Scientific Paper Analyzer: 2nd Draft Requirements Document

Version: 2.0
Date: October 08, 2025
Authors/Contributors: [Your Name/Team Lead], Grok (AI Assistant)
Team Members (from Proposal): Ali Talasaz, Evan Parra, Bharath Sai Vallabhaneni, Ashritha Aloori

Purpose: This document refines the initial requirements for the Scientific Paper Analyzer component of the SciPaper-Hub repository. It focuses solely on the analyzer (email-based PDF ingestion, summarization, and Q&A), treating it as a standalone module for graduate-level project separation, while noting future integration points with the recommender system. Requirements are structured per SDLC best practices (e.g., IEEE SRS standards): elicitation from user notes/proposal, analysis for completeness/feasibility, and specification in a traceable, verifiable format. Key updates incorporate scalability (serverless), evaluation methods, processing pipeline, frontend UI, limits on papers, and document classification. This draft is for team review; feedback will inform v3.0.

1. Introduction

1.1 Project Overview

The Scientific Paper Analyzer is a cloud-deployed, conversational AI tool designed to simplify interactions with academic literature. Users submit scientific papers via email (PDF attachments or links), and the system processes them for summarization, key insight extraction, and natural-language Q&A. Built on a modular, serverless architecture, it ensures scalability, accuracy, and usability for students, researchers, and educators. This analyzer will eventually integrate with the recommender system in SciPaper-Hub for personalized suggestions based on analyzed papers, but remains independent for now.

1.2 Scope

- **In Scope:** PDF ingestion via email, parsing/embedding, summarization, Q&A with conversational memory, basic classification/validation, simple web UI alternative, evaluation metrics, and serverless deployment on GCP (Cloud Run/Functions).
- **Out of Scope:** Full recommender logic (e.g., similarity-based suggestions across papers), advanced ML training loops (deferred to recommender module), non-English/multilingual support, or real-time collaboration features.

- **Integration Points:** Shared Firestore for state (e.g., parsed embeddings reusable by recommender); Pub/Sub events for future triggering of recommendations.

1.3 Assumptions and Constraints

- **Assumptions:** Users have access to Gmail/email clients; PDFs are primarily scientific papers in English; open-source models (e.g., Gemma 2 9B) are sufficient for LLM tasks; GCP free-tier limits suffice for prototyping.
- **Constraints:** Limit to 10 concurrent papers per user/session (or 1 at a time for simplicity to manage costs/complexity); no proprietary APIs beyond GCP; English-only classification; max PDF size 50MB; deployment budget < $50/month.
- **Dependencies:** IBM Granite Docling/Embeddings, Hugging Face Transformers, GCP services (Pub/Sub, Firestore, GCS, Cloud Run/Functions).

1.4 Glossary

- **RAG:** Retrieval-Augmented Generation.
- **Embedding:** Vector representation of text for similarity/search.
- **Conversational Memory:** Stored chat history for context-aware responses.
- **LLM as Judge:** Using multiple LLMs to evaluate response quality (e.g., accuracy, coherence).

2. User Requirements

Elicited from proposal roles and user notes; prioritized by MoSCoW (Must-Have, Should-Have, Could-Have, Won't-Have).

2.1 User Roles

- **Students/Researchers (Primary Users):** Upload/analyze papers for quick insights and Q&A.
- **Faculty/Reviewers:** Validate summaries/Q&A for teaching/peer review.
- **Developers/Maintainers:** Manage deployment, monitoring, and updates.

2.2 Requirements by Role

Students/Researchers

- **UR-1.1 (Must):** Upload scientific papers via email (PDF attachment or link) or simple web UI; receive confirmation within 10 seconds.
- **UR-1.2 (Must):** Generate concise summaries, extract metadata (e.g., title, authors), highlight 3-5 key findings/methodologies/results.
- **UR-1.3 (Must):** Ask natural-language questions; receive grounded answers with citations to document sections.

- **UR-1.4 (Should):** Maintain conversational memory across emails (resumable via thread ID); limit to 1 active paper per session for simplicity (expandable to 10 with user auth).
- **UR-1.5 (Could):** Provide open questions from the paper (e.g., future work suggestions).

Faculty/Reviewers

- **UR-2.1 (Must):** Access saved interactions/summaries for validation; export as Markdown/PDF.
- **UR-2.2 (Should):** Evaluate response quality via built-in metrics (e.g., LLM-as-judge scores).
- **UR-2.3 (Could):** Flag inaccuracies for system improvement logging.

