IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151183

AI-Based Student Enquiry Chatbot Using Retrieval-Augmented Generation for Institute

Manisha S. Patil[1], Divya J. Khairnar[2], Kartik S. Patil[3], Ashwini Y. Deore[4], Diksha S. Chaudhari[5]

Assistant Professor, Department of Computer Science & Engineering (Data Science), R. C. Patel Institute of Technology, Shirpur, Maharashtra, India[1]

UG Student, Department of Computer Science & Engineering (Data Science), R. C. Patel Institute of Technology, Shirpur, Maharashtra, India[2]

UG Student, Department of Computer Science & Engineering (Data Science), R. C. Patel Institute of Technology, Shirpur, Maharashtra, India[3]

UG Student, Department of Computer Science & Engineering (Data Science), R. C. Patel Institute of Technology, Shirpur, Maharashtra, India[4]

UG Student, Department of Computer Science & Engineering (Data Science), R. C. Patel Institute of Technology, Shirpur, Maharashtra, India[5]

Abstract: Timely access to information on academic departments, admission standards, syllabus regulations, and campus facilities is often necessary for college students and prospective applicants. This administrative information is dispersed across dozens of distinct circulars, brochures, and syllabus documents at many institutions, such as R. C. Patel Institute of Technology (RCPIT), Shirpur. As a result, users must manually search multiple PDF files or make repeated in-person inquiries at campus offices. This study discusses the design and development of a domain-specific inquiry chatbot that uses Retrieval-Augmented Generation (RAG) to handle this problem. Our approach limits its knowledge corpus to authorized institutional PDF publications only, in contrast to generic conversational models that rely on ungrounded parametric memory or unverified web data. The system pipeline creates 384-dimensional dense vector embeddings using the all-MiniLM-L6-v2 Sentence Transformer, extracts raw text from college PDFs, sanitizes and divides the content into coherent passages, and indexes them using Facebook AI Similarity Search (FAISS). When a user asks an inquiry, the system uses semantic similarity to extract the most contextually relevant chunks, which it then provides to a generative big language model in order to synthesize concise, accurate answers. During preliminary testing, the prototype which was created in Python for eventual incorporation into the institute's official website successfully indexed 19 institutional PDF documents into 318 vector segments. While formal empirical evaluations which include retrieval precision, response groundedness, latency, and student usability are described as continuing study, preliminary experiments show accurate passage retrieval across departmental brochures.

Keywords: Retrieval-Augmented Generation (RAG), conversational agent, student inquiry system, natural language processing, Sentence Transformers, FAISS vector index, educational technology, institutional knowledge base.

I. INTRODUCTION

Every academic year, a significant amount of official paperwork is released by academic institutions, such as administrative circulars, departmental brochures, academic calendars, examination rules, and entrance prospectuses. Finding pertinent information in these scattered documents can be time-consuming for potential applicants and enrolled students looking for concise, practical solutions. Conventional institutional websites rely on intricate navigation menus or static Frequently Asked Questions (FAQ) pages. Static FAQs are helpful for basic guidance, but they are hard for administrative personnel to keep up with on a regular basis. They also fail when a response spans numerous papers or departments and cannot easily handle natural changes in student language. By allowing students to ask questions in common English and obtain comprehensive, trustworthy responses, conversational artificial intelligence provides an efficient means of closing this gap. Recent assessments of automated conversational agents in higher education have highlighted their usefulness in decreasing regular clerical duties and simplifying student access to academic and administrative guidance [5], [7]–[10].

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151183

In this study we create a chatbot for students at R. C. Patel Institute of Technology. This is an engineering college in Shirpur, Maharashtra. Standard large language models cannot be used for this task because their knowledge is fixed at the time they were trained. They do not know about the rules of the campus the fees schedule or what departments are offering. Also these models often make up things that're not true. This is called hallucination. To fix this problem we use a system called Retrieval-Augmented Generation. Of just making up answers this system uses real documents. When a student asks a question the system looks for the relevant parts from official RCPIT documents. These parts are then used to help the model give an answer that is based on the information, from the college [6], [12], [13].

