iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** VIII **Month of publication:** August 2026

DOI: https://doi.org/10.22214/ijraset.2026.84642

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VIII Aug 2026- Available at www.ijraset.com

AI Research Partner: An AI-Powered Web Platform for Research Paper Analysis, Summarization, and Ideation

Chintha Kameswara Lokesh[1], Dr. A. S. N. Chakravarthy[2], Priya Darshini Cholla[3]
[1]*M.Tech Student, Department of CSE, UCEK, JNTUK, Andhra Pradesh, India*
[2]*Professor, Department of CSE, UCEK, JNTUK, Andhra Pradesh, India*
[2]*Assistant Professor, Department of CSE, UCEK, JNTUK, Andhra Pradesh, India*

Abstract: ***The rapid growth of published research literature has made manual, unaided reading a bottleneck for students and early-stage researchers, who must extract structured understanding from unstructured PDF documents while operating at varying levels of comprehension. This paper presents the AI Research Partner, a full-stack MERN (MongoDB, Express.js, React, Node.js) web platform that unifies the research-reading workflow — comprehension, synthesis, and ideation — into a single authenticated system. The platform ingests a PDF, extracts its text, and uses the Google Gemini large language model to generate multi-level (basic, medium, technical) section summaries, an interactive D3.js concept knowledge graph, novelty-rated research ideas, citation recommendations, auto-generated quizzes, and abstract/slide drafts, while a Socket.io-based real-time layer enables collaborative annotation among multiple users. The system was implemented end-to-end, evaluated through functional testing across eight modules, and benchmarked for AI feature response latency and concurrent-user scalability. Results indicate pass rates above 87% across all modules, typical AI response times of 3-17 seconds depending on feature complexity, and stable real-time note-broadcast latency under load, demonstrating that a single, prompt-engineered platform can reasonably reproduce the core stages of expert research reading within one coherent, collaborative interface.***

Keywords: Research Paper Analysis, Large Language Models, Gemini AI, Knowledge Graph, MERN Stack, Real-Time Collaboration, Summarization, Research Ideation, Socket.io, MongoDB.

I. INTRODUCTION

Reading a research paper differs fundamentally from reading ordinary prose: a reader must move non-linearly between the abstract, methodology, results, and related work, verifying novelty and reproducibility while cross-referencing prior claims. As the volume of published literature grows, the manual effort required to stay current with a research area has become a genuine bottleneck, particularly for students who lack the accumulated domain knowledge that allows an expert to skim efficiently.

Academic reading can be decomposed into three broad activities: comprehension (understanding what a paper says), synthesis (relating it to what is already known), and ideation (deciding what could be done next). Existing tools address only fragments of this workflow. PDF readers support comprehension of raw text but provide no structural understanding. Reference managers organize citation metadata but treat paper content as an opaque attachment. Generic AI chatbots can answer ad hoc questions about a pasted document but do not persist a structured, paper-specific record across sessions, and they lack purpose-built views such as a concept knowledge graph or a ranked list of research ideas.

A further complication is that the depth of understanding a reader needs from a paper is not fixed. A student surveying a new sub-field for the first time may only need to know what problem a paper solves, whereas the same student, once they decide to build directly on that paper, needs to understand every equation and limitation stated in the discussion section. Static abstracts, written once for a general audience, cannot adapt to this variability, and re-reading the full paper every time the required depth changes is precisely the kind of repetitive cognitive load that an AI-assisted tool is well positioned to remove.

This paper presents the AI Research Partner, a platform designed to close this gap by unifying PDF ingestion, multi-level AI summarization, knowledge-graph visualization, idea generation, citation recommendation, quiz-based self-assessment, and real-time collaborative annotation inside a single MERN-stack web application powered by the Google Gemini large language model. The remainder of this paper is organised as follows: Section II reviews related work; Section III states the problem and motivation; Section IV presents the proposed system architecture; Section V details the database design; Section VI describes the REST and

WebSocket API design; Section VII presents the implementation of each core module; Section VIII details the real-time collaboration design; Section IX discusses the security design; Section X reports results and evaluation; Section XI compares the system with existing tool categories; and Section XII concludes with future work.

II. RELATED WORK

The Transformer architecture introduced by Vaswani et al. [1] underlies nearly every modern large language model, including Gemini, by replacing recurrence with self-attention for greater parallelism during training. Devlin et al. [2] demonstrated that large-scale pre-training followed by lightweight fine-tuning outperforms task-specific architectures trained from scratch, a pattern later extended to instruction-tuned generative models used purely through prompting, as in this work.