Developers/Maintainers

- **UR-3.1 (Must):** Monitor system logs/metrics; scale via serverless config.
- **UR-3.2 (Should):** Update models/pipelines without downtime (e.g., blue-green deployment).
- **UR-3.3 (Could):** Integrate donation link in responses (e.g., "Support us: [Link]").

3. System Requirements

3.1 Functional Requirements

Categorized by pipeline stage; traceable to user needs.

- **FR-1: Ingestion**
    - FR-1.1: Support PDF input via email attachment/link or web UI upload; validate as scientific paper in English (e.g., classify using LLM prompt: "Is this a scientific research paper? Yes/No + Reason").
    - FR-1.2: Reject out-of-scope docs (e.g., receipts, non-English, blank forms, medical records) with polite email reply; log for auditing.
    - FR-1.3: Limit to 1 paper per session (thread); queue if >10 total per user (track via Firestore).
- **FR-2: Processing Pipeline**
    - FR-2.1: Parse PDF using IBM Granite Docling to Markdown/JSON (extract text, tables, equations).
    - FR-2.2: Generate embeddings with IBM Granite model; store in FAISS vector store for RAG.
    - FR-2.3: Summarize using open-source LLM (e.g., Gemma 2 9B via HF Transformers); maintain running summary.
    - FR-2.4: Handle Q&A with RAG retrieval; cite sources; use conversational memory (Firestore).
- **FR-3: Output/Interaction**
    - FR-3.1: Reply via email with summaries/Q&A + donation link.

    - FR-3.2: Provide web UI (e.g., Streamlit/Gradio) for alternative upload/Q&A (endpoint triggers same pipeline).
    - FR-3.3: Extract/display metadata, key findings, open questions.
- **FR-4: Evaluation**
    - FR-4.1: Use LLM-as-judge (e.g., query 2-3 models like Llama 3.1/Gemma; average scores for accuracy/coherence on scale 1-5).
    - FR-4.2: Log evaluations; alert if <90% average (for maintainers).

3.2 Non-Functional Requirements

Measured per ISO 25010; aligned with scalability notes.

- **Performance:** Average response <3s (p95 <5s); handle 50 concurrent users via Cloud Run auto-scaling.
- **Scalability:** Containerized serverless (Docker on Cloud Run); auto-scale 0-100 instances; use Pub/Sub for decoupling.
- **Reliability/Availability:** 99% uptime; dead-letter queues for errors; session persistence in Firestore.
- **Usability:** Simple, guided interfaces (email natural; web UI intuitive for non-tech users); preserve formatting in outputs.
- **Security:** IAM roles; secrets in Secret Manager; anonymize user data (hash emails); validate inputs for malware.
- **Maintainability:** Modular code (e.g., separate ingestion/processing); logging for improvements; version control via GitHub.
- **Accuracy:** ≥90% response grounding (measured via LLM judge); citations mandatory.
- **Portability:** GCP-focused but containerized for easy migration (e.g., to AWS).

4. System Architecture Overview

- **High-Level:** Email ingress (Cloud Function + Pub/Sub) → Processing (Cloud Run container: Docling + Embeddings + LLM) → State (Firestore) → Egress (Email reply).
- **Frontend:** Simple web app (Streamlit on Cloud Run) as alternative to email.
- **Evaluation Integration:** Post-processing step in pipeline queries multiple LLMs for judging.

5. Project Timeline (Adapted for SDLC)

- **Week 1:** Requirements refinement (this draft review).
- **Week 2:** Architecture design (update diagrams).
- **Week 3:** Modeling (sequence/class diagrams).
- **Weeks 4-5:** Prototype (ingestion, processing, classification).
- **Week 6:** Experimental testing (evaluation with LLM judge).
- **Week 7:** Deployment (Cloud Run, web UI).

- **Week 8:** Evaluation/closure (feedback, final report).

6. Risks and Mitigations

- **Risk:** LLM hallucinations → Mitigation: RAG grounding + judge evaluation.
- **Risk:** Scalability costs → Mitigation: Limits on papers; serverless auto-scaling.
- **Risk:** Invalid docs → Mitigation: Classification check early in pipeline.
- **Risk:** Integration with recommender → Mitigation: Design modular APIs/State.

7. Next Steps

- Team review: Provide feedback by [Date, e.g., Oct 15, 2025].
- Incorporate into v3.0; align with proposal timeline.
- Prototype key features like classification and web UI.