To systematically address this, we need to utilize the natural bottlenecks in our transit networks: toll plazas. With their structured lanes, predictable traffic flow, and existing fixed-camera infrastructure, toll gates provide an ideal, controlled environment for monitoring rider safety. The main thing this work does is create a easy to repeat software process. This process takes in PDFs cleans and splits the messy text into parts that make sense makes special number groups for each part finds similar parts really fast and then gives answers that people can read and understand through a website chat tool.

The rest of this paper is set up like this:

- Section 2 looks at research finds what is missing and says what the project wants to achieve.
- Section 3 explains how the system is built and how it works.
- Section 4 shows the code, the data, some early tests and how we plan to check everything properly.
- Section 5 talks, about what we found the problems we faced and what is not working well.
- Section 6 compares this system to ways of asking questions.
- Section 7 wraps up the study. Says what we want to do next.

II. RELATED WORK AND LITERATURE REVIEW

Conversational agents that can talk and help people have become very popular in schools over the world. Ade-Ibijola [5] did a study of various chatbots used in education and they wrote about how these tools help with work and let students help themselves. Large language models on the hand can make up things that sound right but are not true. This is called "hallucinations" [6]. In a school setting giving information about whether a student qualifies for something or showing old fees can really confuse students.

Using a way to get information, from school documents helps reduce these mistakes but there are still times when the school documents do not have the answer and that is when other plans are needed [6]. More recent studies by Peyton et al. [7] Ramandanis and Xinogalos [8] Kuhail et al. [9] And Guan et al. [10] Say that chatbots can make students more interested and easier to reach. These tools only work well in real life if the answers are correct, clear and true.

TABLE I. SUMMARY OF WORK AND LITERATURE REVIEW

Ref.	Contribution	Relevance to this project
[1]	Formulated the RAG framework combining parametric LLMs with non-parametric retrieval memory. Defines the core architectural paradigm: retrieve relevant college passages prior to generating answers	Defines the core architectural paradigm: retrieve relevant college passages prior to generating answers.
[2]	Introduced Sentence-BERT for generating semantically comparable sentence embeddings.	Supplies the embedding model (all-MiniLM-L6-v2) to represent document chunks and student queries.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151183

Ref.	Contribution	Relevance to this project
[3]	Engineered FAISS for high-throughput similarity search in high-dimensional vector spaces.	Powers the vector database and nearest-neighbor search for low-latency chunk retrieval.
[4]	Demonstrated dense passage retrieval outperforming sparse lexical baselines on recall.	Justifies using dense semantic search over keyword matching to extract relevant college rules.
[5]	Systematic review of various chatbot implementations across higher education.	Underscores the organizational need and practical benefits of automated campus helpdesks.
[6]	Comprehensive survey on hallucination mechanisms, detection, and mitigation in NLG.	Emphasizes the necessity of grounding responses in source excerpts and adding fallback guardrails.

III. RETRIEVAL-AUGMENTED GENERATION AND DENSE RETRIEVAL

A. Retrieval-Augmented Generation

Lewis et al. [1] Created the Retrieval-Augmented Generation (RAG) approach by combining -trained sequence-to-sequence models with an outside document index. Their test results showed that using information in generating answers greatly reduces mistakes in tasks that need a lot of knowledge. For college offices RAG offers a benefit: rules, costs and course outlines can be changed quickly in the database without spending a lot of time and resources on changing the model.

To get the parts of information Karpukhin et al. [4] Developed Dense Passage Retrieval (DPR) showing that using two encoder models that put questions and parts of text into the space works much better than older methods like BM25 for finding the best documents. Dense retrieval understands the meaning behind the words so student questions can find the information even if they are not exactly the same as the words, in official documents.

B. Sentence Embeddings and Vector Search

To create representations of text parts, Reimers and Gurevych [2] created Sentence-BERT (SBERT). By using triplet network setups SBERT produces fixed-length sentence embeddings that carry semantic meaning. These embeddings can be compared with cosine similarity or Euclidean distance. This approach reduces the computing load of cross-encoder models while still keeping fine semantic details.

