helping the team respond proactively to potential issues.
- Logging and metrics are designed to avoid storing sensitive user data (PII), maintaining compliance with privacy and security policies.

4 Non-Functional Requirements (NFR)

Our project's Non-Functional Requirements (NFRs) describe aspects that do not relate to specific behaviors or features but characterize how it performs certain functions. These requirements focus on project features such as performance, usability, security, reliability, scalability, and maintainability. For this project, each NFR is detailed in terms of measurable criteria (*e.g., response times, availability, accessibility*) and includes a description of our team's specific approach to achieve it, ensuring that the requirements are testable, measurable, and aligned with project goals.

4.1 Performance

NFR-1: Median search response should be \leq 2 seconds and chat/summarize operations \leq 20 seconds for 95% of requests, allowing outliers to exceed these limits without breaking the system. Achieved through caching frequently accessed results, connection pooling, concise AI prompts, and timeouts/retries for long-running requests. This ensures fast, responsive interactions for nearly all users.

4.2 Scalability

NFR-2: The system should reliably support a defined cohort of up to 50 simultaneous users in a classroom or lab setting. While multiple institutions and thousands of concurrent users are out of current scope, the system is designed with a stateless Node.js gateway and horizontally scalable hosting to facilitate future expansion. Supabase Postgres provides efficient, growth-ready storage.

4.3 Security & Privacy

NFR-3: API/provider keys are never exposed in the browser; all secrets are stored securely in the server or Supabase. NFR-4: Row-Level Security (RLS) ensures each user can only access their own data. NFR-5: Users can fully delete their content, including uploaded documents and vector embeddings, maintaining compliance with privacy expectations.

4.4 Reliability & Availability

NFR-6: Target 99% availability during demo or peak usage. Temporary offline behavior assumes cached responses and local summaries where feasible; full fallback with AI responses is only supported if the secondary provider is online. Health monitoring, fallback provider switching, and circuit breakers enhance reliability.

4.5 Usability & Accessibility

NFR-7: Keyboard navigation and ARIA attributes are implemented for accessibility. Chat streams use `role="log"` and `aria-live="polite"` to provide real-time updates to screen readers, making the system usable for visually impaired users.

4.6 Maintainability & Observability

NFR-8: Modular, well-structured code with linting and formatting allows easier maintenance and future feature expansion. NFR-9: Structured logging captures errors and provider responses (excluding PII). A minimal metrics dashboard monitors system health and performance, supporting observability.

4.7 Portability & Cost

NFR-10: Provider-agnostic gateway allows switching between AI providers. Cost is minimized through default provider selection, request batching, caching repeated queries, and careful monitoring of hosting and API usage. Dependencies: Functionality assumes active internet for AI provider calls; local fallbacks (*cached data/summaries*) are limited and may be partial.

5 Risks & Mitigations

- **API Rate Limits:** Calls to OpenAlex, Crossref, or Semantic Scholar may be throttled. *Mitigation:* Multi-source fallback, caching, and batch queries.

- **LLM Hallucinations:** AI may invent citations or inaccurate facts. *Mitigation:* Author-Year enforcement, side-panel references, and provider switching.
- **PDF Parsing Errors:** Poorly formatted PDFs may yield broken chunks. *Mitigation:* Overlapping chunking, alternative parsers (LlamaParse/Unstructured).
- **User Privacy:** Personal data leakage or unintended sharing. *Mitigation:* Supabase Row-Level Security (RLS), “delete my data” endpoint, and secrets on server only.

6 Data Entities & Attributes

6.1 Core Entities

- **User / Profile:** Stores user information including unique ID (*uuid*), email, display_name, creation timestamp, and settings such as preferred AI provider or model. This entity enables personalization and secure authentication.
- **Paper:** Represents research papers retrieved from Semantic Scholar or other sources. Attributes include unique ID (*uuid*), associated user ID, paper ID from source (*s2_paper_id*), title, authors, venue, year, abstract, URL, and creation timestamp. Supports saving and batch summarization.
- **Document (uploaded):** Stores PDFs uploaded by users. Attributes include unique ID (*uuid*), associated user ID, title, file URL, and creation timestamp. This entity is used in the RAG pipeline for parsing and embedding.
- **Chunk:** Represents a segment of parsed document text, used for retrieval in RAG. Attributes include unique ID (*bigserial*), parent document ID, chunk index, content, and embedding vector (384-dim) for similarity search.
- **Chat Message:** Records interactions between the user and AI assistant. Attributes include message ID, user ID, role (*user, assistant, or system*), message content, and creation timestamp. Supports persistence and historical context for AI responses.

6.2 Inputs / Outputs

- **Inputs:** Includes email/password for authentication, search queries for paper discovery, PDF uploads for RAG, free-text prompts for chat, and provider/model selection for AI calls.
- **Outputs:** Includes search results with metadata, summarized bullet points and one-line takeaways, AI-generated chat answers with inline citations, and saved items in user libraries or workspaces.