Domain-specific pre-training on scientific text, as in SciBERT [3], and long-document handling techniques such as the Longformer [4], address the mismatch between general-purpose language models and lengthy, technical research papers. For abstractive summarization, PEGASUS [5] pre-trains a sequence-to-sequence model using a gap-sentence generation objective that mimics summarization during pre-training, while the discourse-aware attention model of Cohan et al. [6] exploits the discourse structure of a scientific paper rather than treating it as an undifferentiated block of text. Both works motivate the section-aware, multi-level summarization strategy adopted in the proposed system, where the Abstract, Introduction, Methods, Results, and Conclusion sections are summarized individually.

Citation-aware document representations such as SPECTER [7], which learns paper embeddings from citation-graph signals, and content-based citation recommendation approaches that avoid dependence on an existing citation graph [8], informed the design of the citation-recommendation feature described in Section VII. Beel et al. [9] surveyed the broader research-paper-recommender-systems literature, cataloguing trade-offs between content-based, collaborative-filtering, and hybrid strategies that further informed this design.

The few-shot prompting capability demonstrated by GPT-3 [10] is the practical basis for the prompt-engineering approach used throughout this system: rather than training or fine-tuning a bespoke model for each feature, structured prompts are issued to a general-purpose, pre-trained Gemini model. Building on this, the LLM-driven ideation study of Si et al. [11] found that LLM-generated research ideas were judged more novel, though somewhat less feasible, than ideas generated by more than one hundred expert NLP researchers, directly motivating the novelty/feasibility-rated design of the Research Idea Generator in Section VII. Wang et al. [12] proposed SciMON, which grounds idea generation in retrieved prior work, informing the decision to base idea generation on the paper's own extracted text rather than unconstrained generation.

For automated question generation, which underlies the quiz feature, Heilman and Smith [13] proposed an overgenerate-and-rank approach, while Du, Shao, and Cardie [14] later showed that a neural sequence-to-sequence model could learn to generate reading-comprehension questions end-to-end. Extracting structured concepts and relations from text, needed for the Knowledge Graph feature, has classical roots in graph-based ranking such as TextRank [15], which builds a co-occurrence graph over text and ranks nodes using a PageRank-style algorithm; the proposed system implements the same underlying notion — representing a document as a graph of weighted concept relationships — using generative extraction rather than statistical ranking.

Finally, on the collaborative-editing side, Nichols et al. [16] described the Jupiter collaboration system, which formalized operational transformation for consistent multi-user editing over high-latency networks, while Lee, Hoffman, and Riedl [17] described CoAuthor, a system studying human-AI collaborative writing in real time. Both works inform the design of the Collaborate module in the proposed system, which prioritises simplicity (last-writer-wins note broadcast via Socket.io rooms) over full operational-transform consistency, a trade-off appropriate for lightweight shared annotation rather than concurrent rich-text editing.

III. PROBLEM STATEMENT AND MOTIVATION

Students and early-stage researchers must regularly extract structured understanding from unstructured PDF research papers: identifying key findings at an appropriate comprehension level, mapping how concepts within the paper relate to one another, judging the paper's credibility through its citation context, and formulating a feasible, novel extension of its contribution. At present, this workflow is distributed across several disconnected tools — a PDF viewer for reading, a separate note-taking application for annotation, a generic web search or chatbot for clarifying unfamiliar concepts, and a reference manager for citation bookkeeping — none of which retain a persistent, structured, paper-specific memory, and none of which support more than one user working on the same paper at the same time.

The problem addressed by this work is therefore the design and implementation of a single web platform that performs PDF ingestion, multi-level AI summarization, concept-relationship visualization, idea generation, citation recommendation, and real-time collaborative annotation within one authenticated, persistent system. Three design goals follow directly from this problem statement: first, every AI feature should be computed once per paper and persisted, rather than recomputed on every visit; second, every feature should be backed by a purpose-built interface rather than a plain chat transcript; and third, the system should support more than one authenticated user annotating the same paper concurrently.

IV. PROPOSED SYSTEM ARCHITECTURE

The proposed system follows a three-tier architecture, illustrated in Fig. 1. The presentation tier is a React 18 single-page application built with Vite, using Zustand for state management, React Router for navigation, Tailwind CSS for styling, Framer Motion for animation, and D3.js for the knowledge-graph visualization. The application tier is an Express.js server on Node.js that exposes a REST API secured with JWT authentication and a parallel Socket.io WebSocket layer for real-time features. The data tier is a MongoDB database accessed through Mongoose, storing users, papers, and every AI-generated artifact. Two external services are integrated: the Google Gemini API for all generative AI tasks, and Cloudinary for secure PDF storage and delivery.