For fast nearest-neighbour searches over high-dimensional embeddings Johnson et al. [3] Released Facebook AI Similarity Search (FAISS). FAISS supplies GPU and CPU optimized indexing structures that allow quick similarity queries, in vector spaces. In our system SBERT creates embeddings for parsed institutional text blocks and FAISS delivers quick nearest-neighbour answers when students ask questions.

IV. PROBLEM STATEMENT AND OBJECTIVES

A. Problem Statement

Students and parents who want details, about RCPITs academic programs, admissions criteria and institutional policies often have to sift through many long PDF files or visit administrative offices during working hours. Going through these documents by hand is hard and rule-based chatbots break when questions do not match strict keyword patterns. The main problem is to build a search and chat system that takes free-form natural language questions finds the right parts of RCPIT's official PDF records and gives clear correct answers without making up facts.

B. Project Objectives

To tackle the research and engineering challenges mentioned earlier this project focused on the following goals:

1) *Curate an Authentic Repository:* Gather and organize a reliable knowledge base made up only of official RCPIT PDF publications. The content must be. Authentic, ensuring that all documents come from trusted institutional sources.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151183

2) *Text Ingestion and Cleansing:* Perform an instant, geometry-aware headcount to calculate the exact number of individuals mounted on each vehicle. The AI must be trained to reliably differentiate between actual human passengers and inanimate objects—such as heavy backpacks, delivery boxes, or cargo strapped to the bike—to catch dangerous triple-riding scenarios without triggering false alarms.

3) *Dense Vector Representation:* Convert each text segment into a vector representation using the all-MiniLM-L6-v2 Sentence Transformer model. This step helps capture the meaning of the text in a numerical format.

4) *Fast Vector Indexing:* Use FAISS to create an efficient vector index. This allows for quick and accurate similarity searches, across all stored text chunks, when dealing with large volumes of data.

5) *Grounded Generation:* Design structured RAG prompts that guide the language model to generate responses based on retrieved passages. The system should produce natural answers and handle queries that fall outside the available knowledge with a clear fallback mechanism.
6) *Web Interface Integration:* Create an user-friendly conversational interface that can be easily embedded into the official RCPIT website allowing users to interact with the system seamlessly.

V. RESEARCH GAP

Although dense retrieval, semantic embeddings and RAG architectures are well explained in theory there is still a gap when it comes to using them in small local engineering colleges in regional areas. Pre-trained language models. Whether from companies or open-source projects. Don't know anything about the specific rules of a college like its admission quotas, department facilities or academic schedules. These models also can't understand the documents that colleges use, which are often messy PDFs with many columns, strange layouts, fancy designs and hidden tables.

Building a chatbot that actually works for a college needs hands-on work. It means pulling text out of those complicated PDFs cleaning it up breaking it into pieces storing it in a system that can quickly find relevant parts and then linking all of that to a simple web page students and staff can use. This project focuses on solving those problems, for R. C. Patel Institute of Technology Shirpur.

VI. PROPOSED SYSTEM ARCHITECTURE

A. Overview of the proposed system:

The proposed system architecture is arranged into two loosely linked operational workflows: (a) the Offline Document-Preparation Pipeline and (b) the Online Question-Answering Pipeline (Fig. 1). In the Offline path source documents are extracted, cleaned, split into chunks mapped into vector space and indexed in FAISS. In the Online path incoming user questions are embedded with the vector model, matched to indexed passages by similarity and sent together with the query to a generative language model that writes a short natural answer.

B. PDF Ingestion and Text Extraction:

To make sure everything is correct and follows the rules the knowledge base is made from official RCPIT publications. PDF files from the institution can be hard to work with. They have things like columns, tables, pictures, at the top lines that're just for looks and strange ways of writing. The system looks at the letters and numbers removes the stuff that isn't text and finds the main part of the text. Before moving we check that the text is complete and not cut off in an unexpected way.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151183