7 User Stories

1. **Search & Save:** Users can search scholarly databases with keywords or filters and save selected papers to their personal library for later reference, enabling efficient organization of research material.

2. **Batch Abstract Summaries:** Users can generate concise bullet-point summaries and a one-line takeaway for multiple papers at once, helping them quickly assess relevance without reading each abstract in full.
3. **General Chat:** Users can ask open-ended questions related to research topics or study material, and the AI provides contextual answers within approximately 20 seconds for 95% of queries, facilitating faster understanding.
4. **PDF Chat with Citations:** Users upload PDFs of research papers, ask targeted questions, and receive answers with inline [#n] citations. A snippet viewer allows users to verify source context, supporting accurate knowledge extraction.
5. **Insert Evidence in Writer:** The AI assistant can automatically insert grounded paragraphs with proper citations directly into the user's writing workspace, improving writing efficiency and accuracy.
6. **Provider Switching:** Users can change the AI provider or model in settings, and all subsequent AI calls use the selected provider, giving flexibility to compare outputs or optimize cost/performance.
7. **Delete My Data:** Users can delete papers, uploaded documents, chunked embeddings, and chat history from both storage and the database, ensuring control over personal data and compliance with privacy requirements.

7.1 Example User Workflow

7.1.1 Step 1: User Login / Onboarding

1. User navigates to the login page.
2. Options: Email/password login or Google/SSO login.
3. On successful login, the user sees the dashboard displaying:
 - Saved papers
 - Recent activity
 - Quick access to AI tools (Search, Chat, PDF Upload)

7.1.2 Step 2: Search & Save Papers

1. User enters keywords or selects filters (year, author, venue) in the search bar.
2. System queries the scholarly database (e.g., Semantic Scholar API).
3. Results are displayed with titles, authors, abstracts, and *Save* buttons.
4. User selects papers of interest and clicks *Save to Library*.
5. Saved papers appear in the personal library with options to annotate, tag, or delete.

7.1.3 Step 3: Batch Abstract Summaries

1. User selects multiple papers from the library.
2. Clicks *Generate Summaries*.
3. System produces:
 - Bullet-point summaries for each paper
 - A concise one-line takeaway per paper
4. User quickly identifies relevant papers without reading full abstracts.

7.1.4 Step 4: General Chat

1. User opens the *AI Chat* interface.
2. User types open-ended questions, e.g., “What are the latest methods in brain tumor detection?”
3. AI returns contextual answers within approximately 20 seconds for 95% of queries.
4. User may refine questions or ask follow-ups.

7.1.5 Step 5: PDF Chat with Citations

1. User uploads a research PDF.
2. User asks targeted questions, e.g., “What methodology was used in this study?”
3. AI extracts relevant text and provides:
 - Inline citations [#1], [#2] referencing pages/sections
 - Option to view text snippets for verification

7.1.6 Step 6: Insert Evidence in Writing

1. User opens the *Writer/Editor* interface.
2. User requests AI to insert grounded evidence.
3. AI generates a paragraph with proper citations directly in the document.
4. User can edit or accept the insertion.

7.1.7 Step 7: Provider Switching

1. User navigates to *Settings* → *AI Provider*.
2. Selects a different AI provider/model.
3. All subsequent AI calls use the new provider.
4. User can compare outputs or optimize for cost/performance.

7.1.8 Step 8: Delete My Data

1. User navigates to *Settings* → *Privacy* → *Delete Data*.
2. Selects data to delete:
 - Saved papers
 - Uploaded PDFs
 - Chunked embeddings
 - Chat history
3. System permanently deletes selected data from storage and database.
4. User receives confirmation of deletion.

7.1.9 Step 9: Logout

1. User logs out.
2. All sessions are terminated to ensure privacy and security.

8 System Architectural

The system architecture is organized into several modular layers (see Figure 2). The **frontend** is developed using React, Vite, and Tailwind CSS, providing an efficient and responsive interface with key pages such as Home, Dashboard, AI Writer, and Recent activities, alongside reusable components like ChatArea and ResearchPanel. The **gateway** layer, built with Node.js and Express, exposes core endpoints (`/api/search`, `/api/chat`, `/api/summarize`, `/api/parse`, `/api/index`, `/api/rag_chat`) to manage communication between the frontend, backend services, and AI models. Data persistence is handled by **Supabase**, which integrates Postgres for relational storage, authentication services, file storage, and pgvector for embedding-based similarity search. The **AI provider layer** is designed to be flexible, defaulting to DeepSeek via OpenRouter while supporting seamless switching to OpenAI, Anthropic, Gemini, or open-source LLMs. Finally, the **RAG pipeline** leverages MiniLM embeddings with 384 dimensions, cosine similarity, and top- k retrieval to provide context-grounded responses with inline citation markers for traceability.