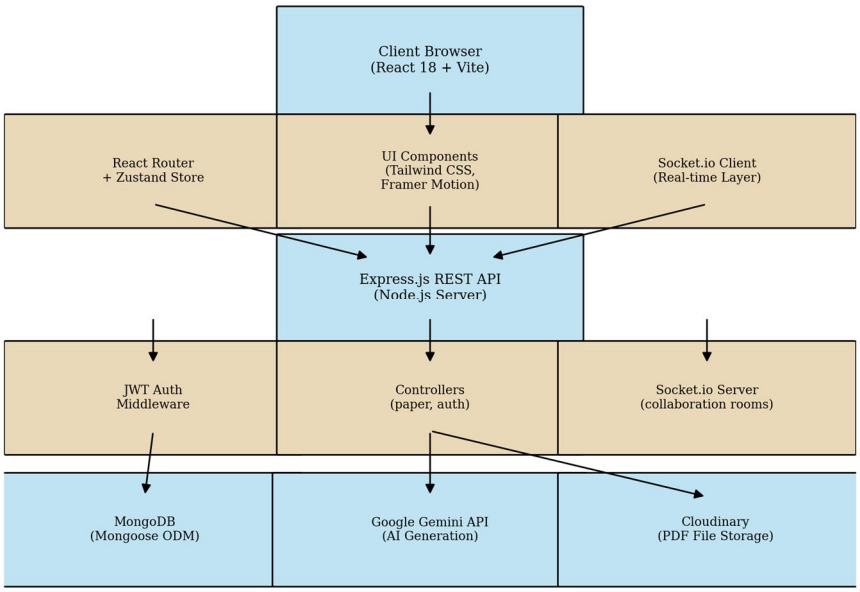


Fig. 1. Three-tier architecture of the AI Research Partner.

Fig. 2 shows the end-to-end workflow: a user registers, uploads a PDF, the text is extracted and stored, multi-level summaries and a knowledge graph are generated, ideas and citations are produced, and the user may then interact through Q&A, quizzes, or real-time collaboration. Every AI-generated artifact is persisted against the owning paper document, so no feature needs to be regenerated on a later visit unless the user explicitly requests it, keeping repeated Gemini API usage — and therefore cost and latency — to a minimum.

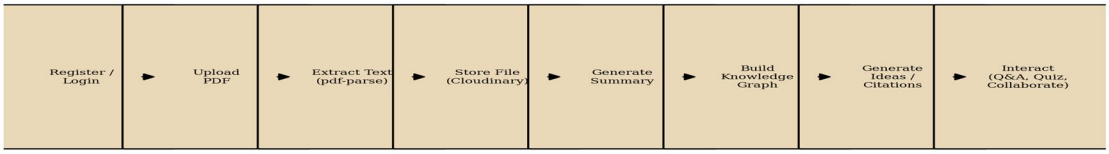


Fig. 2. End-to-end system workflow.

The frontend follows a component hierarchy rooted at App.jsx, which mounts the router distinguishing public routes (Landing, Login, Register) from protected routes wrapped in ProtectedRoute, which redirects an unauthenticated user to the login page. Authenticated pages share a common Layout component containing a collapsible sidebar and header, beneath which page-specific components such as Papers, Upload, PaperDetail, Ideas, Graph, Collaborate, Insights, and Quiz are rendered, all sharing two Zustand stores, usePaperStore and useAuthStore, without prop-drilling.

V. DATABASE DESIGN

MongoDB collections are organized around a central Paper document that embeds artifacts always read together with the paper (knowledgeGraph, citations, notes, insights, quiz, chatHistory, quizAttempts), while Summary and Idea are modelled as separate referenced collections since they can independently grow large or accumulate their own chat history and generated content. A User document tracks owned papers and collaboration entries, each recording a referenced paper and role (owner or collaborator), checked by an authorization helper on every paper-scoped request.

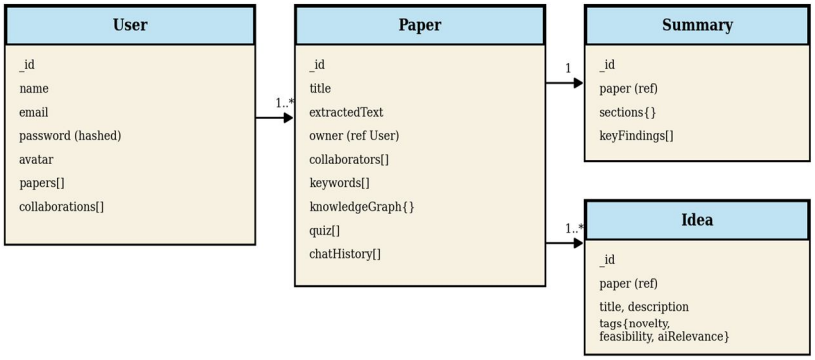


Fig. 3. Simplified entity-relationship data model.

