6. Technology Stack & Justification

This section describes all major technologies selected for ReadStudyAssist AI, along with clear justifications for why each technology was chosen and how it supports the system architecture defined in Section 5.

6.1 Backend Technologies

Gemini (Google Generative AI)

Purpose:
Provides summarization, context-aware Q&A, and quiz generation capabilities.

Justification:

- Strong performance for text summarization and document understanding tasks.
- Reliable function calling support simplifies structured tasks such as quiz generation.
- Cost-effective and available with generous free-tier limits—important for academic projects.

Local File Storage

Purpose:
Stores uploaded raw PDFs and extracted text files.

Justification:

- Minimal overhead and ideal for a local server architecture.
- No operational costs compared to cloud storage.
- Simple access patterns: read/write operations directly from backend.
- Matches the scope of the app while still allowing easy migration to cloud storage later.

Python Language

Purpose:
Python serves as the primary backend language for implementing the document ingestion, LLM orchestration and output formatting.

Justification:

- Python has strong ecosystem support for text-processing and AI workloads.
- Python integrates natively with Gemini.
- The team's familiarity with Python reduces development time and minimizes technical risk.

6.2 Frontend Technologies

Streamlit

Purpose:
Streamlit provides the user interface where learners upload documents, ask questions, view summaries, and access quizzes.

Justification:

- Extremely fast to prototype and adapt, which is important within the short capstone timeline.
- Simplified state management and built-in widgets make file upload and text display trivial.
- Avoids the complexity of a full React frontend while still offering a polished UI.
- Plays well with Python, enabling tight integration without writing a separate frontend codebase.

6.3 AI Capabilities

Prompt Templates & Function Calling

Purpose:
Standardize interactions with the LLM for summarization, Q&A, and quiz generation.

Justification:

- Ensures predictable outputs and consistent quality across use cases.
- Reduces prompt engineering overhead and improves maintainability.
- Function calling helps enforce structured responses (e.g., quiz JSON format), reducing post-processing errors.

Document Processing Libraries

Tools: To be decided
Purpose:
Extracts text from user-uploaded PDF files.

6.4 Infrastructure

Metadata Logging with Langfuse

Purpose:
Tracks LLM calls, response latency, errors, and user interactions.

Justification:

- Provides observability and debugging insights during development.
- Helps measure LLM performance across summarization, Q&A, and quiz generation.

Local Deployment

Purpose:
Run the backend and UI on a single machine.

Justification:

- Minimal setup required, reducing deployment complexity.
- Suitable for development and demonstration use cases.
- No need for container orchestration or cloud computer given project scope.

Version Control – GitHub

Purpose:
Source control, issue tracking, and team collaboration.

Justification:

- Standard for collaborative academic and industry projects.
- Easy CI/CD integration if needed later.
- Keeps code, diagrams, and documentation consistent across team members.

6.5 Alternative Technologies Considered

Category	Alternatives	Reason Not Selected
Frontend	React, Vue	No experience working with them; longer development time
LLM Providers	GPT-4, Claude	Higher cost and/or more limited free-tier access
Storage	Any Cloud Storage	Not required for the app; increases complexity