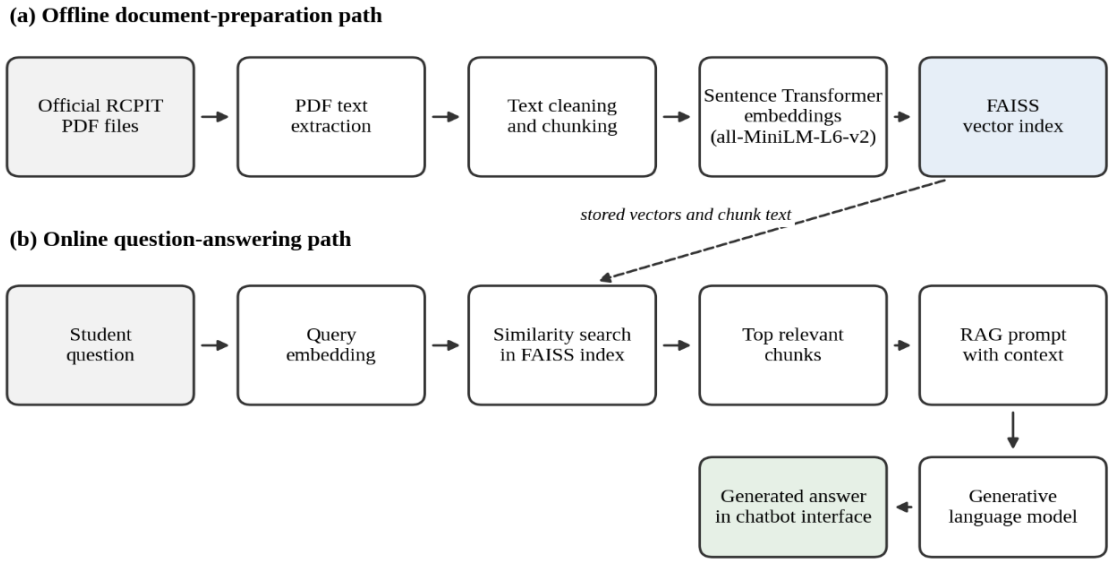


Fig. 1. Architecture of the RCPIT RAG student enquiry chatbot.

This architectural separation keeps the system maintainable. Whenever campus circulars, admission criteria or course handbooks change administrative personnel simply run the Offline ingestion routine again. The new FAISS index then instantly supplies updated evidence to the Online answer generation component without any code changes or UI redeployment.

Table 2. Functional stages of the end-to-end RAG inquiry pipeline

No.	Stage	Component	Output
1	Document Ingestion	Official RCPIT PDF publications	Authenticated source PDF documents
2	Text extraction	PDF text extraction	Raw text
3	Cleaning and chunking	Text-cleaning and splitting step	Chunks with source-document information where available
4	Embedding	Sentence Transformers (all-MiniLM-L6-v2)	Dense vector for each chunk
5	Indexing	FAISS	Vector index of the chunks
6	Retrieval	Query embedding and similarity search	Top relevant chunks for a question
7	Generation	Generative language model with RAG prompt	Natural-language answer

C. Text Processing, Normalization and Chunking:

Raw text, from PDFs often has formatting, extra line breaks and repeated header or footer lines that interfere with tokenization. The preprocessing module cleans up spaces standardizes punctuation and removes administrative banners. After cleaning the smooth text is split into parts called "chunks" that keep meaning. Chunking is essential because full academic documents cover topics while student questions usually ask about a single regulation or deadline. Segments are set to keep meaning whole while staying within the limits of the embedding transformer. Every chunk retains metadata that shows which publication and section it comes from. The answer can trace evidence fully.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151183

D. Vector Embedding generation:

To capture what students are asking and what the documents say the system uses a -trained model called all-MiniLM-L6-v2 from the Sentence Transformers library [2]. This small model that has various dimensions gives a mix of accuracy and speed. It works well on school computers. The model puts ideas (like "admission fee" and "college charges") close together in a space of numbers even if the words are different. When students ask questions the same model is used to turn those questions into numbers. This makes sure the questions and the documents are, in the kind of space so they can be compared properly.

E. Spatial FAISS Vector Indexing and Semantic Retrieval

The generated chunk embeddings are stored using the FAISS library [3]. When a student asks a question the vector embedding of that question is compared with the candidate segments in the index using either product or cosine similarity. The top-kkk segments with the similarity scores are selected. The performance of this retrieval step depends on how clean the text's how well the chunk size is set how accurate the embeddings are and how relevant the institutional documents are, to the topic.