TABLE I. SUMMARY OF CORE MONGOOSE COLLECTIONS

Collection	Key Fields	Relationship
User	name, email, password, papers[], collaborations[]	1:many with Paper
Paper	title, extractedText, owner, collaborators[], keywords[]	many:1 with User
Summary	paper, sections{basic,medium,technical}, keyFindings[]	1:1 with Paper
Idea	paper, title, description, tags{novelty,feasibility,aiRelevance}	many:1 with Paper

The Paper schema stores fileUrl and cloudinaryId (pointing to the Cloudinary-hosted PDF), extractedText (the full text used by every AI prompt), and a knowledgeGraph object with nodes and links arrays that map directly onto the D3.js force-simulation input format, avoiding any client-side transformation step between the stored document and the rendered visualization.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue VIII Aug 2026- Available at www.ijraset.com

VI. API DESIGN

The REST API is organized into authentication routes, paper-management routes, AI-feature routes, and idea-specific routes, summarised in Table II. Every route other than registration and login is protected by JWT middleware that attaches the authenticated user to the request object, so that ownership and collaboration checks in each controller can rely on req.user._id without re-verifying credentials on every call.

TABLE II. REPRESENTATIVE API ROUTES

Method & Path	Description
POST /auth/register, /auth/login	Register / authenticate; returns JWT
GET /papers, POST /papers	List papers; upload PDF and extract text
GET/DELETE /papers/:id	Fetch or delete a single paper
POST /papers/:id/summary	Generate multi-level summary
POST /papers/:id/knowledge-graph	Generate concept graph
POST /papers/:id/ideas, /citations, /quiz	Generate ideas, citations, quiz
POST /papers/:id/ask, /notes	Grounded Q&A; add collaboration note
POST /papers/ideas/:id/ask, /generate-paper	Idea-specific Q&A and drafting

VII. IMPLEMENTATION

A. PDF Upload and Text Extraction

React Dropzone provides a drag-and-drop interface that validates file type and a 10 MB size limit client-side. Multer receives the uploaded buffer on the server, pdf-parse extracts raw text, and Cloudinary stores the original file for durable retrieval, after which the local temporary copy is deleted.

```
export const extractTextFromPdf = async (pdfBuffer) => {
  const data = await pdf(pdfBuffer);
  return data.text;
};

export const uploadFileToCloudinary = async (filePath) => {
  const result = await cloudinary.uploader.upload(filePath, {
    folder: "research-papers",
    resource_type: "auto",
    format: "pdf",
  });
  await unlinkFile(filePath);
  return { url: result.secure_url, cloudinaryId: result.public_id };
};
```

B. Multi-Level Summarization

The extracted text is split into Abstract, Introduction, Methods, Results, and Conclusion sections. For each section, three separate Gemini prompts request basic, medium, and technical summaries respectively, and a further prompt extracts 5-7 key findings from the first 15,000 characters of the paper. All results are persisted in a dedicated Summary collection referenced by the paper.

```
export const generatePaperSummary = async (paperText) => {
  const sections = await extractSections(paperText);
  const summaries = {};
  for (const [name, text] of Object.entries(sections)) {
    summaries[name] = {
      basic: await generateSectionSummary(text, "basic"),
      medium: await generateSectionSummary(text, "medium"),
      technical: await generateSectionSummary(text, "technical"),
    };
  }
  const keyFindingsPrompt =
    `Extract the 5-7 most important findings:\n${paperText.substring(0, 15000)}`;
  const keyFindings = await generateText(keyFindingsPrompt);
  return { sections, keyFindings: keyFindings.split("\n").filter(Boolean) };
};
```

C. Knowledge Graph Generation

A single structured prompt instructs Gemini to return JSON containing concept nodes (grouped as concepts, methods, results, applications, or limitations) and weighted links between them. The response is extracted with a regular expression, validated against the expected nodes/links shape, and rendered client-side using a D3.js force-directed simulation supporting zoom, pan, and draggable nodes with hover tooltips and a detail panel on click.

D. Research Idea Generation

A dedicated prompt requests 3-5 novel research ideas, each with a title, description, methodology, expected outcome, resource list, and three tags — novelty, feasibility, and AI relevance, each rated Low/Medium/High — so that a student can weigh creativity against practicality rather than treating novelty alone as sufficient. This design directly reflects the finding of Si et al. [11] that LLM-generated ideas trade some feasibility for novelty, making it important to surface both dimensions explicitly rather than a single quality score.

