Project Requirements

Scientific Paper Analyzer: Cloud-Deployed Summarization and Q&A Tool

Team Members:

Ali Talasaz, Evan Parra, Bharath Sai Vallabhaneni, Ashritha Aloori

Introduction

The requirement analysis for the Scientific Paper Analyzer Bot outlines a clear framework for building an intelligent, interactive system that can simplify the process of working with academic literature. The system is designed to automatically summarize research papers, extract key insights, and enable users to engage with documents through natural-language queries. By combining document ingestion, summarization, retrieval, and conversational Q&A into one platform, the bot offers an efficient way for students, researchers, and faculty to interact with complex content.

Functionally, the system supports ingestion of documents in PDF format, generates running summaries, highlights methodologies and results, and provides question-answering grounded in the document's content. Retrieval-Augmented Generation (RAG) mechanisms ensure context-aware responses supported by citations and conversational memory.

On the non-functional side, the system prioritizes speed, scalability, and accuracy, ensuring quick responses, support for many simultaneous users, and answers that remain grounded in the document. Usability is also key, with an interface designed for both technical and non-technical users. Beyond this, features such as session persistence, structured formatting, and secure logging enhance reliability and user trust.

Collectively, these requirements provide a robust foundation for a cloud-deployed conversational AI tool optimized for academic research assistance. The sections below outline more specific requirements, covering both user and system perspectives.

User Requirements

Requirements were gathered from user role definitions below and proposal documentation, then prioritized using the MoSCoW framework.

User Roles

- **Students / Researchers** – upload papers, request summaries, ask Q&A.
- **Faculty / Reviewers** – validate document insights, use for teaching or peer-review support.
- **Developers / System Maintainers** – ensure deployment, updates, and reliability.

Requirements by Role

- **Students / Researchers**
  - **UR-1.1 (Must):** Allow users to upload academic papers in PDF format via a simple web interface.
  - **UR-1.2 (Must):** Automatically generate concise summaries and highlight 3–5 key findings or results.
  - **UR-1.3 (Must):** Support natural-language questions with accurate, citation-backed answers from the document.
  - **UR-1.4 (Should):** Maintain conversational memory for context-aware Q&A across sessions.
  - **UR-1.5 (Could):** Provide open research questions or future work suggestions from the paper.
  - **UR-1.6 (Won't):** Support non-PDF formats and multi-user collaboration, deferred for future versions.
- **Faculty / Reviewers**
  - **UR-2.1 (Must):** Validate summaries and answers for correctness and relevance.
  - **UR-2.2 (Should):** Use the tool to quickly identify key methodologies, results, and insights from the paper.
  - **UR-2.3 (Could):** Save interactions and analyses for later reference or review.

- **Developers / Maintainers**
  - **UR-3.1 (Must):** Manage user sessions and logs while maintaining scalability for multiple concurrent users.
  - **UR-3.2 (Should):** Update models, vector stores, and interfaces without downtime.
  - **UR-3.3 (Could):** Implement automated monitoring and alerting tools to detect system issues proactively.

System Requirements

Functional Requirements:

- The system should summarize a user-provided document
- The system should extract and display document metadata
- The system should identify and highlight 3 key findings
- The system should answer user questions based on the document
- The system should retrieve relevant context for each question
- The system should track and use past conversation history
- The system should maintain a running summary across updates
- The system should present open questions from the paper
- The system should validate document answers with citations
- The system should support input format in PDF

System Functional Requirements

Requirement Description	Priority	Complexity	Expected Outcomes	How to Implement (Technique / Function / Config)
The system should summarize a user-provided document	High	Medium	Bot replies with a coherent summary	Use a document loader + LLM summarization chain (e.g., map-reduce or refine)
The system should extract and display document metadata	Medium	Low	Metadata (title, authors, date) is visible to user	Use PyMuPDF or pdfplumber to extract frontmatter and meta fields
The system should identify and highlight 3 key findings	High	Medium	Summary includes three concise, high-signal findings	Use LLM prompt engineering to extract "Top 3 takeaways" from summary
The system should answer user questions based on the document	High	High	Bot provides accurate responses grounded in the document	Implement Retrieval-Augmented Generation (RAG) with FAISS + LLM
The system should retrieve relevant context for each question	High	Medium	Q&A cites relevant section(s) from the paper	Use vector-based retrieval (e.g., similarity search in FAISS index)
The system should track and use past conversation history	Medium	Medium	Answers reflect prior user interactions	Use a chat memory module (e.g., ConversationBufferMemory from LangChain)
The system should maintain a running summary across updates	Medium	Medium	Updated summary reflects new content without loss of context	Append refined summaries to a persistent state (e.g., session dict or db)
The system should present open questions from the paper	Medium	Medium	Summary includes open research questions	Prompt LLM to extract "remaining questions" from document
The system should validate document answers with citations	High	Medium	Answers include source references (section/page)	Include chunk metadata in retrieval output and parse into response
The system should support input formats: PDF, text, and markdown	Medium	Low	User can upload papers in multiple formats	Use LangChain document loaders: PyMuPDF, TextLoader, UnstructuredMarkdown

Non-Functional Requirements:

- The system should provide responses within 3 seconds on average
- The system should handle at least 50 concurrent users
- The system should maintain high response accuracy (≥90%)
- The system should preserve formatting and structure of retrieved info
- The system should be accessible to non-technical users through a simple, guided interface
- The system should maintain session-level state during conversations
- The system should log queries and responses for future improvement

System Non-Functional Requirements

Requirement Description	Priority	Complexity	Expected Outcomes	How to Implement (Technique / Function / Config)
The system should provide responses within 3 seconds on average	High	Medium	Users receive quick feedback, ensuring an interactive experience	Optimize LLM chain latency, use async calls, caching, and pre-loading models
The system should handle at least 50 concurrent users	High	Medium	Multiple users can interact with the bot simultaneously without lag	Use FastAPI with async workers, scalable deployment on GPUs or serverless
The system should maintain high response accuracy (≥90%)	High	High	Answers and summaries are consistently grounded in the document content	Use RAG pipeline with reranking; fine-tune LLM or use evaluation metrics like F1
The system should preserve formatting and structure of retrieved info	Medium	Medium	Answers retain technical formatting such as equations, lists, or headings	Use Markdown rendering and HTML-safe outputs
The system should be accessible to non-technical users through a simple, guided interface	Medium	Medium	Users can intuitively access and interact with the bot via a clear web/chat UI	Provide guided commands, tooltips, and a help command or onboarding flow
The system should maintain session-level state during conversations	Medium	Medium	Users can continue context-aware chats without needing to re-upload documents	Use session memory and user-specific IDs to store state
The system should log queries and responses for future improvement	Low	Medium	Query logs help evaluate performance, usage, and identify failure cases	Store logs securely using cloud database or local storage with timestamps