F. Grounded Answer Generation

The text that was found is put into a system message that is sent to a language model. The message clearly tells the model to create an clear answer that only uses the given text. Also strong rules are, in place: if the text doesn't have information to answer the question the model must say no instead of making up things that are not true. This helps make sure that what is said is based on information.

VII. IMPLEMENTATION

A. Technical Implementation Environment

The complete framework is implemented in Python. Document parsing, preprocessing and vectorization use NLP libraries along with Hugging Faces sentence-transformers toolkit. FAISS is used for indexing and retrieving the nearest-neighbor vectors. A dedicated query handler loads the -computed index converts student questions into embeddings and searches, for relevant chunks. The generative model takes the augmented prompt to generate the response text, which is then displayed asynchronously in the conversational client interface.

B. Knowledge-Base Construction

The development record shows two data ingestion milestones, listed in Table 3. I first ran a preprocessing batch that read 19 PDF documents from institutions. Then I ran a processing step that split the data into 318 separate text blocks. I turned those 318 text blocks into 318 vector embeddings. Stored them in the FAISS index. These numbers only tell us how big the prototype repository is. They do not say how accurate or precise the system is or whether the domain coverage is complete.

Table 3. Documented knowledge-base repository statistics.

Item	Count	Source of the figure
PDF files identified in preprocessing run	19	Documented initial extraction run
Chunks produced	318	Subsequent refined chunking execution
Vectors in the FAISS index	318	Final prototype indexing execution

C. Qualitative Preliminary Retrieval Trial

Functional verification was carried out in a way using common campus questions for example asking which academic engineering branches are offered at RCPIT. The retriever found excerpts that were spread across separate departmental overviews and brochure pages. Functional verification shows that the pipeline works from start, to finish. However this trial was only exploratory. It did not include formal ground-truth annotations timing tests or statistical scores.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151183

D. Planned Evaluation Framework

A comprehensive academic evaluation must look at performance beyond basic conversation. We present a test framework built on a carefully selected set of student questions. These questions are paired with reference answers that have been verified by humans and come from campus documents. The benchmark includes ways of asking the same thing multi-step questions and questions that are, outside the system's scope to test how well it handles failures. Table 4 lists the evaluation dimensions we suggest [11].

E. Web Integration Strategy

The conversational assistant is designed for use on the RCPIT website. The conversational assistant may be embedded as a JavaScript widget that appears in the browser footer or it may be hosted on a separate subdomain dashboard. The exact details for how to put the assistant in place will be decided after the conversational assistant has undergone stress-testing and security reviews. Because of this no specific hosting plan, for the assistant is stated at this time.

Table 4. Proposed evaluation measures.

Measure	Evaluation Objective
Retrieval Precision / Relevance	Determine if retrieved chunks contain the required factual basis.
Response Faithfulness (Groundedness)	Confirm that every assertion in the response is supported by retrieved context.
Factual Accuracy	Verify that answers align truthfully with approved college regulations.
System Latency	Measure execution time from user prompt submission to full token rendering.
Fallback Reliability	Verify the model acknowledges missing information without hallucinating.
End-User Usability	Gauge student satisfaction regarding clarity, conciseness, and ease of use.

VIII. RESULTS AND DISCUSSION

A. Results Obtained

The project milestones show that the data ingestion, embedding, FAISS indexing and prompt-conditioned generation systems work well. Initial queries about departments were able to retrieve relevant sections from different PDF files. Repository metrics listed in Table 3(Documented knowledge-base repository statistics) were properly recorded. However since an annotated ground-truth benchmark has not yet been completed for the proposed metrics this paper maintains scientific rigor by avoiding any claims about specific accuracy percentages, latency limits or performance advantages over baseline systems.