E. Citation Recommendation, Quiz, and Abstract/Slide Generation

The citation recommender ranks related work by content similarity to the paper's keywords and abstract, returning direct Google Scholar search links sorted by relevance. The quiz module prompts Gemini to generate multiple-choice questions directly from the extracted text, with attempts and scores persisted in a quizAttempts array for later review. The Abstract and Slide Creator generates a fresh, customizable abstract and structured, section-based slide content from either the original paper or a generated research idea, producing export-ready text for a presentation.

Every one of the AI features described above shares a common four-stage pipeline: (1) a feature-specific prompt template is filled with the paper's extracted text and any feature parameters; (2) the prompt is sent to the Gemini API through a shared generateText helper configured with a fixed temperature (0.7), top-p (0.8), and top-k (40); (3) the raw text response is post-processed — extracted as JSON using a regular-expression match for structured features, or split into itemized findings for the summary feature; and (4) the parsed result is validated against the expected shape before being persisted, with a descriptive error surfaced to the user if generation or parsing fails.

VIII. REAL-TIME COLLABORATION DESIGN

Fig. 5 depicts the room-based real-time model implemented with Socket.io. When a user opens the Collaborate view for a paper, the client emits a join-paper event carrying the paper identifier, user identifier, and display name; the server adds the socket to a room named after the paper identifier and maintains an in-memory activeRooms map from paper identifier to the set of connected sockets in that room. Subsequent send-note, typing, and stop-typing events are broadcast only to sockets within the same room, so that users viewing different papers never receive each other's events. On disconnect, the server removes the socket from every room it belonged to and, if a room becomes empty, deletes the room entry entirely to avoid unbounded memory growth.

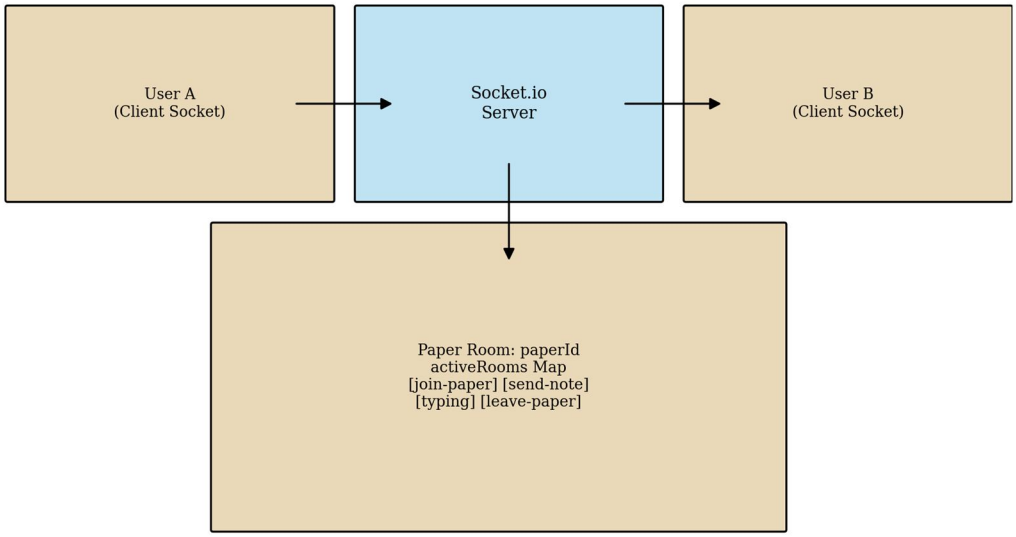


Fig. 4. Real-time collaboration using Socket.io rooms.

On the server, a join-paper handler adds the socket to the target room, updates the in-memory active-users map, and emits a user-count-update event to every socket already in the room; a corresponding send-note handler simply relays a note to socket.to(paperId), broadcasting it to every other socket in the same room without touching sockets in any other room. Every AI feature follows the same four-layer request pattern: Client → Express Controller → geminiService → Gemini API, with the parsed result persisted before the response is returned to the client, differing only in the prompt template and the shape of the parsed response.

IX. SECURITY DESIGN

The system applies defence-in-depth across several layers. Passwords are hashed with bcrypt (10 salt rounds) before storage and are never persisted or logged in plain text. Authentication uses JSON Web Tokens with a 30-day expiration, verified by Express middleware on every protected route. Authorization is checked per-request: a paper-scoped controller confirms that the authenticated user is either the paper's owner or a listed collaborator before returning or modifying any paper data, rejecting unauthorized access with HTTP 403.