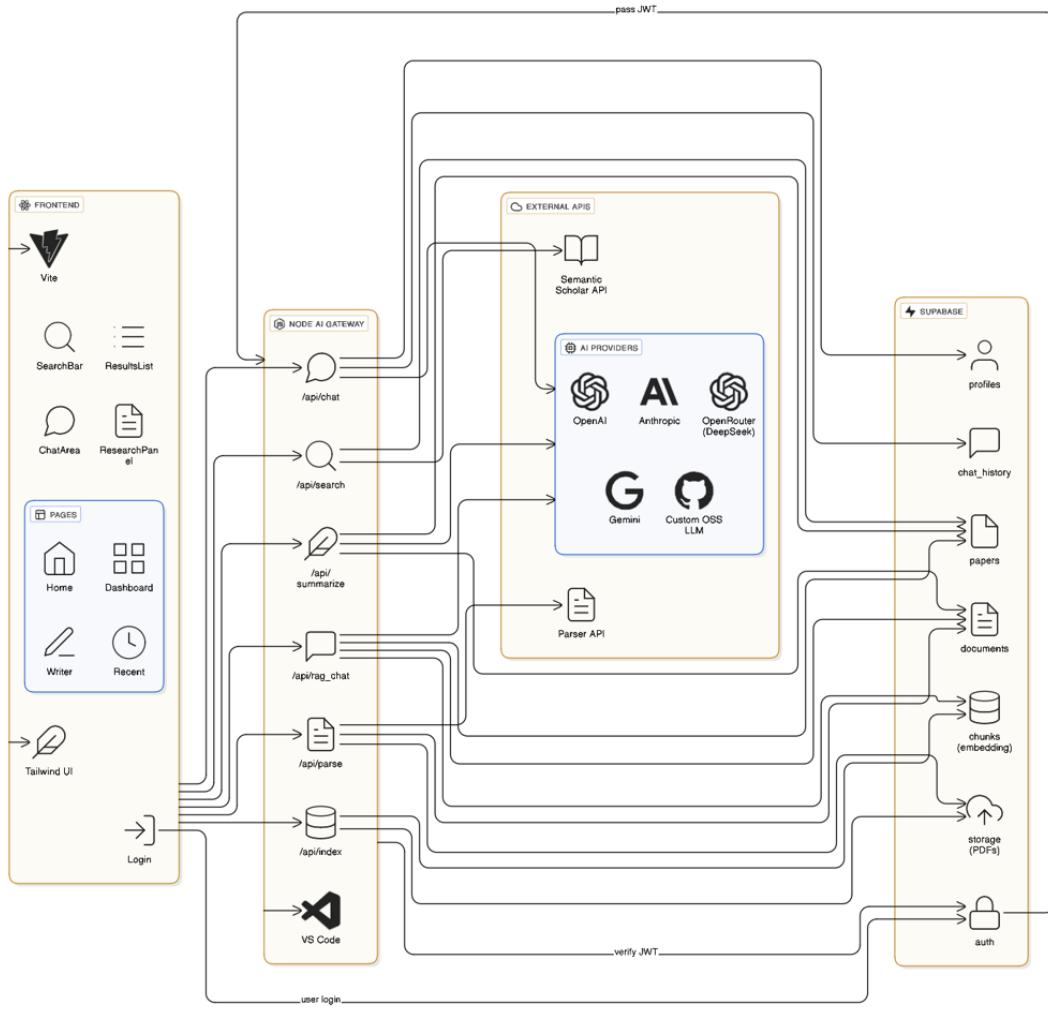


Figure 2: High-level system architecture (frontend, gateway, DB, providers). Place `architecture.png` in project folder.

9 Acceptance Criteria

- **Discovery:** A keyword query returns at least 5 relevant papers with title, authors, year, and link within 3 seconds (median).
- **AI Writer:** Generates a five-section essay (Introduction, Applications, Limitations, Future Work, Conclusion) with Author–Year citations that are clickable and match the References panel.
- **PDF Chat:** Uploading a PDF enables user queries that return grounded answers with at least one citation and snippet source.
- **Authentication:** Users cannot access or delete data belonging to other users (verified via Supabase RLS).
- **Performance:** 95% of summarization/chat requests complete within 20 seconds.

10 Conclusion

This document has comprehensively captured the requirements and design considerations for the AI Research Assistant (**AIRA**). It details the project scope, primary and secondary users, functional requirements, non-functional requirements (NFRs), core data entities, user stories, and architectural notes. By systematically defining functional and non-functional requirements, we ensure that AIRA is not only feature-rich but also performant, scalable, secure, and maintainable. The user stories illustrate practical scenarios that guide development, while the data entities and architecture provide a clear blueprint for implementation. Together, these requirements and architectural guidelines establish a foundation for a robust, user-centric system that supports efficient academic research, AI-assisted writing, and interactive knowledge extraction from PDFs. The framework is designed to accommodate future expansions, alternative AI providers, and continued improvements, ensuring that AIRA remains a flexible and reliable tool for students, researchers, and other users.