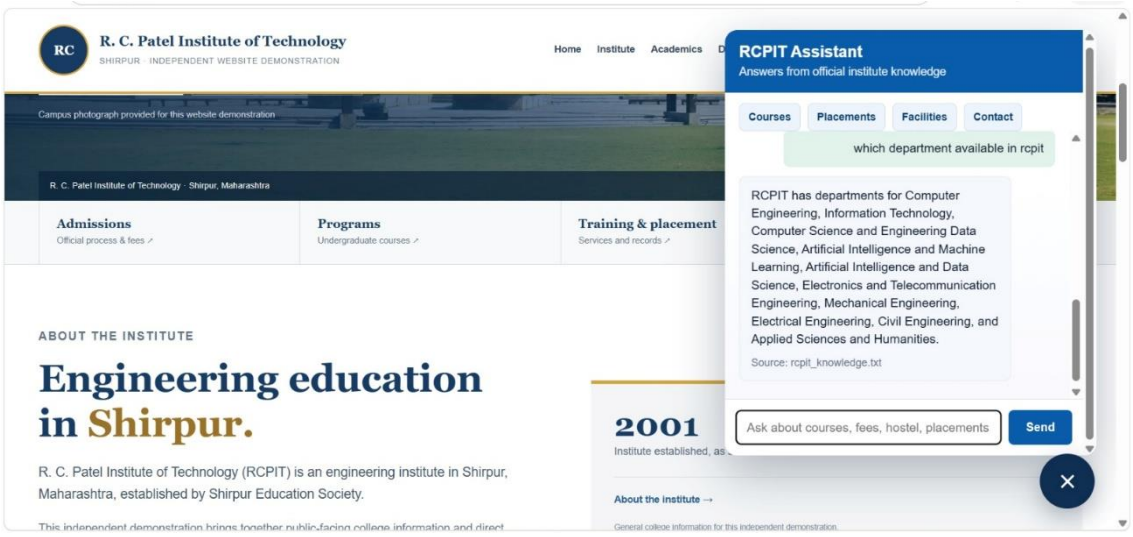


Fig 2. Result page of the portal for a sample photograph.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151183

B. Contextual Discussion

The main benefit of this document-grounded method is that generated responses stay connected to real institutional documents. By limiting the context to verified PDFs the system avoids depending on open-domain web data. This gives control over the truthfulness of the information. Also keeping track of which text chunk comes from which source supports transparency for the institution.

Still there are some risks. If the source PDFs have typos, outdated fee schedules or incorrect OCR alignment the retriever might pass along information to the generator. Questions that use -standard student slang or heavy misspellings can lead to semantic confusion, which lowers retrieval quality. Even when the context is correct generative language models sometimes leave out details or give misleading answers [6]. These issues suggest that regular document reviews, prompt adjustments and human monitoring are necessary especially, for important institutional announcements.

IX. COMPARISON WITH EXIXTING WORK

Because direct empirical benchmarks, against systems have not yet been executed on institutional datasets Table 5 presents a qualitative comparative evaluation that compares the proposed RAG chatbot with two standard query-answering paradigms: static institutional FAQ web pages and ungrounded parametric language models.

Table 5. Qualitative comparison of approaches to student enquiries

Evaluation Criterion	**Pre-Scripted Institutional FAQ**	**Ungrounded Parametric LLM**	**Proposed Document-Grounded RAG Assistant**
Knowledge Origin	Static catalog of predefined question-answer pairs.	World knowledge internalized during pre-training.	Dynamic institutional PDF repository, queried per request.
Linguistic Flexibility	Inflexible; struggles when queries deviate from cataloged wording.	High; comprehends complex natural phrasing.	High; dense vector representations accommodate semantic paraphrasing [2], [4].
Cross-Document Synthesis	Incapable; answers are confined to isolated predefined entries.	Variable; draws from parametric memory without explicit sourcing.	Capable; retrieves and synthesizes relevant context from diverse files.
Update Overhead	High manual burden to modify, add, or prune static text entries.	Extremely high; necessitates model re-training or parameter fine-tuning.	Minimal; requires re-running the offline parsing and indexing pipeline.
Source Verifiability	Pre-written; lacks granular citations to formal administrative circulars.	Non-existent; cannot reliably trace facts back to specific passages.	High; indexed segments maintain explicit file metadata.
Primary Vulnerability	Rapid obsolescence and restricted question coverage.	Factual hallucinations and ungrounded assertions [6].	Parsing errors on degraded PDFs or query-passage semantic drift.