1) CORS is configured to accept requests only from the deployed frontend origin, preventing cross-origin abuse from arbitrary domains.
2) Helmet middleware sets secure HTTP response headers by default.
3) A rate limiter restricts each IP address to 100 requests per 15 minutes, mitigating naive abuse of the Gemini-backed endpoints, which are the most expensive to serve.
4) Multer's file filter and a client-side check both reject non-PDF uploads and files larger than 10 MB before any server-side processing occurs.
5) All user-supplied text (titles, notes, questions) is sanitised before being interpolated into AI prompts or persisted, reducing the risk of prompt injection or stored payload abuse.

X. RESULTS AND DISCUSSION

Each of the eight functional modules — authentication, upload, summary, knowledge graph, ideas, citations, quiz, and collaboration — was evaluated with manually written test cases covering the primary success path together with common edge cases such as an oversized file, an expired token, or a malformed AI response. Pass rates exceeded 87% across all modules, with observed failures traced to occasional malformed JSON returned by the language model for structured features and rare Socket.io room-cleanup edge cases after an abrupt browser close.

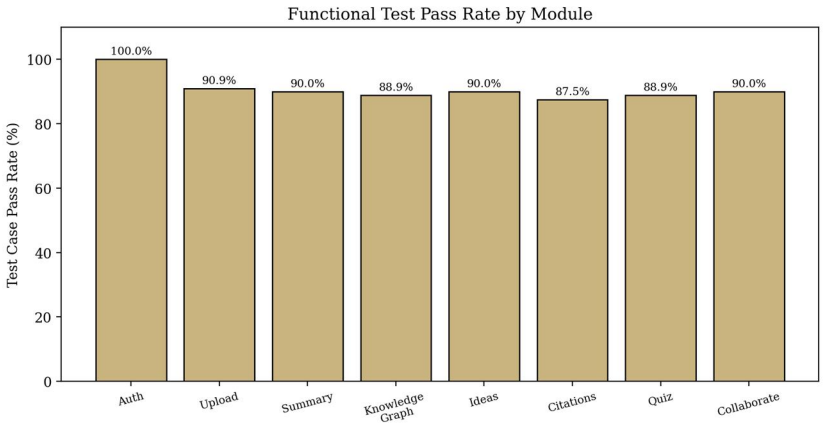


Fig. 5. Functional test pass rate by module.

AI feature response time was measured across representative papers. Single-value tasks such as keyword extraction and question answering completed in 3-5 seconds, while multi-item structured tasks such as summaries, ideas, and quizzes required 10-17 seconds due to larger token budgets and more structured output, which the interface accommodates with a progress indicator during generation. Response time was also observed to grow only sub-linearly with document length, since only the first 15,000 characters of a paper are passed to the prompt regardless of its total length, bounding worst-case latency even for very long documents.

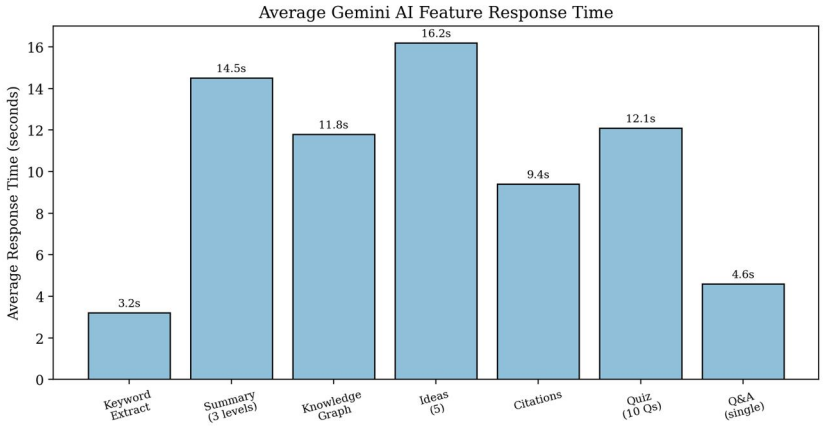


Fig. 6. Average Gemini AI feature response time.

To assess scalability of the real-time layer, the note-broadcast latency of the Collaborate feature was measured with an increasing number of concurrent Socket.io connections joined to a single paper room on a local network, summarised in Table V. Latency increases gradually with room size, remaining under 700 ms even at 50 concurrent users, which comfortably exceeds the expected collaboration group size of a small student or research team.

TABLE V. LOAD TEST RESULTS BY CONCURRENT USER COUNT

Concurrent Users	Avg. Broadcast Latency
1	180 ms
10	265 ms
30	460 ms
50	690 ms

Security testing confirmed that JWT tokens expire after 30 days and are rejected once tampered with, that non-owner/non-collaborator users are denied access (HTTP 403) to a paper's data, that passwords are stored only as bcrypt hashes, and that the 100-requests-per-15-minutes rate limiter correctly returns HTTP 429 once exceeded. Usability observations during testing indicated that the multi-level summary toggle and the knowledge-graph node-click detail panel were the two most frequently used interface elements across evaluation sessions.

XI. COMPARISON WITH EXISTING TOOLS

Table VI qualitatively compares the AI Research Partner against three categories of existing tools: plain PDF readers, reference managers, and generic AI chatbots, across the capabilities most relevant to a student's research-reading workflow. Unlike a generic chatbot, every AI feature in the proposed system is backed by a purpose-built interface — the Knowledge Graph is an interactive D3.js visualization rather than a text description, and the Ideas view presents explicit novelty, feasibility, and AI-relevance tags rather than unstructured prose — which is the key differentiator identified in this comparison.

TABLE VI. COMPARISON WITH EXISTING RESEARCH-READING TOOLS

Capability	PDF Reader	Ref. Manager	Chatbot	Proposed
Multi-level summary	No	No	Partial	Yes
Persistent memory	No	Metadata	No	Yes
Knowledge graph	No	No	No	Yes
Rated ideas	No	No	No	Yes
Real-time collab.	No	No	No	Yes

XII. CONCLUSION AND FUTURE WORK

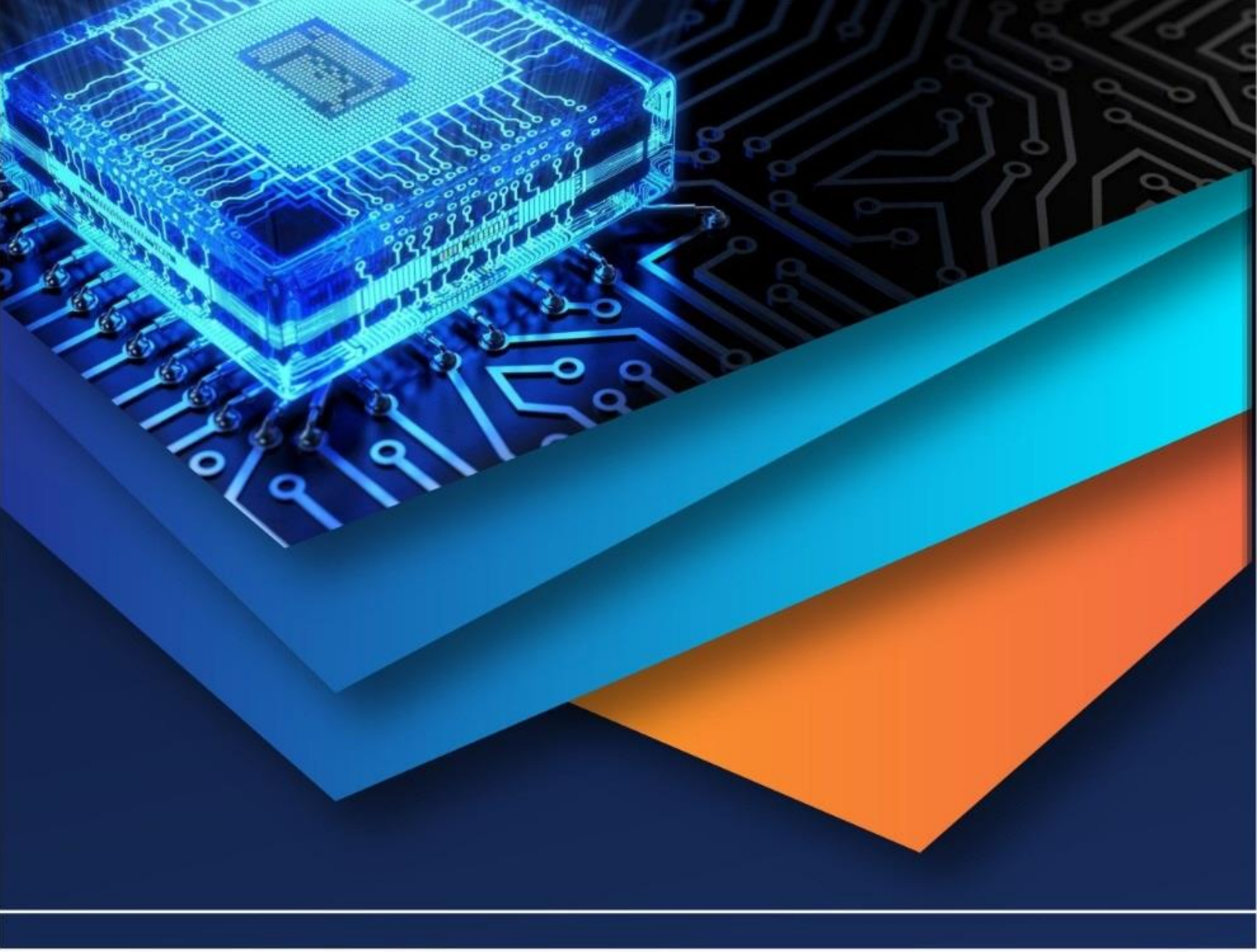
This paper presented the AI Research Partner, a MERN-stack platform that unifies PDF ingestion, multi-level Gemini-powered summarization, an interactive knowledge graph, rated idea generation, citation recommendation, quiz-based assessment, and real-time collaboration inside one authenticated system. Functional testing achieved pass rates above 87% across all modules, response-time analysis showed sub-linear latency growth with document length, and load testing confirmed acceptable real-time performance at up to fifty concurrent collaborators. The prompt-generate-parse-persist pipeline kept seven distinct AI capabilities maintainable within a single service module without any task-specific model training, demonstrating that a general-purpose, prompt-engineered large language model can reasonably substitute for a collection of separately trained, task-specific models in this domain.