In the world of research this work doesn't say it has created new algorithms for transformers or vector indexing. Instead it takes known systems [1]–[4] and uses them to make a real-world solution that is based on documents. This solution is made specifically for an engineering colleges needs [5]. It's not clear yet if this system works better than methods, in terms of how accurate it is, how fast it works or how engaged students are. These things will be checked using the testing method described earlier.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.151183

X. LIMITATIONS AND FUTURE SCOPE

A. *Limitations*

- The chatbot answers questions using the information found in the indexed PDF files.
- The chatbot cannot answer questions about policies or topics that are not covered in the documents.
- The chatbots performance still needs to be tested and verified using the method described in Section 4.5.
- Whenever college policies or academic schedules change the administrative staff must. Re-index the documents.
- Regular updates help ensure that the chatbot provides current information instead of outdated guidelines.

B. *Future Work*

- Adding citations: Including the names of documents and specific pages when giving answers.
- Improving table and image reading: Using tools that can understand complicated tables and old papers that have been scanned.
- Combining ways to find information: Using both strong search methods and simple word matching to make sure the right results come up.
- Testing thoroughly: Using the testing plan described in Section 4.5 on real student questions to check how well the system finds answers and how fast it works.

XI. CONCLUSION

This paper shows how we created an automated student inquiry chatbot for R. C. Patel Institute of Technology Shirpur. The chatbot was made using Retrieval-Augmented Generation. The system takes campus PDF documents takes the text from them makes the text look the same turns the text into special numbers using all-MiniLM-L6-v2 puts those numbers in a special place using FAISS and uses a language model to make answers that are true and sound like real conversation. The system gives a way for schools to help students with questions using the real information from the school. We have not yet done the tests to show how good, fast and happy users are, with the system.

REFERENCES

[1] P. Lewis et al., “Retrieval-augmented generation for knowledge-intensive NLP tasks,” in Advances in Neural Information Processing Systems (75\60. Us,” IEEE Transactions on Big Data, vol. 7, no. 3, pp. 535–547, 2021. doi: 10.1109/TBDATA.2019.2921572
[4] V. Karpukhin et al., “Dense passage retrieval for open-domain question answering,” in Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP), 2020, pp. 6769–6781. doi: 10.18653/v1/2020.emnlp-main.550
[5] C. W. Okonkwo and A. Ade-Ibijola, “Chatbots applications in education: A systematic review,” Computers and Education: Artificial Intelligence, vol. 2, art. no. 100033, 2021. doi: 10.1016/j.caeai.2021.100033
[6] Z. Ji et al., “Survey of hallucination in natural language generation,” ACM Computing Surveys, vol. 55, no. 12.
[7] K. Peyton, S. Unnikrishnan, and B. Mulligan, “A review of university chatbots for student support: FAQs and beyond,” Discover Education, vol. 4, art. no. 21, 2025. doi: 10.1007/s44217-025-00397-7.
[8] D. Ramandanis and S. Xinogalos, “Investigating the support provided by chatbots to educational institutions and their students: A systematic literature review,” Multimodal Technologies and Interaction, vol. 7, no. 11, art. no. 103, 2023. doi: 10.3390/mti7110103.
[9] M. A. Kuhail, N. Alturki, S. Alramlawi, and K. Alhejori, “Interacting with educational chatbots: A systematic review,” Education and Information Technologies, vol. 28, pp. 973–1018, 2023. doi: 10.1007/s10639-022-11177-3.
[10] R. Guan et al., “How educational chatbots support self-regulated learning? A systematic review of the literature,” Education and Information Technologies, vol. 30, pp. 4493–4518, 2025. doi: 10.1007/s10639-024-12881-y.
[11] S. Es, J. James, L. Espinosa Anke, and S. Schockaert, “RAGAs: Automated evaluation of retrieval augmented generation,” in Proc. 18th Conf. European Chapter of the Association for Computational Linguistics: System Demonstrations, 2024, pp. 150–158. doi: 10.18653/v1/2024.eacl-demo.16.
[12] 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, pp. 4171–4186. doi: 10.18653/v1/N19-1423.
[13] A. Vaswani et al., “Attention is all you need,” in Advances in Neural Information Processing Systems (NeurIPS), vol. 30, 2017.