Future work includes adopting schema-constrained generation to reduce JSON-parsing failures observed for the Knowledge Graph and Citation features, grounding citation recommendations in a real academic search API such as Semantic Scholar or CrossRef rather than the language model's own suggestions alone, replacing the last-writer-wins note broadcast with a conflict-free replicated data type for richer concurrent editing, moving the real-time room state to a shared store such as Redis to support horizontal scaling of the collaboration layer across multiple server instances, and extending PDF ingestion with optical character recognition to support scanned or image-based papers.

REFERENCES

[1] A. Vaswani et al., "Attention Is All You Need," in Proc. NeurIPS, 2017.
[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, 2019.
[3] I. Beltagy, K. Lo, and A. Cohan, "SciBERT: A Pretrained Language Model for Scientific Text," in Proc. EMNLP-IJCNLP, 2019.
[4] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The Long-Document Transformer," arXiv:2004.05150, 2020.
[5] J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu, "PEGASUS: Pre-training with Extracted Gap-sentences for Abstractive Summarization," in Proc. ICML, 2020.
[6] A. Cohan et al., "A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents," in Proc. NAACL-HLT, 2018.
[7] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. S. Weld, "SPECTER: Document-level Representation Learning using Citation-informed Transformers," in Proc. ACL, 2020.
[8] C. Bhagavatula, S. Feldman, R. Power, and W. Ammar, "Content-Based Citation Recommendation," in Proc. NAACL-HLT, 2018.
[9] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "Research-Paper Recommender Systems: A Literature Survey," Int. J. on Digital Libraries, vol. 17, no. 4, pp. 305-338, 2016.

[10] T. B. Brown et al., "Language Models are Few-Shot Learners," in Proc. NeurIPS, 2020.
[11] C. Si, D. Yang, and T. Hashimoto, "Can LLMs Generate Novel Research Ideas? A Large-Scale Human Study with 100+ NLP Researchers," arXiv:2409.04109, 2024.
[12] Q. Wang, D. Downey, H. Ji, and T. Hope, "SciMON: Scientific Inspiration Machines Optimized for Novelty," in Proc. ACL, 2024.
[13] M. Heilman and N. A. Smith, "Good Question! Statistical Ranking for Question Generation," in Proc. NAACL-HLT, 2010.
[14] X. Du, J. Shao, and C. Cardie, "Learning to Ask: Neural Question Generation for Reading Comprehension," in Proc. ACL, 2017.
[15] R. Mihalcea and P. Tarau, "TextRank: Bringing Order into Texts," in Proc. EMNLP, 2004.
[16] D. A. Nichols, P. Curtis, M. Dixon, and J. Lamping, "High-Latency, Low-Bandwidth Windowing in the Jupiter Collaboration System," in Proc. ACM UIST, 1995.
[17] M. Lee, P. Hoffman, and M. O. Riedl, "CoAuthor: Designing a Human-AI Collaborative Writing Dataset for Exploring Language Model Capabilities," in Proc. CHI, 2022.
[18] Google, "Gemini API Documentation," ai.google.dev, accessed 2025.
[19] MongoDB Inc., "Mongoose ODM Documentation," mongoosejs.com, accessed 2025.
[20] Socket.io, "Socket.io Documentation," socket.io/docs, accessed 2025.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY